FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Modeling and Realistic Simulation of a Dexterous Robotic Manipulator

Francisco Manuel Barbosa Ribeiro

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

Supervisor: Vítor Hugo Machado Oliveira Pinto

Co-supervisor: João Pedro Correia dos Reis

October 19, 2022

Abstract

The drive to increase efficiency in manufacturing industries is growing, as the market tends to be more and more competitive. Current technological progress has led to the expansion of automated tasks in industrial environments, however there are still manufacturing duties that need to be handled by skilled human operators, due to their high level of complexity. Further developments in cyber-physical systems can be the answer to reducing human involvement in manufacturing tasks, therefore saving human resources for high-value tasks where they are actually needed.

Recent developments in dexterous robotic manipulation technologies allowed for the design of very compact, yet capable, multi-fingered robotic hands. These can be designed to emulate the human touch and feel, reducing the aforementioned need for human expertise in highly detailed tasks. Nevertheless, these are very expensive equipment that require special attention when designing and testing control algorithms, as this can cause damage and unnecessary wear to the components.

Since the safety of the equipment, as well as safety of nearby humans, is of high priority in the design of collaborative robotic systems, such as the one composed by a dexterous robotic hand attached to a manipulator, simulation tools can be used in order to test and validate control solutions. These can be used to evaluate performance in an environment closely matching the real world, assuring correct execution of objectives without causing any danger to the integrity of the hardware. The more realist the simulation is the easier it is to transition the solution from the virtual to the real world.

The developed work focused on designing models of a Schunk Five Finger Robotic Hand, coupled to the UR5 collaborative manipulator for the Gazebo and MuJoCo simulation platforms. This allowed to assess the relative appropriateness of each of these platforms for the specific case considered in this work. A pipeline was also developed in order to allow for the use of the ROS framework on a MuJoCo simulation taking advantage of the array of capabilities of this framework and the observed suitability of this simulator regarding this task.

Resumo

A motivação para que se aumente a eficiência nas indústrias de manufatura está a crescer à medida que o mercado tende para ser cada vez mais competitivo. O progresso tecnológico atual levou à expansão da automação de tarefas em ambientes industriais, no entanto há ainda funções que têm que ser cumpridas por operadores humanos especializados, devido ao seu elevado grau de complexidade. Desenvolvimentos adicionais em sistemas ciber-físicos podem apresentar a resposta para a redução de envolvimento humano em tarefas de fabrico, reservando-se assim recursos humanos para tarefas de elevado valor onde estes são realmente necessários.

Desenvolvimentos recentes em tecnologias de manipulação robótica destra permitiram a criação de mãos robóticas bastante compactas e ao mesmo tempo muito capazes. Estas podem ser desenhadas no sentido de imitar o toque humano, reduzindo a necessidade de capacidades humanas previamente mencionadas, em tarefas detalhadas. Mesmo assim, estes constituem equipamentos dispendiosos que requerem atenção especial aquando da sua utilização para elaboração e teste de algoritmos de controlo uma vez que isto pode causar danos e desgaste desnecessários nos componentes.

Uma vez que a segurança dos equipamentos, assim como dos humanos nas imediações, é de elevada prioridade no projeto de sistemas robóticos colaborativos, como é o caso do sistema composto por uma mão robótica dextra acoplada a um manipulador, ferramentas de simulação podem ser usadas no sentido de testar e validar soluções de controlo. Estas podem ser usadas para avaliar desempenho num ambiente com características que se aproximam do mundo real, assegurando a correta execução de objetivos sem impor qualquer perigo à integridade do equipamento. Quanto mais realista for a simulação mais fácil será a transição da solução do mundo virtual para o mundo real.

O trabalho realizado focou-se na elaboração de modelos de uma mão robótica Schunk com 5 dedos acoplada a um manipulador colaborativo UR5, compatíveis com as plataformas de simulação realista Gazebo e MuJoCo. Isto permitiu avaliar qual destas plataformas seria a mais adequada para o caso específico considerado nesta dissertação. Uma *pipeline* foi também desenvolvida para permitir a utilização de ROS no controlo de uma simulação em MuJoCo tirando assim partido das funcionalidades desta ferramenta assim como das capacidades de simulação realista do MuJoCo observadas para esta tarefa.

Agradecimentos

Esta Dissertação não é o culminar do trabalho realizado apenas no período a ela associado mas sim de um percurso muito mais longo de aprendizagem que me permitiu chegar até aqui. Percurso este que foi marcado e acompanhado por diversas pessoas.

Em primeiro lugar, gostaria de deixar uma nota de enorme apreço ao meu orientador, Doutor Vítor Pinto. A sua abordagem descontraída mas ao mesmo tempo profissional foi uma grande mais valia no desenvolvimento deste projeto, assim como a sua diponibilidade, apoio e amizade em todos os momentos. As oportunidades que me proporcionou sempre que possível levaram ao meu crescimento como académico e como pessoa. O seu exemplo de ética de trabalho foi uma das coisas que me manteve motivado mesmo nas horas de maior exaustão.

Deixo também o meu agradecimento a todos os meus amigos, muitos deles que já me acompanham na vida e neste percurso estudantil há mais de 10 anos. Sem eles esta jornada não teria sido a mesma, perdendo grande parte da sua alegria e sentido.

Um agradecimento especial à Engenheira Margarida Costa por todas as vezes que me incentivou, animou e ajudou a ver o valor deste trabalho mesmo quando eu não o conseguia ver. Por todos os minutos que passou a ouvir-me lamentar de problemas com simuladores, robôs e incompatibilidades do ROS...

Por fim, não poderia deixar de agradecer à minha família, em particular ao meu Pai e Mãe por sempre me terem apoiado a seguir o caminho que eu sentisse ser o mais adequado. Por todo o esforço que fizeram para me permitir chegar até este ponto sem nunca me pedirem nada em troca, pelo seu exemplo de humildade e sobretudo por me terem tolerado nos dias em que a paciência para conversas era pouca.

Obrigado,
Francisco Ribeiro

"Really, you should always discuss the defeats
because you can learn much more from failure than from success."

Niki Lauda

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Automation of industrial manufacturing processes, through the use of robots designed to perform previously manual operations, is an area of development that has received high level of attention in the last decades. Highly structured industrial environments, such as those in automotive production plants, have been identified since the beginning as prime candidates for task automation and automatizing, being the early adopters of this technology [1].

To step towards higher levels of production automation, technologies including robotic manipulators, conveyors and Automated Guided Vehicles (AGV) have been implemented in factories. This frees human resources from labor intensive tasks as painting, part picking and distribution throughout the plant and advancement of the product in the production line. Workers are therefore typically assigned to complex tasks as quality control and precision assembly where digitization is yet to happen. However, with the escalating concern regarding production efficiency and robotic technology evolution, some of the tasks that require human expertise are being analysed in order to transfer these skills and knowledge of the operators to cyber-physical systems.

1.2 Motivation

Considering the automotive industry example, even after years of research and development on new and improved robotic systems, there is still a clear division of operations between those that have a high level of automation and those that are performed solely by operators. Some examples are the spot welding of panels, and painting, and those that are performed using touch, which are mainly associated with paint defect verification, wire fitting, trim installation and final assembly [1].

This is a common trend among different types of industries where it is usually seen a high usage of automation in machining tasks, but a low level when it comes to assembly activities [2]. It's natural that highly detailed and complex tasks are carried out manually, either because the available technology is still limited when compared to a skilled human worker, or the required

investment is too high. However, in a world where automated production is the main solution to allow for competitiveness in an increasingly demanding market, it is clear that technical advances in robotics technology, that would allow a reasonable fulfilment of the highly detailed tasks at a competitive cost, are a driving force in improving productivity in today's factories [2].

While less complex robotic grippers are already well developed and widespread in industrial applications, they don't provide the level of flexibility and dexterity required to achieve human level capabilities in highly detailed tasks. The lack of softness and compliant dynamics that are unique to human limbs makes it complicated to extend the application of less complex grippers to a variety of tasks and types of grasped objects.

Multi-fingered robotic hands can reach a very high degree of dexterity, however their hardware is not only expensive but also sensitive to damage when misused, which makes it hard to use when trying new control approaches that may harm the robot. Therefore, simulation is a very important tool in this spectrum. Due to the high dimensionality of the problem and also high dependency in contact computations the task of building a realistic simulation is not trivial and deserves some attention.

1.3 Expected Scientific Contributions

The intended goal of this work is to create a realistic simulation environment for a dexterous robotic hand on a collaborative manipulator. This is an essential tool in order for developers to test new approaches, control algorithms, train machine learning agents and much more without having the need to use the real hardware.

In this process, it is expected that multiple options are explored regarding physics-based simulation environments to find the most suitable for the task as hand. Besides this, a pipeline for controlling the robotic hand and manipulator that can be adapted to simulation and real hardware will also be developed so that the transition to the real world from the created simulation environment is as simple and with the least amount of adjustments as possible.

Thus, the present project introduces the use of state-of-the-art tools in the field of robotic control and simulation in order to digitally replicate the model of the robotic system in question. The created pipeline is also a significant advancement as it allows for the connection of the well known and feature rich ROS (Robot Operating System) framework to simulation software not natively supported by it and which may be more suited to process the particularities of dexterous hand manipulation.

1.4 Document Structure

This report is composed of five more chapters following this Introduction. In Chapter 2 some fundamental concepts regarding the task of grasping are introduced along with an overview of the paradigm of Deep Reinforcement Learning and a brief description of the components needed for the prototype. Chapter 3 presents recent methods to solve grasping and manipulation problems

found in the literature and these are concisely described. The development work is shown in Chapter 4 explaining the tasks performed to achieve the desired functionality of the prototype. The results that were able to be extracted and their analysis are in Chapter 5. Finally, in Chapter 6 some final remarks are done regarding the outcome of the work and some future investigation directions to improve on what is presented are pointed out.

Chapter 2

Background

2.1 Grasp Model

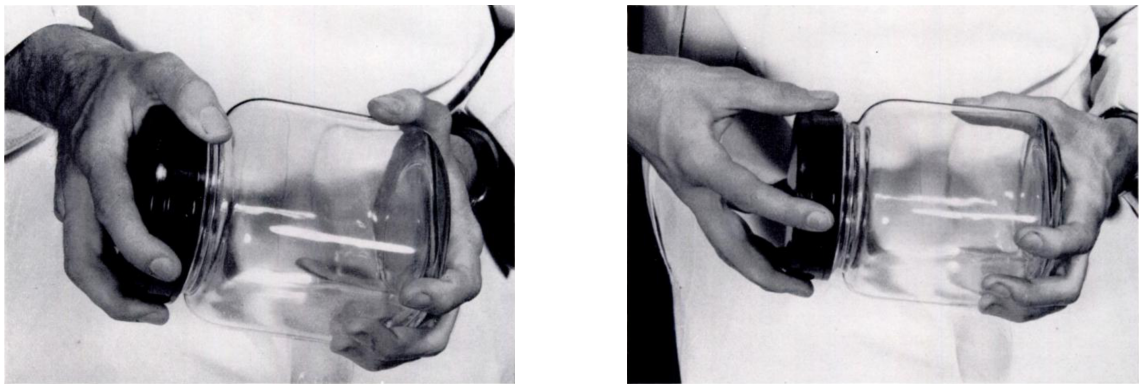
The grasping of an object refers to its immobilization due to the establishment of an equilibrium between the forces that occur as product from the interaction between the surface of the object and the one of a gripper [3]. The formalization of this concept requires some background in the analysis of human hand configuration and the representations used to capture them. That being the case, in the present section a taxonomy for classification of grasps is shown and some aspects regarding the mathematical representation of grasps and contacts are discussed.

2.1.1 Human Grasp Taxonomy

The human hand has been a very attractive subject of study even before robotics. Understanding the basic mechanisms behind human hand movement through joints and associated action of individual muscles was well understood by the 1950s, due to research in the medical and human anatomy fields. However, only from then onward, the motion of the hand as a whole started to receive attention [4].

In [4] a scheme to divide hand configurations based not only on anatomical properties but also according to a functional point of view is proposed. Grasps are divided in only two branches of power and precision grasps. In a power grasp (Figure 2.1a) there is usually a large area of contact with the object and typically the palm of the hand has an important role in the task being carried out. These kind of grasps are primarily used when a high degree of force has to be applied for example when rotating the lid of a jar or holding a heavy object and can impose a very reduced degree of movement on the object with relation to the hand itself. In precision grasps (Figure 2.1b) the tips of the fingers are used to hold the object which allows these kinds of grasps to reach very high dexterity at the expense of a lower limit of applied force.

The hands and the ways humans use them are far too diverse to restrain the classification of grasps to only two classes. Therefore a more granular taxonomy is introduced in [5], where the first branching of the classification tree is also based on power and precision grasps. This more extensive approach allows for a comprehensive consideration of task-related arguments and also



(a) (b)

Figure 2.1: Examples of the two different grasps of the same object. (a) Power Grasp. (b) Precision Grasp. [4]

a better understanding regarding the impact of objects geometric features in grasp selection. The proposed taxonomy can be observed in Figure 2.2.

In the case of power grasps it is first required to analyse the task at hand to verify if clamping is required. Only after this the object characteristics are taken into account. For precision grasps there is no other verification regarding the task and only object properties are considered to refine the configuration of the grasp. Having this, it is clear that a grasp choice should first be task dependent and only then be adjusted according to the object to be picked up. This is intuitive in the sense that the same object may be used for different purposes, each one with particular task requirements. Taking the example of the way a human opens a tightly closed jar, we first employ a power grasp to loosen the lid and once it becomes free a precision grasp is used in order to quickly rotate the lid using the fingertips [5].

The presented Cutkosky grasping taxonomy is still used to this day in the area of anthropomorphic robotic hands in tasks ranging from validation of dexterity of designed hands [6] to the design of empirical grasp planning systems through a biomimetic approach regarding the human hand [7].

Since this work is focused towards highly detailed tasks, the precision grasps will be a more suitable approach for the development of the robotic hand control.

2.1.2 Grasp Analysis

There are two important matrices used for the representation of a grasp configuration and analysis: the grasp matrix $\boldsymbol{G}$ and the hand Jacobian $\boldsymbol{J_h}$. They allow for a compact definition of the relations between velocities and force transmission from the hand joints to the contact points and ultimately to the grasped object [8].

The torques $\boldsymbol{T}_{ij}, j = 1,...,m$ applied at each of the m joints of the finger i result in the force $\boldsymbol{f}_i$ applied at the fingertip. Considering all the joints that compose the n fingers of the hand,

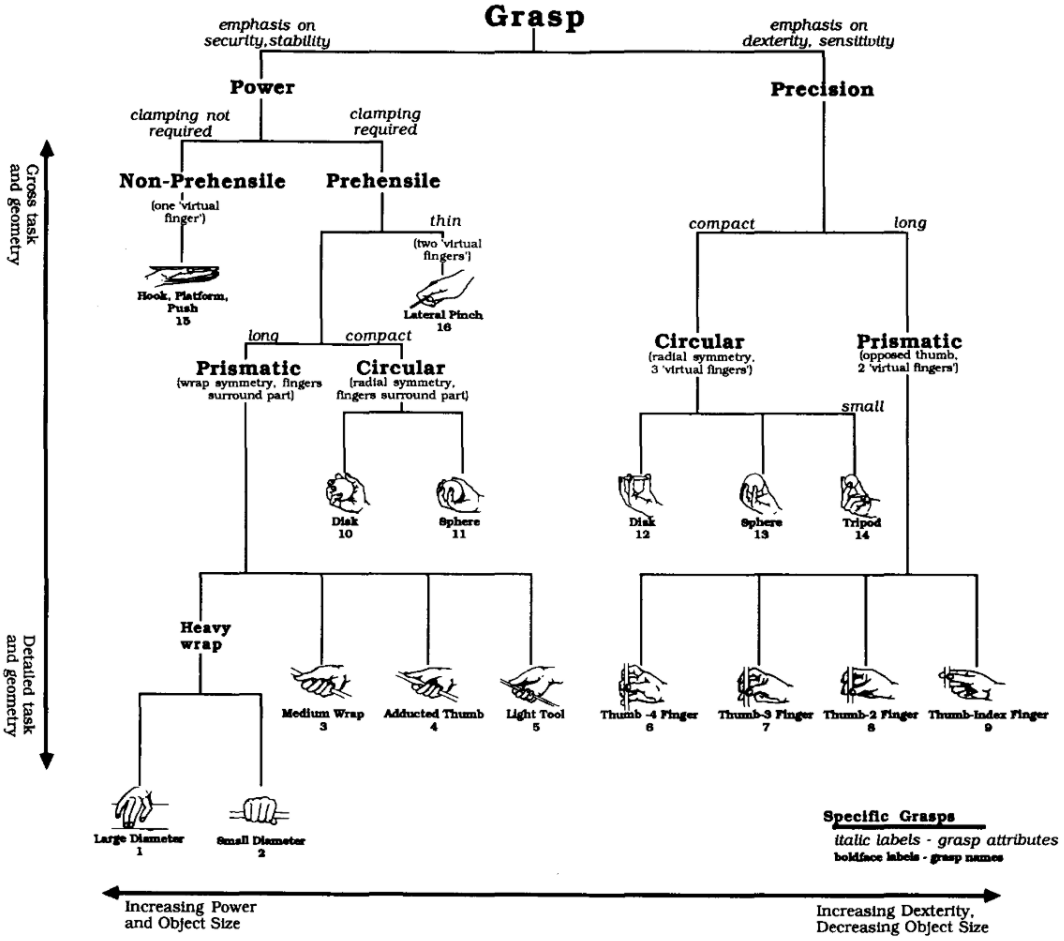


Figure 2.2: Taxonomy of manufacturing grasps [5]

the torques can be grouped in a vector $\boldsymbol{T} = [\boldsymbol{T}_{1j}^T ... \boldsymbol{T}_{1j}^T]^T \in \mathbb{R}^{nm}$. The same can be done for the velocities in the finger joints $\dot{\boldsymbol{\theta}}_{ij}$, grouping them in a single vector $\dot{\boldsymbol{\theta}} = [\dot{\boldsymbol{\theta}}_{1j}^T ... \dot{\boldsymbol{\theta}}_{nj}^T] \in \mathbb{R}^{nm}$.

Changing the reference from global to local, the forces and velocities at the fingertips can have a different representation. In this case the vector $\boldsymbol{f} = [\boldsymbol{f}_{1k}^T ... \boldsymbol{f}_{nk}^T]^T \in \mathbb{R}^{nr} (k = 1, ..., r)$ groups all the different force components applied onto the object at at the contact points, and the vector $\boldsymbol{v} = [\boldsymbol{v}_{1k}^T ... \boldsymbol{v}_{nk}^T]^T \in \mathbb{R}^{nr}$ is the equivalent for all the velocity components. The number r related to the number of different components of the forces and moments applied at each of the points of contact and depends on the considered contact model used: r=1 for point contact without friction, r=2 and r=3 for hard finger in the two-dimensional (2D) and three-dimensional (3D) spaces respectively and finally r=4 for soft finger model. A more detailed description of each of these models can be found in 2.1.3.

The movement of the grasped object can also be described with reference to its own center of mass (CM). The translational velocity $\boldsymbol{\upsilon}$ of the CM and the angular velocity $\boldsymbol{w}$ of the object with respect to CM are represented as a twist $\dot{\boldsymbol{x}} = (\boldsymbol{\upsilon}, \boldsymbol{w})^T \in \mathbb{R}^d$, where $d = 3$ for 2D and $d = 6$ for 3D. In terms of force the application of $\boldsymbol{F}_i$ at the point $\boldsymbol{p}_i$ implies the generation of a torque

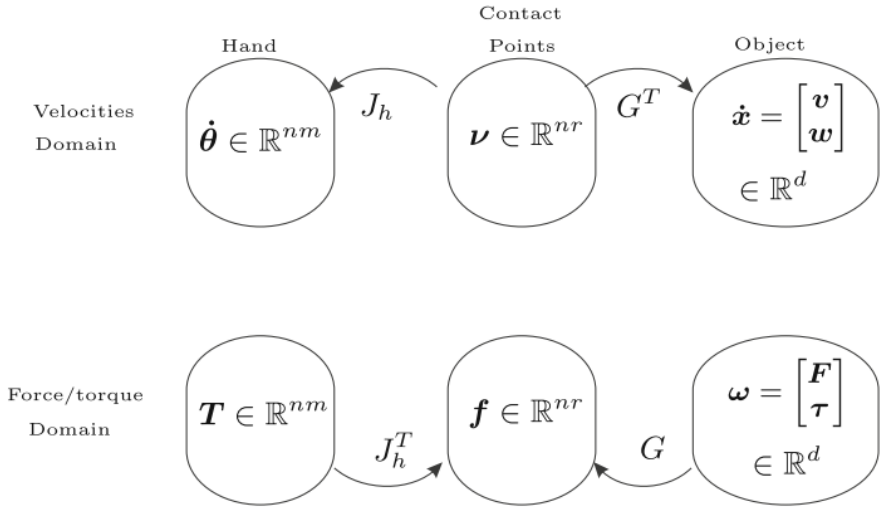


Figure 2.3: Relations between grasp force and velocity domains [7]

$\boldsymbol{\tau}_i = \boldsymbol{p}_i \times \boldsymbol{F}_i$ concerning the object's CM. Both of these vectors can be grouped into constituting a wrench vector $\boldsymbol{\omega}_i = [\boldsymbol{F}_i, \boldsymbol{\tau}_i/\rho]^T$ where ρ is a constant that can be the object's radius of gyration or the largest distance from the CM to any point of the surface of the object [9].

As can be observed from the graph in Figure 2.3, the relations between the aforementioned vectors in the different reference spaces are expressed by $\boldsymbol{G}$ and $\boldsymbol{J_h}$. The hand Jacobian $\boldsymbol{J_h} = diag[\boldsymbol{J_1}, ..., \boldsymbol{J_i}] \in \mathbb{R}^{nr \times nm}$, where $\boldsymbol{J_i} \in \mathbb{R}^{r \times m}$ is the Jacobian regarding the i^{th} finger, relates variables in the hand joint space and the fingertip space:

$$\boldsymbol{\nu} = \boldsymbol{J_h}\dot{\boldsymbol{\theta}} \tag{2.1}$$

$$\boldsymbol{T} = \boldsymbol{J_h^T}\boldsymbol{f} \tag{2.2}$$

The grasp matrix $\boldsymbol{G} \in \mathbb{R}^{d \times nr}$, on the other hand, expresses the relation of the variables in the fingertip and the object level:

$$\boldsymbol{\nu} = G^T\dot{\boldsymbol{x}} \tag{2.3}$$

$$\boldsymbol{\omega} = G\boldsymbol{f} \tag{2.4}$$

An in-depth derivation of $\boldsymbol{G}$ and $\boldsymbol{J_h}$ can be found in [8].

2.1.3 Contact Modelling

There are different models used in the literature to describe the interaction between two objects. These can be, for example, point-contact models, Hertzian Contact Models, Soft Contact Models, anthropomorphic contacts and viscoelastic contacts, among others [10]. From these models, the three most interesting ones regarding the study of grasp analysis are Point Contact without Friction (PCwoF) [11], Hard Finger (HF) [12] and Soft Finger (SF) [13]. The role of these models

is to select which of the components of the contact twists are transmitted from the hand to the object taking into account the properties of the area where the contact occurs [8].

When the contact area is very insignificant and the coefficient of friction between the surfaces of the fingertips and the object is negligible, the used model is PCwoF. Due to this, there are only considered the normal component of the contact force and the normal component of the translational velocity of the contact point on the hand. Therefore, both the components of the tangential velocity and all the three components of the angular velocity are not transmitted to the object, frictional forces and moments are neglected [10].

The HF model, also referred to as Point Contact with Friction (PCwF), is used when the friction between surfaces is significant but the contact area is very small. In this specific case the torques due to friction are minor. This leads to the transmission of normal and both tangential translational velocity components and the complete three components of the contact forces. Neither one of the angular velocity components nor the moment components are transmitted [10].

Finally, the SF contact is only used when both the surface friction and the contact are substantial enough to generate considerable friction forces and moments through the contact normal direction. In this case, the angular velocity about the contact normal and the normal component of the contact moment are transmitted, besides all three translational velocities and all three contact forces components [10].

Any of these contact models can be defined through a selection matrix $B_i \in \mathbb{R}^{l_i \times d}$ which selects l_i components of the twist and wrench to transmit [14].

For PCwoF the selection matrix will be

$$l_i = 1, \boldsymbol{B}_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.5}$$

and in the case of HF

$$l_i = 3, \boldsymbol{B}_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.6}$$

while for SF it is going to be in the form

$$l_i = 4, \boldsymbol{B}_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.7}$$

2.1.4 Restraint Analysis

There are multiple definitions of analytical measures that can be used describe a grasp. A brief description of some of the measures analyzed in the literature can be found in Table 2.1. Among all the presented measures, the two that are considered to be the most important are force closure and form closure [5]. Therefore, these will be further detailed.

2.1.4.1 Form Closure

As an informal definition, if the locking of all the joints of the hand is capable of removing all the DoF of the object with respect to the palm, then the grasp is considered to have form closure. This corresponds to the configuration of the hand as a tight cage around the object so that no movement is possible, even when friction is not considered [8].

To achieve a formal definition, a gap function $\psi_i(u,q)$ for each of the n_c contact points between the hand and the object is introduced in [8]. Null values correspond to a contact which become positive if the contact is broken and are negative if there is penetration of the object. For a given grasp the configuration of the object and the hand are represented by $\bar{u}$ and $\bar{q}$ respectively. Having this, the form closure condition can be represented by the implication

$$\psi(\bar{u} + \mathrm{d}u, \bar{q}) \geq \mathrm{d}u = 0 \tag{2.8}$$

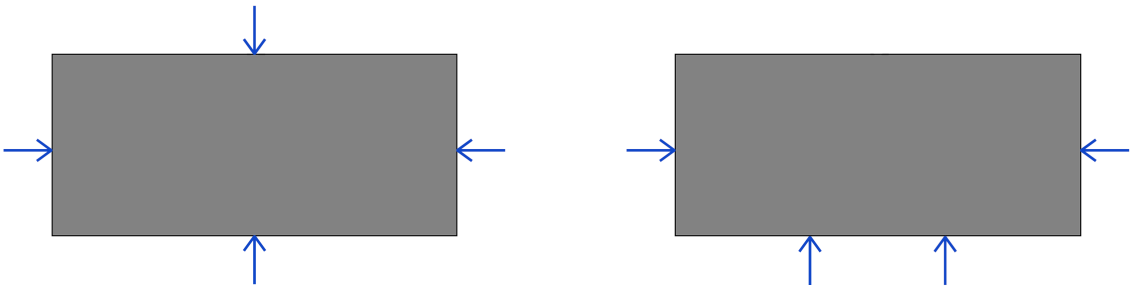
where ψ is the vector of the gap functions at all of the n_c contacts. The gap function can be expanded in a Taylor series about $\bar{u}$ generating form closure tests of various orders. First-order form closure is only composed by the first order derivative of the distance function, which means that the only relevant features are the locations of the interaction points and the directions of the contact normal. Higher order form closure is capable of taking into account the curvature of the object and the finger [15].

Focusing on the first-order form closure requirements, although not sufficient to assure this condition, the number of contact points is a necessary condition for this verification. The authors of [16] have proven that $n_v + 1$ contacts are necessary to apply form closure onto an object that has a total of n_v DoF. Figure 2.4a represents a case of 4 contact points supporting an object in a planar

Table 2.1: Definitions of analytical measures used to describe a grasp. Adapted from [5]

Analytical Measure	Definition
Compliance	How forgiving the grasp is in terms of allowing the object to move with respect to the hand due to external disturbances without breaking the grasp
Connectivity	Expresses the amount of degrees of freedom (DoF) of the object movement with respect to the hand
Force Closure	Ability to maintain the grasp under any object wrench
Form Closure	The object is unable to be moved if all the joint angles are locked and the palm is fixed in space
Grasp Isotropy	Measures the ability to accurately apply forces and moments to the object according to the grasp configuration
Internal Forces	Evaluates what kinds of internal forces can be applied to the object maintaining equilibrium
Manipulability	Checks if the fingers can move the object arbitrarily
Resistance to Slipping	Quantifies the disturbance magnitude that the grasp can sustain before slipping of the object
Stability	This verification is true if the grasp returns to the initial state after a disturbance has been removed

grasp where the contacts are placed in a way that forms a cage around the object, therefore it can be intuitively concluded that form closure is achieved. In Figure 2.4b, on the other hand, in spite of having 4 contacts for the 3 DoF form closure is not verified as none of the contacts is bounding the upward movement.



(a) (b)

Figure 2.4: Planar grasp examples. (a) Planar grasp with form closure. (b) Planar grasp without form closure.

2.1.4.2 Force Closure

Force closure allows for a reduced number of contact points when compared to form closure as it considers friction forces in the balance of non-contact wrenches [17]. Considering, for example, the planar case represented in Figure 2.5, the object is form closed in the direction of the applied contacts, however the only thing that stops it from moving in the orthogonal direction is enough friction due to a squeezing force so we consider it to be force closed if this friction condition is verified. Hence, it is important to understand how the contacts are represented in terms of friction models.

Figure 2.5: Planar grasp capable of supporting force closure

The Coulomb friction defines a limit surface in the shape of a cone $\mathscr{F}_i$ for the frictional components of the forces at the fingertips $\boldsymbol{f}_i$. This limit surface scales linearly with the product $\mu_i f_{in}$ where μ_i is the coefficient of friction at the contact i and f_{in} is the normal component of the force at the corresponding contact point. The Coulomb friction cone $\mathscr{F}_i$ is defined in $\mathbb{R}^3$ as [8]:

$$\mathscr{F}_i = \{(f_{in}, f_{it}, f_{io}) \,|\, \sqrt{f_{it}^2 + f_{io}^2} \leq \mu_i f_{in}\} \tag{2.9}$$

where the subscripts t and o refer to the two orthogonal components tangential to the contact. A visual representation of an approximation of a friction cone can be observed in Figure 2.6. Note that this case corresponds to the HF contact model as it also considers the transmission of the same three force components and none of the moments regarding the object wrench.

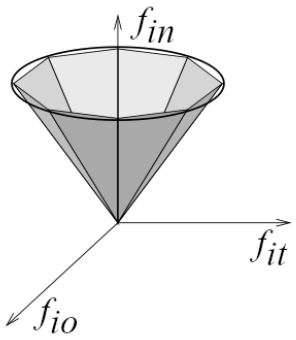


Figure 2.6: Friction cone example. Adapted from [17]

This concept can be generalized to contact models that consider more or less transmitted friction components, $\mathbb{R}^{l_i-1}$. The general friction cone $\mathscr{F}_i$ is therefore defined in the subspace of $\boldsymbol{f}_i$ as

$$\mathscr{F}_i = \{\boldsymbol{f}_i \in \mathbb{R}^{l_i} | \, ||\boldsymbol{f}_i^f||_w \leq f_{in}\}, \tag{2.10}$$

where $||\boldsymbol{f}_i^f||_w$ represents the norm of the friction components of the applied forces at the fingertip i.

For the case of PCwoF, only the normal component of the force is considered so the norm of the friction components for this case will be null, i.e. $||\boldsymbol{f}_i^f||_w = 0$. On the other hand, in the case of SF contact not only both the tangential components but also the normal component of the contact moment are transmitted. Consequently, for SF, $||\boldsymbol{f}_i^f||_w = \frac{1}{\mu_i}\sqrt{f_{it}^2 + f_{io}^2} + \frac{1}{v_i \rho}|m_{in}|$, where v_i is the torsional frictional coefficient [8].

Having established this, it can be said that a grasp has the force closure analytical property if there is a set of contact forces $\boldsymbol{f}$ that satisfy the condition:

$$\left.\begin{aligned} \boldsymbol{G}\boldsymbol{f} &= -\boldsymbol{\omega}_e \\ \boldsymbol{f} &\in \mathscr{F} \end{aligned}\right\} \forall\, \boldsymbol{\omega}_e \tag{2.11}$$

where $\boldsymbol{\omega}_e$ is any non-contact wrench applied to the grasped object and $\mathscr{F}$ is the composite friction cone for all the contact points.

2.2 Postural Synergies

The adaptability of human hands is and has always been the focus of research in different fields. The field of neuroscience is no different, and resources have been directed towards the analysis of brain and nervous system mechanisms responsible for hand control.

Initially, it was intuitively assumed that the control of each muscle and joint were completely independent during the generation of actual grasp configurations [18]. However, most conducted studies revealed the exact opposite, as the lack of individual finger movements was highlighted [19]. Taking these observations as inspiration, Santello *et al.* [20] found interesting relations concerning the coordinated actuation of some of the DoF of human hands, while grasping everyday objects. These authors applied data reduction techniques to data collected from experiments where subjects were asked to shape their hands imagining that they were grasping a given object such as as apple, a computed mouse, a wrench or a pair o scissors.

The data reduction technique applied by Santello *et al.* [20] was the Principle Component Analysis (PCA), and the results showed that 80% of the variance of the data could be accounted for just by considering the first two principle components. This demonstrated that the required dimension of the space used to describe a hand configuration can be much lower than the number of DoF of the hand itself. This new coordinate space along which the movement is carried out was called the *postural synergies* space.

The study of observed synergies starts from the acquisition of hand configuration, as the position of each of the joints, in the form of $\boldsymbol{q} = [q_1\, q_2 \ldots q_{n_q}]^T \in \mathbb{R}^{n_q}$ where n_q is the number of DoF of the hand that are being registered. Then, PCA is applied transposing the data into a different set of variables called the principal components which have minimal correlation. These principal components can be ordered so that the first part of the set can be representative of the variation present in the initial set of variables [21]. The chosen n_z principal components are in this realm referred to as synergies and are represented by the synergy matrix $\boldsymbol{S} \in \mathbb{R}^{n_q \times n_z}$ with $n_z < n_q$.

The approximated configuration of the hand can be reconstructed in terms of the joint angles by using the synergy coordinates $\boldsymbol{z} \in \mathbb{R}^{n_z}$ and the obtained synergy matrix using the expression

$$\boldsymbol{q} = \boldsymbol{S}\boldsymbol{z} + \bar{\boldsymbol{q}} \tag{2.12}$$

in which $\bar{\boldsymbol{q}}$ represents the mean values of the joint angles. This average value can be considered as a "rest position" of the hand as it is the taken configuration when all the synergy variables are set to 0.

2.3 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) consists in the combination of the advantages of Deep Learning (DL) and Reinforcement Learning (RL) for the construction of Artificial Intelligence (AI) systems.

RL is a method of estimating the optimal policy π^* that allows to reach the maximum cumulative reward [22]. This Machine Learning (ML) paradigm is based upon the exploratory nature common to the psychology of all kinds of intelligent life forms [23]. In the same way a child learns from interacting with the world, the RL agent interacts with its environment in order to refine its

behavior for the next iterations. The adjustment is made by analysing the reward associated with the taken action and the state, as it is represented in Figure 2.7.

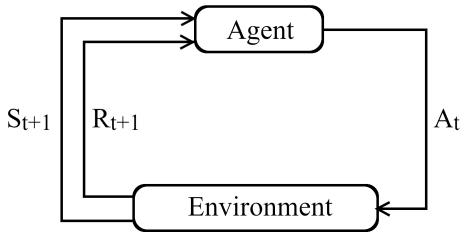


Figure 2.7: Interaction schematic between agent and environment in RL paradigm. Adapted from [24]

The use of RL to find the solution to a problem implicates the modelling of said problem as a Markov Decision Process (MDP). A MDP consists of $M(\boldsymbol{E},\boldsymbol{A},R,\gamma,H,P)$ [25], and a graphic representation of a simple example of such process can be observed in Figure 2.8. $E_t \in \boldsymbol{E}$ represents the state of the agent at each time step, A_t is the action that the agent takes from a state E_t and R is related to the reward associated with that action. The value γ is a constant between 0 and 1 and defines the relative importance of recent rewards compared to the delayed ones. H indicates the maximum number of steps that can be taken inside a single episode and T is a function of the current state and action that gives the probability of transitioning to a certain following state.

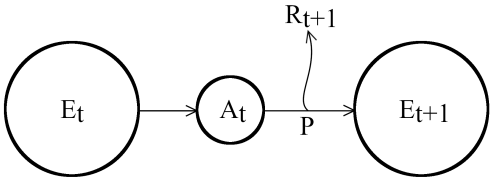


Figure 2.8: Schematic representation of a MDP

Another solution is DL that, since it is based in Artificial Neural Networks (ANN), also draws inspiration from biology, particularly from the organization of human neural networks. ANN revealed the capacity of learning important features without requiring user input, which decreases the effort in developing feature learning algorithms for input data of very large dimensions. Deep Neural Networks (DNN) are a version of ANN with higher complexity in terms of network layers which gives them an even bigger capacity of data representation [24]. Note that this automation of the feature learning process is not without its drawbacks as this creates a black-box problem. This means that the designer can not be sure of what were the learned features.

The problem with RL is its characterization for finite MDP, not allowing for a good adaptation to many real world problems where the number of states and the action space is very large or even continuous. In these situations, DRL methods make it possible to reach a solution by combining

the powerful DL scalability of DNN in very large dimension feature spaces with the capacity of generalization of knowledge of a RL agent. [26].

2.4 SCHUNK SVH 5-Finger Hand

Due to the predicted access to the hardware in the near future, for continued development, the chosen robotic gripper in that of the SCHUNK SVH 5-finger hand (S5FH). This is an anthropomorphic robotic hand with a high degree of similarity to the human hand regarding its shape, size and general appearance. As a consequence of this similarity to human hand dimensions the number of motors within its structure had to be reduced. This is the solution to ensure the speed and strength requirements while keeping the small size. Therefore the S5FH has 20 joints but only 9 DoF [27], [28].

2.4.1 Kinematic Model

Owing to the lower number of motors when compared to the joints, mechanical couplings are performed to obtain full control of the hand. This coupling is reached by means of mechanical transmissions between motors and joints that define mechanical synergies.

Let $\boldsymbol{m}$ be the vector of the 9 motor variables. Having $\boldsymbol{q}_0$ be the vector of offsets that represents joint angles corresponding to the central motor positions, the hand joints can be expressed using the following equation:

$$\boldsymbol{q} = \boldsymbol{S}_m \boldsymbol{m} + \boldsymbol{q}_0 \tag{2.13}$$

where $\boldsymbol{S}_m$ represents the matrix of the mechanical synergies.

Another approach to take into account these mechanical dependencies is to represent joint variables not in order to the motors but instead using as reference a main joint that connects to the same motor. This is done in [28] as it can be observed in the kinematic model representation in Figure 2.9. The multiplication factors used here can be extracted from the matrix $\boldsymbol{S}_m$ as the one presented in [29] by finding the relations between joints mechanically coupled to a common motor.

In spite of full detailed physical properties of the hand, such as size and mass of the individual components, not being provided by the manufacturer in the public documentation, it is known that the hand is in a 1:1 ratio to an average human hand and its weight is $1.3kg$ [29]. Some overall measurements are published and those can be observed in Figure 2.10.

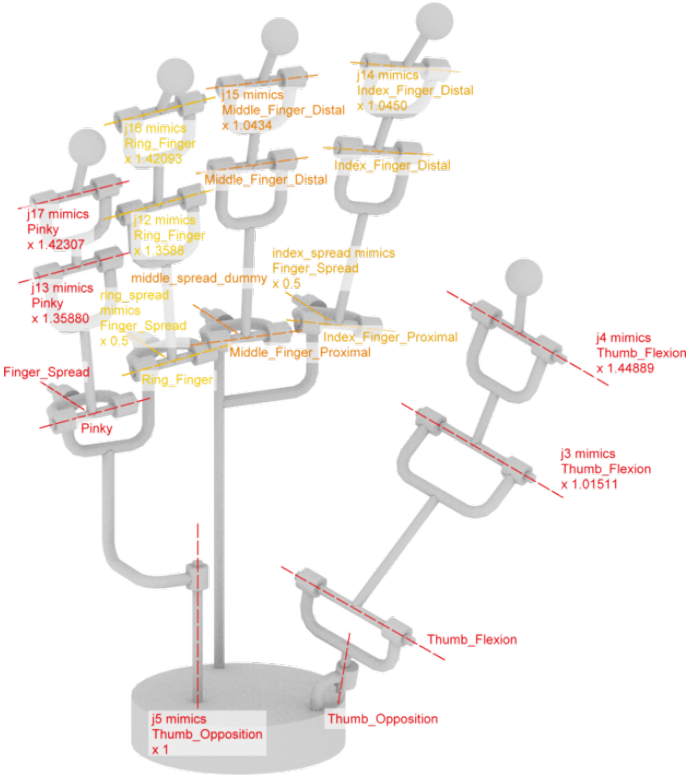


Figure 2.9: Individual joint positions in the kinematic model [28]

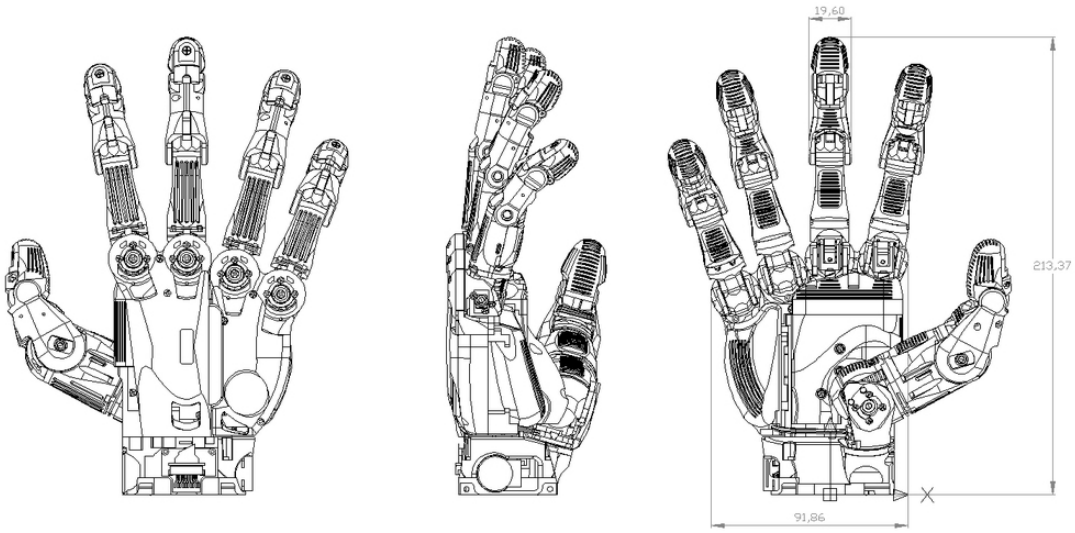


Figure 2.10: Overall measurements of the SVH [28]

2.4.2 Controller Description

The wrist integrates the electronic components used for the control of the hand. Embedded in it there is the Mecovis S5FH-Controller which has been designed to control the S5FH in particu-

lar. This controller contains motor drive circuits, sensor interfaces for position feedback, motion controller and a serial communication interface to a host [28]. A field-programmable gate array (FPGA) is used to implement the firmware's functionality wich provides high flexibility and performance.

Regarding the control itself, it is performed by cascading a position controller and a current controller. The position controller is implemented as a Proportional–Integral–Derivative (PID) controller with anti-windup. The reference signal for this controller to follow is set via the communication interface by writing to an internal register of the FPGA and its output is internally routed to the reference input of the current controller. The current controller is introduced in the form of a Proportional-Integral (PI) controller with anti-windup and its output is connected to a PWM generator that drives a motor.

The constants that define the behavior of the controllers described above are also written via the communication channel. The Robot Operating System (ROS) driver for the S5FH already contains the values and programmed protocols to send them as packets to the Mecovis therefore its usage when dealing with the real hardware streamlines the process of setting up.

Chapter 3

Literature Review

As humans, the manipulation of an object by using the great tool that hands are actually starts in the secure grasping of that object in a position that allows for the manipulation that is planned for the following instants. Based on a review of the existing literature on this subject, this chapter focus the developed solutions for the problem of automatically generating and executing a grasp. The mentioned solutions take into account hand restrictions and environment properties and the manipulation of objects by changing from one hand configuration to another, without loosing the grip over the object.

3.1 Grasping

The problem of automatic grasping of an object by a multifingered robotic hand is widely regarded as a complex task. Thus, it is many times solved by using assumptions and simplifications in implemented solutions. It can be considered as a combination of several demanding tasks, such as the computation of the contact points, force optimization, grasp stabilization, among others [30]. Examples regarding the different tasks are presented in the following sections.

3.1.1 Grasp Synthesis

Although humans can intuitively shape their hands to pick up an object from a surface just by seeing or feeling it, the calculation of grasps, when it comes to a dexterous robotic hands, is not a trivial task. Besides this, it is a task whose complexity scales with the amount of DOF of the gripper and the intricacy of the object's surface. In theory, there is an infinite amount of ways that an object can be grasped. However, the one that leads to the most stable grasp must be found, as in the grasp that tolerates the most external influence, and also the one that is more suitable for the manipulation tasks that are required in the subsequent steps. Therefore, an advanced algorithm to solve the grasp synthesis problem is the subject of many recent research publications which also reveals it as an open problem. Some authors have an emphasis on the hand configuration planning while others pay more attention the planning of the applied forces to realize the grasp, both being of the upmost importance for the quality of grasps.

Regarding hand configuration planning, in 2018, Gal *et al.* [31] introduced a data fusion method to combine observations from a stereoscopic camera and a Kinect sensor [32] for object shape analysis. The objects are classified into three categories: sphere, parallelepiped and cylinder, by using a heuristically defined decision tree applying principles similar to the ones used in fuzzy inference process. Then, this information is used to empirically define a grasp configuration, by selecting a suitable grasp from a taxonomy similar to the one presented in section 2.1.1.

A different approach is followed by the authors of [33]. Here, the authors address the problem of the computational cost of grasp generation for a multi-fingered hands that, due to its high dimensionality, is generally not appropriate for real-time implementations. The grasping pose starts from a configuration obtained from a database that contains data related to contact points used by a less complex parallel gripper. The object is grasped using this pose and it evolves by splitting the fingers to maximize a Contact Point Optimization (CPO) condition and the relative position of the palm is also shifted according to a Palm Pose Optimization (PPO). This dual stage optimization process is illustrated in Figure 3.1.

Figure 3.1: Illustration of the finger splitting process. From left to right, hand starts in a generic position selected from the database, fingers are separated using CPO and finally palm position is adjusted by PPO. Adapted from [33]

Vahrenkamp *et al.* [34] presented a planning algorithm based on the analysis of the object's skeleton. The 3D model information of the object is used to create its skeleton, which is analysed and segmented, in order to find sections which would be suitable for a stable grasp. The surface of the object, associated with each of the segmented parts of its skeleton, is also taken into account, in order to find the most adequate grasping configuration for robustness. This "part-based" approach to the grasp planning is similar to the human process that breaks objects into smaller parts [34], as can be observed in Figure 3.2. The mentioned method produced results that were superior to a regular surface-based grasp planner as it had faster execution, better force-closure metric and higher robustness score across varied objects and different tested robotic hand hardware.

Learning based approaches have also been gaining popularity in the field of grasp configuration planning, as technology advances and modern computer hardware make these approaches viable. Yao *et al.* [35] propose an algorithm based on human experience. It classifies the objects into different types, using a set of features. Each type has an associated optimal grasp configuration that is obtained by analysis of human behaviour, faced with different objects and using this data to train a neural network. A similar solution is suggested by Zhao *et al.* [36], with the key difference that each type of object has not one but a set of possible grasps. This set is then evaluated by a grasp evaluation network based on the force closure metric that evaluates candidate groups and chooses the optimal set of contact points.

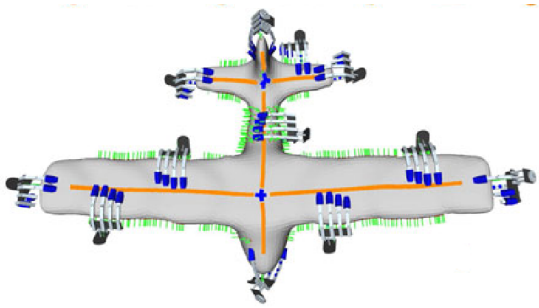


Figure 3.2: Results of the skeleton-based grasp planner. Skeleton represented in orange and possible directions of approach in green. Adapted from [34].

The solution proposed by Xue *et al.* [37] to the trajectory generation of a arm-hand robotic system, in order to grasp a stationary object on a table, is an improved deep reinforcement learning strategy for end-to-end training. The problem is formulated as a cost function that states the grasp cost at each of the states between the current position and configuration of the robotic hand and the final grasping of the target object. The trajectory controller is trained in order to minimize this cost function. The algorithm is also successfully validated in a physical system, demonstrating its obstacle avoidance ability. However, in this case, to reduce the state information complexity, the extracted features are related to the occupancy of the 3D mesh beneath the hand and it would be interesting to use a deep neural network, in order to be able to use high-dimensional state information as an image of the environment.

As previously stated, the calculation of the required applied forces to realize the grasp is equally important as the contact points. In [30], three Grasping Force Optimization (GFO) methods, capable of calculating the optimum force to be applied at each of the contact points, are discussed and compared. All of them are based in the Coulomb law that is used to define the friction cones presented in Section 2.1.4.2. This study leads to the conclusion that, when using HF contact model, the non-linear method has a better compromise between precision and computation time, while maintaining consistent results, even when the location of the contact points change due to slipping. The linear method shown to be more efficient in the particular case of low number of contact points.

While the authors of [30] use a method where the forces were calculated in the fingertip space and had to be then transposed to the joint space, Jia *et al.* [38] use another where the joint torques are directly analyzed, avoiding this process. This allows for a more extensive linearization of the problem improving the efficiency of the solution. Jia *et al.* [38] also introduce a cost function that considers the sum of the squares of the applied torques, which guides the system to use the lowest possible torques. The reduction of the torques has the benefit of lower power consumption and also minimizing the possibility of accidental deformation of the grasped object.

Islek *et al.* [39] present the concept of "safety margin" in human grasping to reduce forces to as low as possible. The force safety margin is defined as the extra strength that humans apply to avoid slippage of the object, above what is strictly necessary to maintain the grasp. The approach

presented by Islek *et al.* [39] is based on a fuzzy controller that analyzed the local normal force and local tangential force at the slip point to generate safety margins for object lifting tasks. This work achieved results comparable to human performance.

Deep learning methods have also been studied regarding the force optimization method. Deng *et al.* [40] proposed the training of a deep neural network using the readings of tactile sensors as input data, towards online slip detection. If slip is detected either because of lack of friction, or due to external disturbances, the applied force reference is increased and the force controller adjusts the configuration, thus stabilizing the grasp.

Some authors considered the concept of synergies to be an asset, regarding the optimization of the grasp synthesis process. In 2012, Rosales *et al.* [41] presented a method capable of generating, in parallel, suitable hand synergy configurations for object grasping and contact forces. Synergies are also leveraged in the work presented by Liu *et al.* [42] to reduce the feature space used to represent the status of the hand, enabling the usage of gradient-based grasp planning, applied to a Q distance that is a differentiable measure of grasp quality introduced [43].

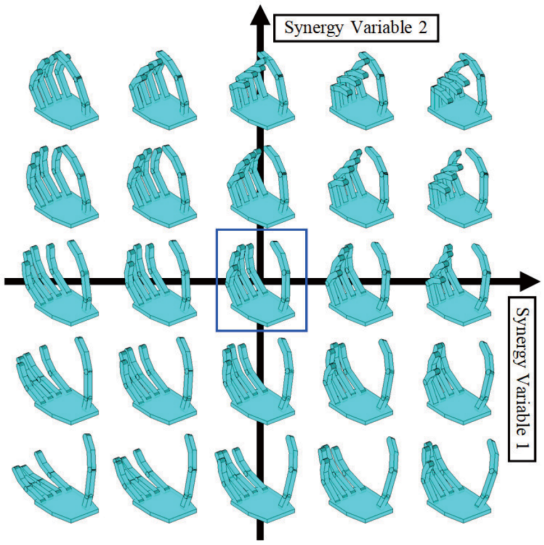


Figure 3.3: Example of hand configurations using a 2D synergy space. The "rest" position is in the center of the graph. [42]

Ficuciello *et al.* [44] present another solution that uses synergies to map human grasp demonstrations onto robotic hands. First, the object is analyzed and compared to entries in a database, through a computer vision system. The database provides an initial valid grasp pose for the associated object and this configuration is executed on the robotic hand in the synergy space. From this initial configuration, the grasp evolves in the direction of maximization of two indexes. The first one refers to the distance of each of the fingertips to the center of a virtual object defined by the fingertips used in the grasp. The second index refers to a correction term based on the force-closure metric. A threshold is imposed on the motor currents to avoid excessive force that would damage the object.

Usually, human demonstrations are used to construct the databases for grasp selection using gloves with sensory functions or vision systems [45]. This can induce errors in the mapping of the configurations to a robotic hand as the structures can be very dissimilar. Jiang *et al.* [46] created a synergy space by direct teleoperation of a robotic hand which allows for the capturing of the human experience, while recording the actual positions of the robot, avoiding the need for position mapping. Later, the data is grouped into similar clusters using K-means algorithm and the posture of the dexterous hand is generated using the synergies in a specific cluster reducing even further the subspace that needs to be considered in the search for a suitable grasp.

3.1.2 Grasp Stabilization

After selecting a grasp configuration, there is the need to maintain it in a stable state, in order to make it resilient to inaccuracies in the considered model and external disturbances. After reviewing the literature related to this topic, the two main mechanisms for grasp stabilization are force control and impedance control. Force control uses real-time force measurements directly to close a feedback loop, in order to precisely control the interaction force with the object. Impedance control uses an indirect approach. The latter technique hinges on the definition of the interaction with the object as a mass-spring system. It then uses the measured force and manipulator position to adjust the mechanical impedance in order to perceive a target impedance [47].

Regarding the force control mechanisms, Sadun *et al.* [48] used a linear control strategy in the form of a Proportional-Integral-Derivative (PID) controller to adjust the forces applied on the object, using low-cost Force Sensitive Resistors (FSR) to close the feedback loop. This work also focuses on a novel enclosure for this kind of sensors, which increases their performance making them an interesting alternative. Having used linear control to control a highly non linear process, the system presents the expected associated problems, such as a lack of adaptability to different conditions of operation.

A different option for providing feedback in force control is the slip detection of the object. In Morita *et al.* [49], a small scale design of a Laser Doppler Velocimeter (LDV) is introduced to enable the usage of sensors built on this principle of operation in robotic dexterous hands. The sensor was validated by applying it in a force control scheme as the single feedback source and the grasp stability was verified even without any direct applied force measurements.

The previously described method considered the hand-object as a complete system in the control computation. A different view of the problem is proposed by Veiga *et al.* [50], inspired on human experience, and where the individual fingers are considered for control purposes using tactile information. This modular perspective presents satisfying results towards grasp stabilization in static conditions, and is indicated as a viable solution for low level control in manipulation methods for which there is a higher level layer that controls the relations between the fingers.

Deng *et al.* [40] present a framework developed with the main objective of provide a robotic hand with the capability of automatically adjusting the grasping force, according to slip observations, object material and contact force from tactile data. This is proposed without the knowledge of object properties such as their shape, size or weight. First step is the detector of contact event

and object material. This is based on a trained deep neural network that takes as inputs the data from the tactile sensors. Then a force estimation module infers the contact force from the tactile information as the result is used as feedback to a force controller of the hand.

Attempts to use synergies to simplify force control strategies have also been carried out. In 2019, Ortenzi *et al.* [51] proposed a method that evaluates the benefit of synergies in this sort of applications. Two alternatives where tested. The first one uses synergy based contact force control where due to the coupling between different fingers is not possible to control each of the individual applied forces but an average force is tracked by the controller. The second alternative uses synergies for the initial grasp of the object and after the grasping is achieved the control is transitioned to individual finger control making it possible to follow individual force references for each of the contact points.

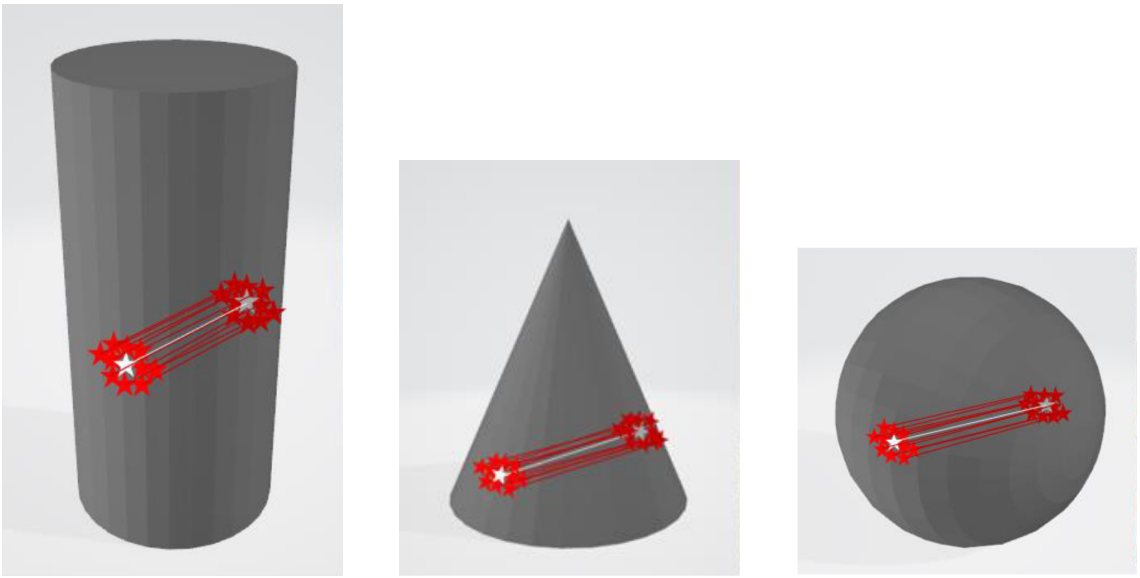
Force controllers can present position tracking of the object that is far from optimal, as they are mainly concerned with maintaining stability though force equilibrium and not with position control. Yajima *et al.* [52] introduce a solution to this problem through a vision system that estimates the CM of the grasped object associated with a position controller that works in parallel with the force controller maintaining the object in the desired location.

Similar to force control of a complex dexterous robotic hand, when there is the need to simultaneously control a high number of DOF, in impedance control this problem is also verified. In the work of Pertuz *et al.* [53], the inherent parallel computation capacity of Field Programmable Gate Arrays (FPGA) is highlighted and used to reduce the hardware complexity of typical implementations of impedance controllers. The use of FPGA presents a clear advantage, when compared to regular microcontrollers, in the tests performed in [53]. A System on Chip (SoC) approach is also tested presenting comparable capacity to FPGA but with a smaller footprint and reduced power consumption. The drawback of the more specific purpose implementations are higher development costs as they take more time to realize and debug. This proposal is further developed in [6]. Here, an FPGA based impedance controller is applied to a 7 DoF robotic hand and a impedance parameter tuning method is presented making use of genetic algorithms such as Particle Swarm Optimization (PSO), which were presented as a novelty when applied to this area. The conducted test scenarios revealed that using motor currents as feedback to the controller is not sufficient particularly in the pre-contact stage where forces are very low. Authors suggest the use of a position controller in this stage as future work and make the transition to impedance control after contact is established.

Considering the human physiology example, stiffness controllers where implemented by considering the inherent Configuration Dependent Stiffness (CDS) and Common Mode Stiffness (CMS) in [54] and [55]. This method proved to be functional by just using articulation position as the inputs of the stiffness controller. However, it is stated by the authors that the inclusion of force feedback or visual perception of the relative position of the grasped object would be beneficial, as the presented implementation only works for fingertip grasps and objects with reduced shape complexity as spheres or cubes.

Regarding learning strategies, the authors of [56] present a combination of synergies and Convolutional Neural Networks (CNN) that reproduce the soft behaviour of human limbs. The training of the stiffness control system is performed using only one example of each grasping motion from human demonstration reducing the needed high volume of data when compared to Deep Learning methods.

Some authors considered the application of learning-based methods not only to individual parts of the grasping process but to the whole stack of procedures instead [57]–[59]. Song *et al.* [57] use a hierarchical structure of four neural networks, each one related to an object location, grasp points decision, grasp posture selection, and trajectory definition, respectively. This method is tested on a setup using a Shadow Hand Lite and a RGB-D camera with depth perception, which is able to reach a success rate of 100% for most of the tested objects. In the work from Merzic *et al.* [58], contact feedback is used in order to improve the grasp quality under noisy object pose sensing. This contact information is used to train DRL agents capable of achieving control policies that can generate robust grasping, even under uncertainty of object model. It was also verified that training under noisy conditions improved the robustness of the grasping policy, even if the contact information is not considered. The work from Kim *et al.* [59] demonstrates a method consisting of a CNN, that processes the input received from a RGB-D sensor and classifies the object into 1 of 10 predefined types. The object information and hand characteristics are used to derive a neural network based objective function that derives the optimal grasp position, as it can be observed in the examples in Figure 3.4. Obtained results show the superiority of this method in terms of stable grasp success rate, when compared to force-closure across all classes of considered objects.



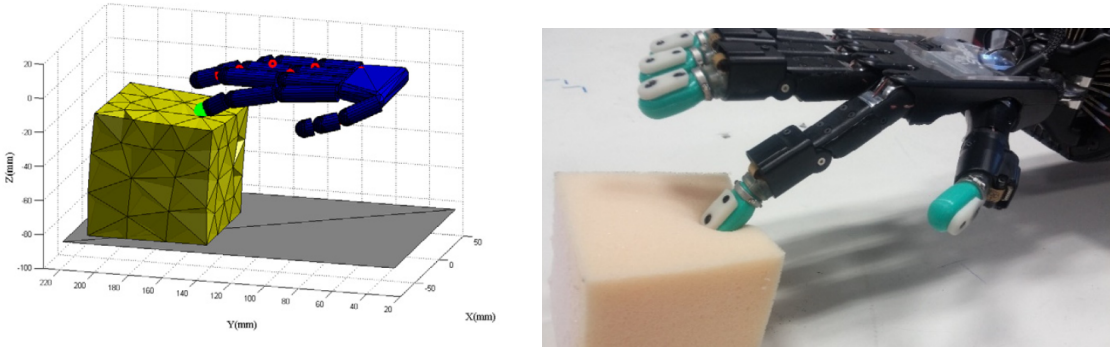
(a) (b) (c)

Figure 3.4: (a), (b), (c) Highest scoring pairs of grasping points for a cylinder, cone and sphere respectively. Adapted from [59]

3.1.3 Grasping Deformable Objects

Most of the current research on automatic robotic grasping is dedicated to rigid objects [60]. However, further study regarding the application of these techniques on non rigid objects would be beneficial to industrial applications, as it drives the capacity of robotic systems even further in the diversity of tasks that they can perform [61].

In 2017, Zaidi *et al.* [62] developed a method to grasp deformable objects in a model-based approach. The main highlight of this work is the possibility to consider very large deformations of the object, which is valid since a second-order model is used for the representation. This model is used for simulation (Figure 3.5a) and validated by experiments with a robotic hand with tactile sensors depicted in Figure 3.5b. A HF contact model is used, and the tangential contact forces are taken into account in the grasp stability analysis increasing the precision of the obtained result. Reference contact forces are calculated iteratively as the process of finger closure occurs, guaranteeing minimum final forces avoiding permanent damage to the object.



(a) (b)

Figure 3.5: Validation of the contact model. (a) Simulation of deformation. (b) Cross validation of deformation in the real world. Adapted from [62]

Nadon *et al.* [63], on the other hand, introduces an algorithm that not only generates grasping points for stable grasping, but also selects these points to allow for a maximum shape control freedom of the grasped object. The main drawback is the high dependency on a contour detection computer vision algorithm for the correct operation of the system, which is complicated to achieve without relying on computationally expensive learning systems, and large volumes of data. The authors refined the system and presented an updated version in [64]. This update was mainly focused on the grasp point selection, using heuristic criteria to quickly remove some grasping configurations that deemed inadequate for the reshaping task.

3.2 Manipulation

Manipulation is considered as the adjustment of the position and orientation of a grasped object, by using the degrees of freedom that the articulations of the hand-arm system provide. This

must be performed without loosing control over the object, as in dropping it. Hence, the planning of a manipulation duty comprises the selection of a state sequence from the robotic system that describes the movement that will fulfill said duty. In literature, methods to solve the problem of manipulation can be found based in different starting points: classical control methods, synergy based control and even some artificial intelligence systems.

An important concept of "virtual contact points" is established in [65]. These are just reference points of contact, that are located in the interior of the object. The difference in distance from the actual contact point and the virtual contact point is proportional to the applied force on the object. This allows to manipulate the object without a visual model of the object's surface. This concept is used by the same authors in [66] to develop a dexterous manipulation framework of unknown objects.

For manipulation purposes, synergies alone sometimes create more problems than they can solve. The coupling between joints of the hands makes it impossible to perform certain movements that are required for complex dexterous manipulation of objects. For this reason, Higashi *et al.* [67] proposed a Functionally Divided Manipulation Synergy control (FDMS) that allows some of the fingers to remains static while holding the object while other manipulate it. The usefulness of this algorithm is demonstrated by operating a pair of scissors and a spray bottle that needs a set of fingers to safely secure the bottle while another finger actuates the spray lever. Fingers that are grouped together in the same set are operated by the same synergies activation.

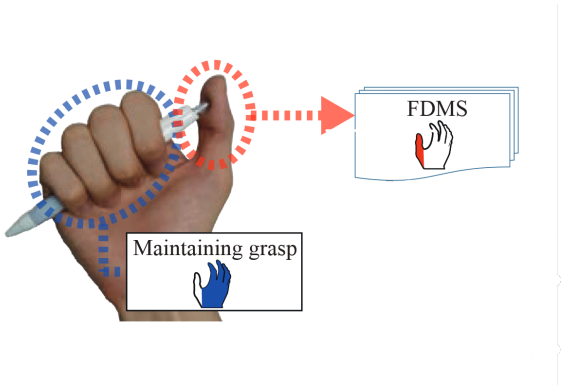


Figure 3.6: Overview of the FDMS method. While four of the fingers are fixed and hold the body of the pen, a single finger is assigned the function of clicking the button. Adapted from [67].

Due to the even higher difficulty to obtain an analytical analysis for this task, as static analysis don't make sense in a manipulation scenario, learning based approaches are very popular and are gaining even more popularity in recent years [68].

Regarding the manipulation of rope like objects, Nair *et al.* [69] have elaborated a learning based approach that is capable of relating the behavior of the non rigid object observed by a vision system with the actions of the robot using large volumes of data acquired autonomously by the robot. This exploratory behavior allows the robot to obtain a model of the object's physical parameters, and use it in a low level control scheme. A high level control is also implemented

by using human demonstration to show the robot different keypoint states of the rope, which the robot is able to mimic using its low level controller.

In 2019, Jain *et al.* [70] published a method for dexterous manipulation learning from visual and tactile information in simulation. This method used visuomotor data for the training of policies in a trial and error scheme, as these sensors can be incorporated in the robotic system and don't implicate the alteration of the environment where it is introduced. First, only camera data is used and the method is successful in simple environments without occlusions and uniform lighting. Further, the extension to tactile data improved training times, as well as its performance in situations with high degree of occlusion.

Andrychowicz *et al.* [71] used reinforcement learning to automatically learn in-hand manipulation without any human demonstration, by using only visual feedback to perform state estimation through a convolutional neural network. The training is performed in simulation, and later the learned policy is transferred to a real setup with success. Interesting observations were made in the sense that even without any human demonstration, human traits were observed the way the hand motion took place.

While the preferred method for training reinforcement learning systems is simulation, as it can be executed faster and without any risk of damaging the physical systems, sometimes a large effort must be made in order to transfer the learning from the virtual to the real environment [72]. For this reason, the authors from [72] perform the training of the policies associated with DRL agents in the real environment, instead of simulations. Attempts are made to train without any models of the task or the object and multi-fingered manipulation skills are capable of being acquired in 4 to 7 hours of training time.

The combination of synergy grasp representation and deep reinforcement learning has also been studied for manipulation applications. In [73], a synergy-based DRL policy is generated for the control of anthropomorphic robotic hands with multiple DoF. Human demonstrations are used for the training of the DLR, in order to generate natural hand poses based on a synergy space representation of the grasping data. The dimensionality reduction of the initially obtained data increases the efficiency of the deep neural network. The synergy based approach is reported to reach 88.38% success rate for object relocation tasks, while a maximum of 50.66% is obtained for the same DRL approach without synergy representation. Deng *et al.* [40] use a similar approach, presenting a learning based method for multi-fingered hand control for in-hand manipulation. The representation of the learning data is also done through the synergy space to achieve dimensionality reduction as the acquired data is very complex and would cause difficulties in the training of the DRL agent.

Ruppel *et al.* [74] extended the bio-mimic of human motion past the robotic hand to the robotic arm paired with it. The learned behavior is combined in control generation with an online trajectory optimization that works as a gap filler between the reference poses that the robot is instructed to follow.

Design of reward functions can be a hard task, since it is difficult to accurately predict the value of a state, particularly if it is an intermediate state that is needed for the fulfilment of a

wider task. Orbik *et al.* [75] introduce automatic generation of a reward function, avoiding the need to design one, which can be a quite complicated in complex dexterous manipulation tasks. The reward function is drawn from demonstrations of desired behaviours and also some random samples statistically inferred from the collected demonstrations. This is done in order to get a better coverage of the state-action space, as well as increasing the reliability of the reward function, even for regions that were not included in the demonstrations. For validation, the MuJoCo physics engine is used [76].

Finally, Saeed *et al.* [26] present a benchmark on different DRL techniques: Deep Deterministic Policy Gradiente (DDPG), Deep Deterministic Policy Gradient with Hindsight Experience Replay (DDPG+HER) and Proximal Policy Optimization (PPO) applied to continuous control tasks such as the pick and place of an object using a robotic arm an in-hand manipulation of cubic objects with a Shadow Dexterous Hand. Tasks were carried out in the MuJoCo physics simulator [76]. DDPG+HER was the most successful candidate reaching 100% success rate in almost all tasks.

3.3 Motion Planning

To execute a desired movement with a robot such as the ones in which this work is focused upon, it is necessary to calculate its trajectory, i.e. to plan the movement.

The trajectory planning generates the input references for the low level controllers to follow and this will lead the manipulator to follow the planned movement. Note that a trajectory may define speed and acceleration besides the position reference.

Tools have been developed for the purpose of motion planning and execution and have reached a high level of maturity. MoveIt is a ROS library that is a state of the art tool for robot motion planning [77]. It is a very widely used software package due to its open source license, and implements the latest technology for motion planning, manipulation, kinematics, control and navigation [78]. Another advantage of MoveIt is its integration with the rest of the ROS framework such as the Gazebo simulator and RViz visualization software.

ROS itself is an open source framework for robot programming. A collection of libraries and tools support the implementation of software for robot control making the development process easier and faster [79]. ROS has a built-in messaging system that works in a publish/subscribe pattern without the user needing to worry about the implementation of such communication middleware. This results in clear interfaces meaning that the same code can be used towards many different kinds of robots. Some other arguments to be made in favor of ROS are the excellent documentation, frequent maintenance of libraries and active community. However, the lack of backwards compatibility between ROS versions sometimes leads to difficulties to overcome during the development such as adjusting the Linux version used or even building some packages from source.

MoveIt can be configured to use a number of different inverse kinematic solvers. The preinstalled packages that perform this function are targeted at serial kinematic chains so a different

solution must be searched in order to solve motion tasks on more complex robots such as multi-fingered robotic hands. BioIK is an open-source software package fully integrated in ROS and MoveIt that is capable of solving inverse kinematics for robots with arbitrary kinematic trees [80]. This package allows for the description of motion goals using a combination of individual sub-goals, each with an associated weight that relates to its priority in the computation process. This empowers a user to specify, for example, the position and orientation of each one of the fingertips in a dexterous robotic hand in order to grasp an object, making it very interesting in the scope of this work.

The mentioned above RViz visualization software, as the designation suggests, is a 3D visualization tool with graphical interface part of the ROS ecosystem. It allows for the inspection of any robot model and its functionality can be expanded by means of plugins. As an example, with the MoveIt plugin commands can be sent directly from RViz to the MoveIt interface to request trajectory for a certain motion.

3.4 Robotic Simulation

Simulation is a very important tool in robotics, allowing tests to be executed without access to the physical robot. The usage of simulation before transitioning to real hardware carries benefits as there is no harm or wear to the hardware which can be very costly and also likely to happen in a phase where the user is quite unfamiliar with it or, in the case of RL training when the agent's behavior is hard to predict. One can also run a wide range of simulations in parallel which can be beneficial for learning based control approaches.

In [81] popular simulation environments are tested and compared. The authors recommend Gazebo when developing for simulation and real systems due to the homogeneous ROS connection that allows the control to be used in the virtual and real world with minimal changes. It also performs best in trajectory tracking tests similar to the use case presented in the present work. MuJoCo performs second best in this use case and is recommended for RL training due to its popularity in OpenAI Gym simulations [82]. As these two platforms show the most promising results they will be analysed further below.

3.4.1 Gazebo

Gazebo [83] is an open source status 3D simulator that is integrated in the ROS framework, which explains its popularity.

It is equiped with 4 different physics engines: Open Dynamics Engine (ODE) [84], Bullet [85], Dynamic Animation and Robotics Toolkit (DART) [86] and Simbody [87] allowing for an easy adaptation to the user's needs. Empowers roboticists to have as detailed of an environment as they desire as it can simulate every aspect of the real world from gravity to inertias and ever sensor noise. Similar to ROS, Gazebo also benefits from the attributes of open source products, having at its disposal both a dedicated community and effective help forums, as well as an active development.

Gazebo is reported to have numerical instability when simulating low inertia articulated bodies, as is the case of a robotic hand due to the small dimensions and weight of the finger links [88]. This may be a problem under the proposed application in this project, however that will be verified later.

3.4.2 MuJoCo

MuJoCo (Multi-Joint dynamics with Contact) [89] is a physics engine that makes it possible to develop, simulate, and visualise contact dynamics of multi-joint systems rapidly and accurately.

As the main focus of this simulator is performance, it is able to navigate very complicated operations from the computational standpoint particularly in highly complex systems with several contact points to be processed, as is the case of a multifingered robotic hand. Other arguments in favor of MuJoCo for use cases that are similar to the one presented here are it's capability to simulate soft, rope like objects or even complex particle systems [90], supports soft contacts natively and is built with multiple contacts in mind hence inverse dynamics are well defined even in this situation [89].

The major downside of MuJoCo for manipulation tasks simulation is the lack of an inverse kinematics solver and path planning, unlike Gazebo which, with its integration with ROS, acquires these features [90].

MuJoCo models are written in the MJCF format which is an XML derivation, however Unified Robot Description Format (URDF) typically used in Gazebo and ROS can also be loaded natively or converted to MJCF if needed.

Chapter 4

Prototype Description

4.1 Gazebo Simulation

The important concepts regarding the Realistic Simulation on Gazebo will be presented in the current Section.

4.1.1 URDF Model

For kinematic computation and simulation purposes, a model that accurately represents the physical characteristics of the robot and its links and joints is of the upmost importance. Therefore, the first step of this work was the development of such model.

The ROS driver package for the Schunk multifingered hand [1] provides a bare-bones model using Unified Robot Description Format (URDF) and Xacro which is an XML macro language. In the original state it is not very useful as there are a lot of components missing, for instance inertial properties and surface characteristics such as friction, being the latter essential when the objective is gripping an object. The finger links and most sections of the hand model can be well approximated by cylinders, hence the following macro could be used to introduce inertial parameters in the associated URDF file.

```
<xacro:macro name="cylinder_inertial_fingers" params="radius length mass *origin">
  <inertial>
  <mass value="${mass}" />
  <xacro:insert_block name="origin" />
  <inertia ixx="${0.5 * mass * radius * radius}" ixy="0.0" ixz="0.0"
           iyy="${0.0833333 * mass * (3 * radius * radius + length * length)}"
           iyz="0.0"
           izz="${0.0833333 * mass * (3 * radius * radius + length * length)}"/>
  </inertial>
 </xacro:macro>
```

[1] ROS Driver package for the SVH hand https://github.com/fzi-forschungszentrum-informatik/schunk_svh_driver, as of July 15, 2022

Unfortunately the hand was not physically available for the measurement of the necessary parameters and the provided documentation is not detailed in this matter, not revealing the individual mass of each link neither their corresponding radius and length. It is predicted that in the future with access to the hardware these parameters can be extracted nevertheless, while that is not the case, the following rough approximation is considered for the links of the hand model:

```
<xacro:macro name="link_inertia">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <mass value="0.01" />
    <inertia
      ixx="0.1" ixy="0.0" ixz="0.0"
      iyy="0.1" iyz="0.0"
      izz="0.1" />
    </inertial>
</xacro:macro>
```

To make the model ready for Gazebo simulation some other information is required. Regarding contact properties we are left in the same situation as described above, not having these details available. The static and kinematic friction coefficients are set to an ideal unitary value as it can be observed below in the "mu" and "mu2" parameters, respectively. Note that this macro is added as an attribute to the "collision" elements of the links that require friction, namely the links of the hand that represent fingers.

```
<xacro:macro name="surface_friction">
  <surface>
    <friction>
      <ode>
        <mu>1</mu>
        <mu2>1</mu2>
      </ode>
    </friction>
  </surface>
</xacro:macro>
```

In order to use the ros_control package [91] it is also necessary to configure the transmissions which will effectively propagate position, velocity and effort variables between actuator and joint spaces. A simple transmission with a gearing ratio of 1:1 between actuator and joint sides is introduced for each one of the 9 DoF that the hand contains, being the example below for the thumb flexion channel.

```
<transmission name="thumb_flexion_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="${name}_Thumb_Flexion">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="thumb_flexion_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
```

```
</transmission>
```

As previously seen in Chapter 2.4, the hand is under actuated having only 9 DoF that control the 20 joints that compose it. This mechanical coupling between joints must be introduced in the model in order for the simulator to accurately represent its movement. Gazebo itself does not provide support for such coupling of joints, yet this functionality can be added by means of a community-made Mimic Joint plugin [2]. This plugin couples a mimic joint to a parent joint and maintains a defined linear relationship between their angles set by a multiplier and an offset. It also has the capability of loading a PID controller in the mimic joint that would follow the angle of the parent joint as a reference, however this is not the case here as the joints are mechanically coupled in reality. A macro was created in order to simplify the usage of this plugin:

```
<xacro:macro name="mimic_joint_plugin_gazebo" params="name_prefix parent_joint
    mimic_joint has_pid:=false multiplier:=1.0 offset:=0 sensitiveness:=0.0
    max_effort:=1.0 robot_namespace:=''">
  <gazebo>
    <plugin name="${name_prefix}_mimic_joint_plugin" filename="
    libroboticsgroup_upatras_gazebo_mimic_joint_plugin.so">
      <joint>${parent_joint}</joint>
      <mimicJoint>${mimic_joint}</mimicJoint>
      <xacro:if value="${has_pid}">
        <hasPID />
      </xacro:if>
      <multiplier>${multiplier}</multiplier>
      <offset>${offset}</offset>
      <sensitiveness>${sensitiveness}</sensitiveness>
      <maxEffort>${max_effort}</maxEffort>
      <xacro:unless value="${robot_namespace == ''}">
        <robotNamespace>${robot_namespace}</robotNamespace>
      </xacro:unless>
    </plugin>
  </gazebo>
</xacro:macro>
```

An example is the coupling of the index finger spread and the pinky finger spread that can be implemented by the instantiating of the macro in the following manner:

```
<xacro:mimic_joint_plugin_gazebo name_prefix="${name}_index_spread"
    parent_joint="${name}_Finger_Spread" mimic_joint="${name}_index_spread"
    has_pid="false" multiplier="0.5" max_effort="10.0"
/>
```

Although not being strictly necessary for the correct representation of system dynamics, a force feedback ground truth sensor was added to the tip of the index finger in order to extract data that is representative of the behavior of the body under contact. This sensor implementation is part of the Gazebo library of sensors in which it is referenced as F3D. A macro was defined to streamline the addition of the plugin to the URDF model:

[2] Gazebo plugins by the Robotics Group at the University Of Patras https://github.com/roboticsgroup/roboticsgroup_upatras_gazebo_plugins

```
<xacro:macro name="f3d_plugin_gazebo" params="robot_namespace:='' body_name">
    <gazebo>
      <plugin name="gazebo_ros_f3d" filename="libgazebo_ros_f3d.so">
        <xacro:unless value="${robot_namespace == ''}">
          <robotNamespace>${robot_namespace}</robotNamespace>
        </xacro:unless>
        <bodyName>${body_name}</bodyName>
        <topicName>${body_name}_f3d</topicName>
        <frameName>${body_name}</frameName>
      </plugin>
    </gazebo>
  </xacro:macro>
```

In a similar fashion, some other objects can be defined to complete the world and build the simulation environment that is presented in Figure 4.1.

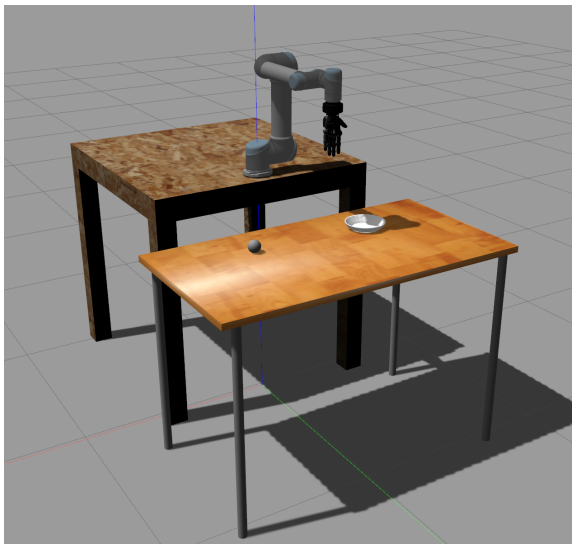


Figure 4.1: Constructed Gazebo environment

4.1.2 MoveIt Configuration Package

Using a robot through MoveIt happens via a configuration package that needs to be created. For this, the framework includes a tool called *MoveIt Setup Assistant* that, through a comprehensive graphical user interface (GUI), allows for the creation of such configuration package.

The process starts by importing the URDF that includes the description of the robot. After this, planning groups, which are used to describe different parts of the robot, can be created. In this particular case, as we are using the BioIK kinematic solver that supports non-trivial kinematic chains, the entire robot including the manipulator and the dexterous hand can be added to the same group. This step is illustrated in Figure 4.2.

Fixed poses can be added to the configuration and two were created as it can be observed in Figure 4.3, namely "home" and "test". The first one places the robot in a neutral position while the

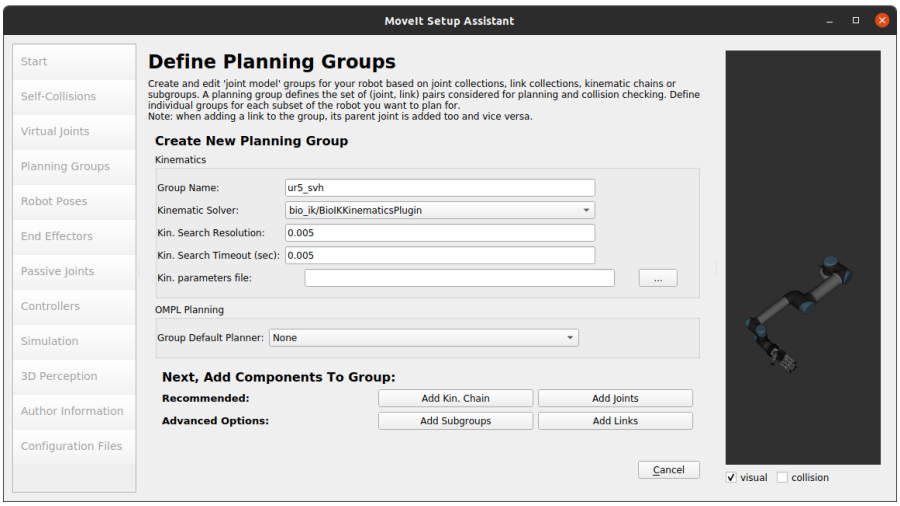


Figure 4.2: Planning group creation in MoveIt setup assistant

second is used to touch the indicator finger with a surface in order to observe contact performance of the simulator. Joint angles of both poses are detailed in Table 4.1.

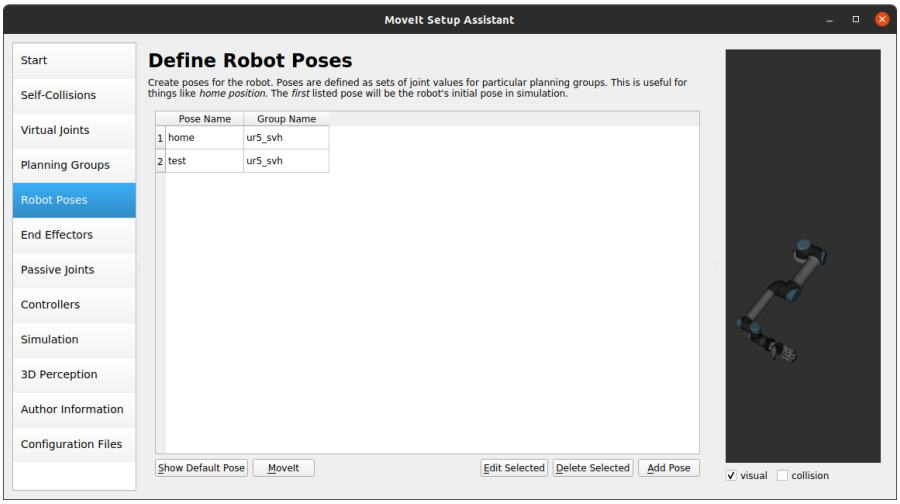


Figure 4.3: Fixed robot pose definition

The next key step is to add passive joints. These joints are not directly actuated on the robot therefore the planner must be informed not to kinematically plan for these joints as they can not be controlled. All the passively actuated hand joints are specified in this step as it is illustrated by Figure 4.4.

Table 4.1: Defined robot poses details

Joint	"home"	"test"
Finger_Spread	0	0
Index_Finger_Distal	0	0
Index_Finger_Proximal	0	0
Middle_Finger_Distal	0	0.6981
Middle_Finger_Proximal	0	0.4363
Pinky	0	0.4363
Ring_Finger	0	0.4363
Thumb_Flexion	0	0
Thumb_Opposition	0	0
shoulder_lift_joint	-1.5447	0.4014
shoulder_pan_joint	1.5707	1.5707
elbow_joint	1.5447	0.9250
wrist_1_joint	-1.5794	2.3387
wrist_2_joint	-1.5794	1.5794
wrist_3_joint	0	0

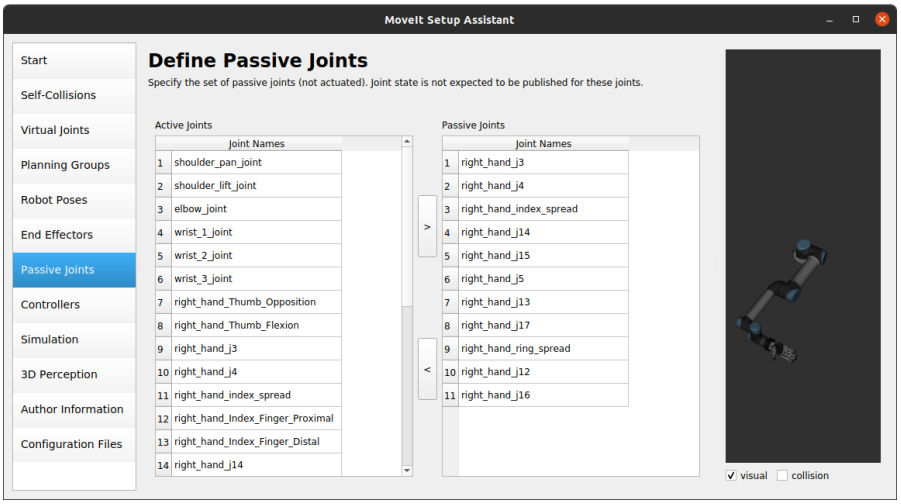


Figure 4.4: Passive joints definition

Finally, controllers are added to the controllable joints of the robot in order to be able to interface with the interfaces defined in the URDF model, making them follow the position trajectory that is computed by the planner. The Setup Assistant is capable of doing this automatically using the "Auto Add" function shown in Figure 4.5.

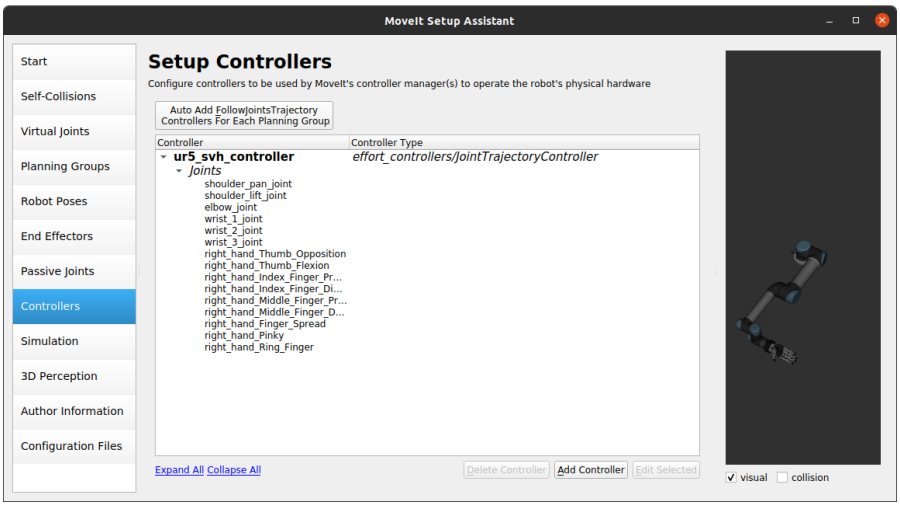


Figure 4.5: Adding simulated controllers

The information in the configuration package is used to launch a move_group node that has the architecture presented in Figure 4.6. The move_group_interface is a C++ interface that provides easy access to the most important functionality of the node, such as planning trajectories and executing them, so it will be the main tool to interact with the robot from a high level.

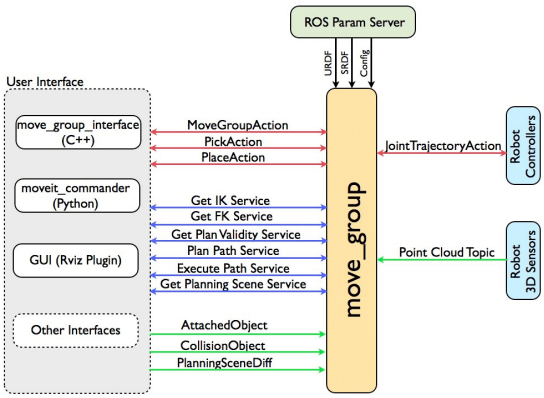


Figure 4.6: High level system architecture diagram of a MoveIt move_group node [77]

4.2 MuJoCo Simulation

The important concepts regarding the Realistic Simulation on MuJoCo will be presented in the current Section.

4.2.1 MJCF Model

The previously presented URDF model can be converted to the MJCF format. The first step is to convert all the mesh files that are provided in .dae to .stl as this is the only format supported by MuJoCo.

Next, some MuJoCo tags need to be added for the model compiler to interpret:

```
<mujoco>
  <compiler
  meshdir="/home/francisco/catkin_ws/src/mujoco_ur5_svh_coupled/meshes"
  balanceinertia="true"
  discardvisual="false" />
</mujoco>
```

These will point to the directory that contains the meshes, indicate if we want the inertia matrices to be automatically balanced and if we want to discard the visual aspect of the robot and replace complex mesh files with simple convex hull geometry files, respectively. More compiler definitions can be found in the MuJoCo documentation[3].

Finally, the Xacro file with the macros can be converted to plain URDF and this resulting file can be compiled to MJCF by using the *compile.cc* tool that comes packaged with MuJoCo.

The pre-built *simulate.cc* tool packaged with MuJoCo can be used to test the generated model after compilation and will show a window containing what is presented in Figure 4.7.

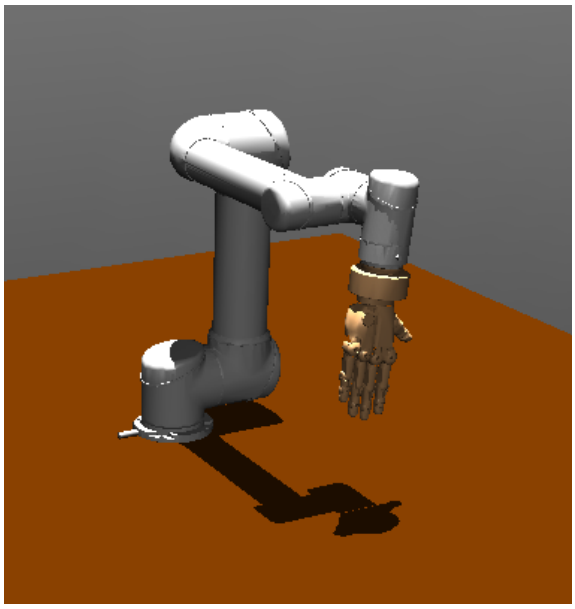


Figure 4.7: Constructed MuJoCo environment

Some things need to be added manually to the model to make it functional, namely the actuators and the equalities that represent the joint coupling.

[3]MuJoCo XML compiler reference https://mujoco.readthedocs.io/en/latest/XMLreference.html#compiler

Actuators are mainly described by their type, name and the joint that they are coupled to. Some other details can be added to add higher degree of fidelity as we can see from the control range and gear applied below to the "shoulder_pan_joint" actuator.

```
<actuator>
  <motor ctrllimited="true"  ctrlrange="-100 100"
  name="shoulder_pan_joint"  joint="shoulder_pan_joint"
  gear="1"/>
</actuator>
```

In order to physically introduce the mechanical interdependence of the SVH hand joints, equality parameters are needed in the model.The equalities have a certain type, in this case they are joint equalities and need some configurations:

```
<equality>
  <joint joint2="right_hand_Thumb_Opposition"
  joint1="right_hand_j5"  polycoef="0 1 0 0 0" active="true"
  name="equality_constraint" />
 </equality>
```

In the presented example above for the coupling between the thumb opposition and the j5 joint we can see that the "polycoef" parameter has the value "0 1 0 0 0", which means that the coefficients a_0 to a_4 for the quartic polynomial that describe the constraint have such values. Therefore the following equation applies:

$$joint_1 = a_0 + a_1\, joint_2 + a_2\, joint_2^2 + a_3\, joint_2^3 + a_4\, joint_2^4 \tag{4.1}$$

4.2.2 MoveIt Configuration Package

Creating the MoveIt configuration package to interact with the simulator in MuJoco is very similar to the process followed previously for the Gazebo case. The only difference is that there is no need to setup controllers at this stage as they will be defined in the created virtual hardware interface described in 4.2.4.

4.2.3 Simulation Environment

Comparing to Gazebo, MuJoCo does not have native integration with the ROS framework. Therefore, in order to take advantage of the motion planning tools that ROS offers, a MuJoCo simulation environment with the capability of subscribing and publishing to ROS topics had to be developed.

The connector was developed C++ and has the main objective of sending to MuJoCo joint reference positions associated with trajectories generated by MoveIt and ROS. MuJoCo on the other hand sends data to ROS containing current state of the robot, which is very important for motion planning, and also position data regarding any object in the scene that the user wants to keep track of.

The subscribing of joint reference positions runs asynchronously while the simulation is executed according to the flowchart presented in Figure 4.8.

This environment can be easily configured to be used with another model as it is configured though a single *.yaml* file. This file is read according to the schema:

```
#Schema
#joints: <-- creates a JointROSConnector instance for a joint. Corresponds to a
    joint in the MuJoCo xml
#- name:
#  control_topic:
#  pose_topic:
#  initpose_topic:
#  p_gain:
#  d_gain:
```

This provides the information to create a new object from the joint_ros_connector class including the name linked to the control topic, pose topic and initial pose topic as well as the proportional and differential gains for the MuJoCo controller responsible for the joint at hand. The controller function will be executed periodically at each simulation step adjusting the effort of each of the actuators in order to follow the joint position reference received. This function is implemented by a PD controller that calculates expression 4.2.

$$ctrl = -p\_gain(joint\_position - joint\_reference) - d\_gain * joint\_velocity \tag{4.2}$$

In Figure 4.9 the class diagram of this connector is presented.

4.2.4 MuJoCo Virtual Hardware Interface

The simulation environment described above, although capable of communicating though ROS messages with other nodes, is limited to receiving joint reference position values for each of the registered articulations of the robot in the model. The issue is that the MoveIt motion planner produces full trajectories which are sequences of joint reference positions with a time instant associated to reaching them. The data frame in which this data in embedded and sent is quite complicated so a solution was found to avoid having to implement a parser to do this task and at the same time keeping the architecture as modular as possible, hence future proofing the system.

Taking the example of the inner working of ROS when communicating with a simulation in Gazebo, MoveIt sends the trajectory to the node running the ROS controllers. The controllers send position or effort goals, depending on their type, to a lower layer. This layer is composed by hardware interfaces that are compatible with the controllers and create a level of abstraction to the actuators associated to it. In a real implementation of a simple robotic system this hardware interface can receive for example an effort reference from the controller and send a motor current reference to the robot via a serial interface.

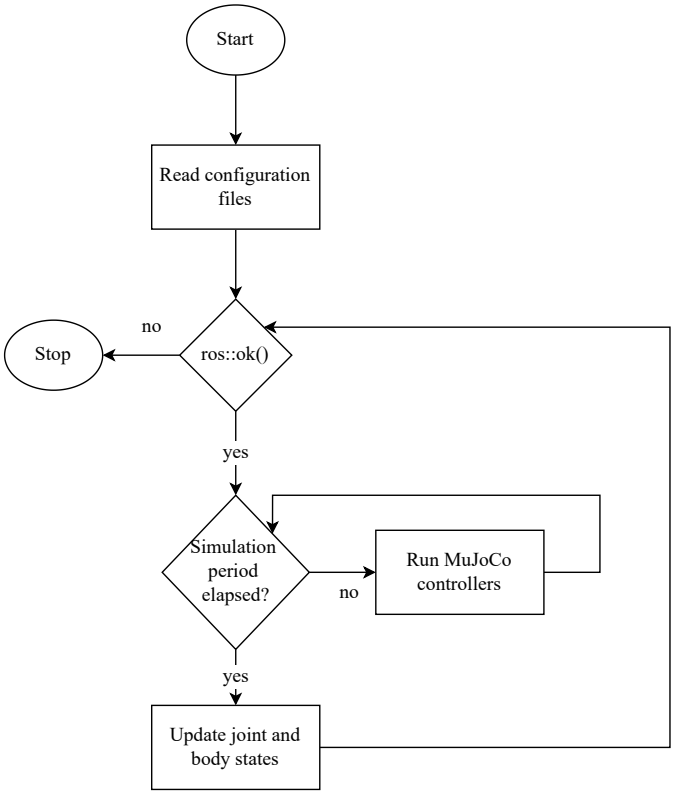


Figure 4.8: Flowchart of the developed simulation environment

This architecture can be slightly modified and replicated in order to produce a Controller interface that receives parsed joint position goals from the controllers and sends those same joint position goals to the MuJoCo simulation environment as it is represented in Figure 4.10.

The *ros_control* package includes many different kinds of controllers. In this case the needed function is only trajectory parsing and sending the position reference directly to the controller interface so Joint Trajectory Position Controllers are used. These are configured in a "controllers.yaml" file in the Controller Interface package:

```
MyControllerInterface:

  # Publish all joint states
  joints_update:
      type: joint_state_controller/JointStateController
      publish_rate: 50


  JointC_PositionController:
      type: "position_controllers/JointTrajectoryController"
      # Since JointC uses position interface this controller type is  used
      joints:
        - shoulder_pan_joint
...
```

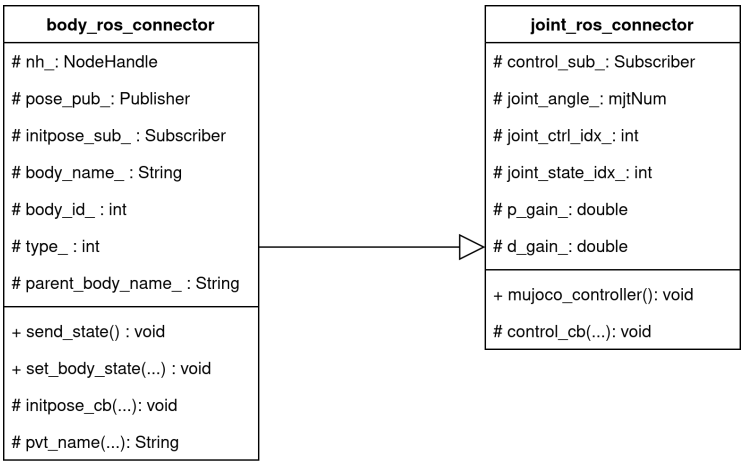


Figure 4.9: Class diagram of the simulation environment

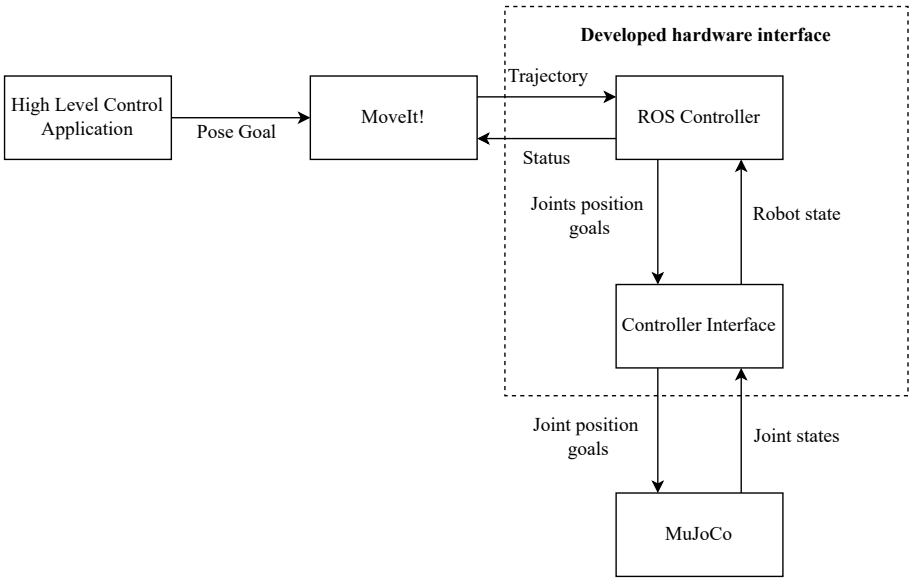


Figure 4.10: Architecture of the system using the developed hardware interface

Chapter 5

Results and Discussion

The described prototypes were used for the collection of data with the objective of comparing the performance of the studied simulator platforms. The observations are presented in this chapter and some inferences are drawn from them. Demos of the robot in Gazebo and MuJoCo going from the "home" to "test" pose can be found in the following links: https://youtu.be/-_W3hOeciS0 and https://youtu.be/EMqRMzmRrfQ.

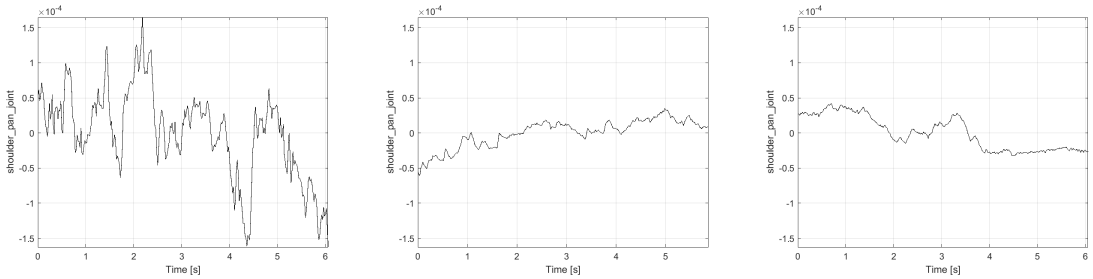
5.1 Gazebo Contact Behavior

Firstly, data regarding joint states of the robot, when the index finger touches the surface in Gazebo simulation was captured and is presented in Figures 5.1 to 5.15. Also, using the force feedback sensor in the index finger, values of the vertical component of contact force are stored and displayed in Figure 5.16. In both cases, data was recorded for three different settings of inertia of the finger links consisting of diagonal matrices of $0.001 kg\,m^2$, $0.01 kg\,m^2$ and $0.1 kg\,m^2$.

The collected joint position data was analysed by computing the mean absolute deviation (MAD) of the data with respect to the running average of itself, calculated with a window size of 10 samples. This statistical indicator, which can be calculated by the expression in 5.1, provides a sense of the degree of oscillation and instability of the reading. Results of this indicator compose the Table 5.1.

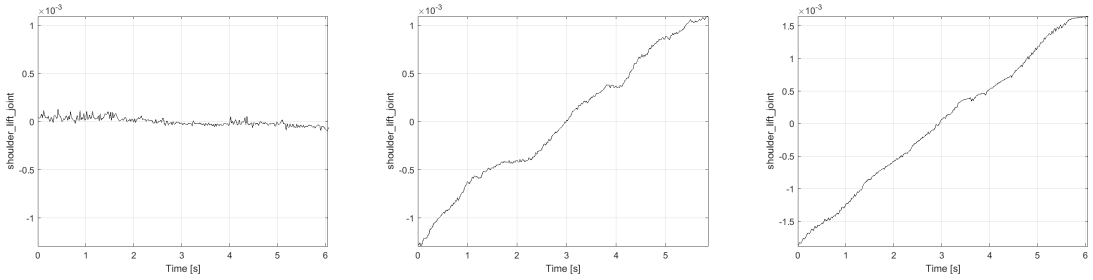
$$MAD = \frac{\sum_i |x_i - \bar{x}_i|}{N} \tag{5.1}$$

where N is the total number of samples, x_i is the i^{th} sample and $\bar{x}_i$ is the running average around the i^{th} sample.



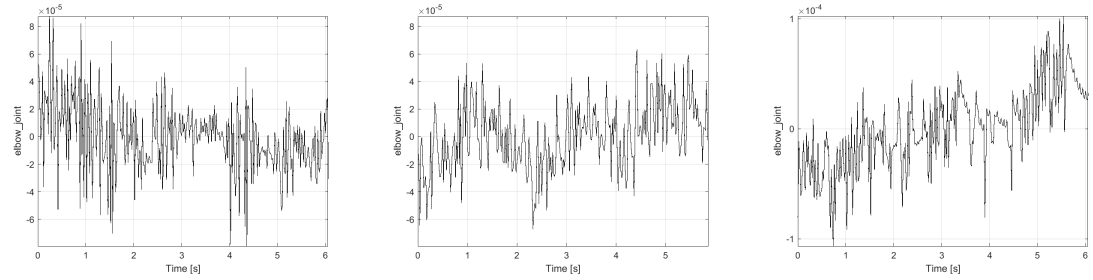
(a) (b) (c)

Figure 5.1: Graphical representation of shoulder_pan_joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



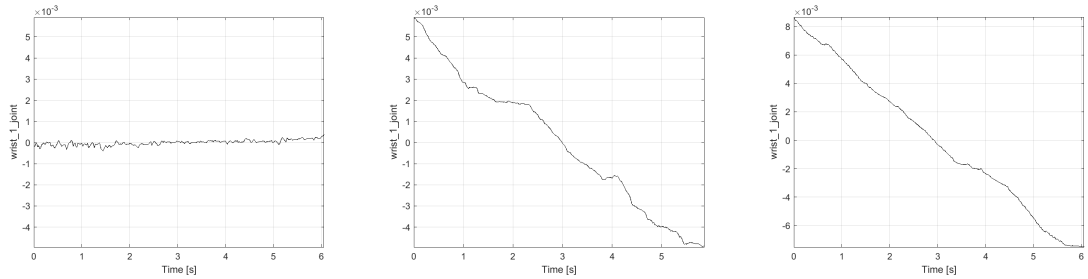
(a) (b) (c)

Figure 5.2: Graphical representation of shoulder_lift_joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



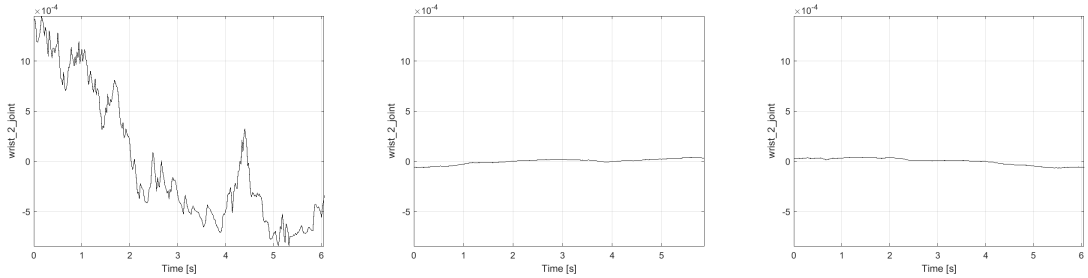
(a) (b) (c)

Figure 5.3: Graphical representation of elbow_joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



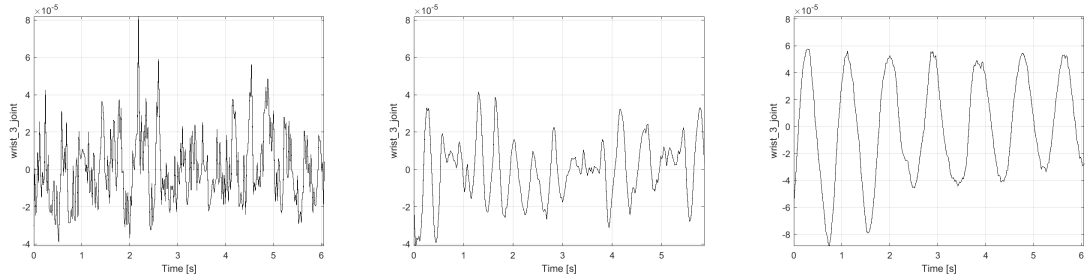
(a) (b) (c)

Figure 5.4: Graphical representation of wrist_1_joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



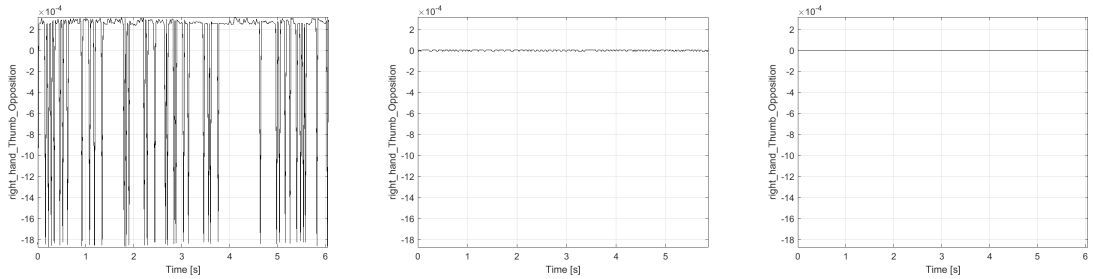
(a) (b) (c)

Figure 5.5: Graphical representation of wrist_2_joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



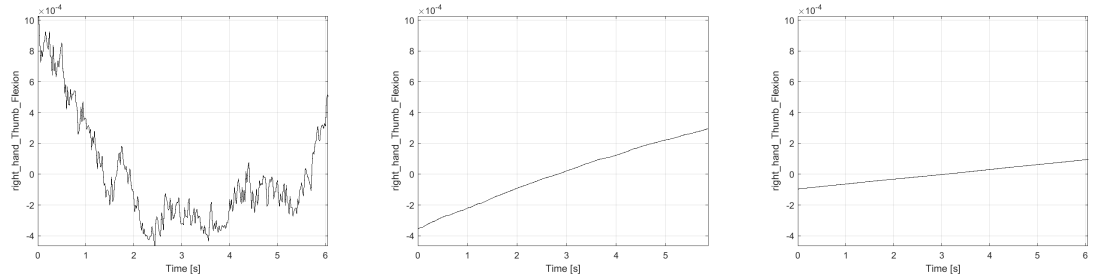
(a) (b) (c)

Figure 5.6: Graphical representation of wrist_3_joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



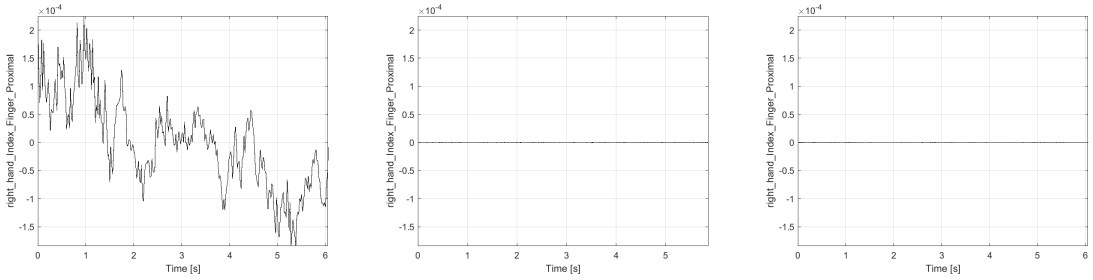
(a) (b) (c)

Figure 5.7: Graphical representation of right_hand_Thumb_Opposition joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



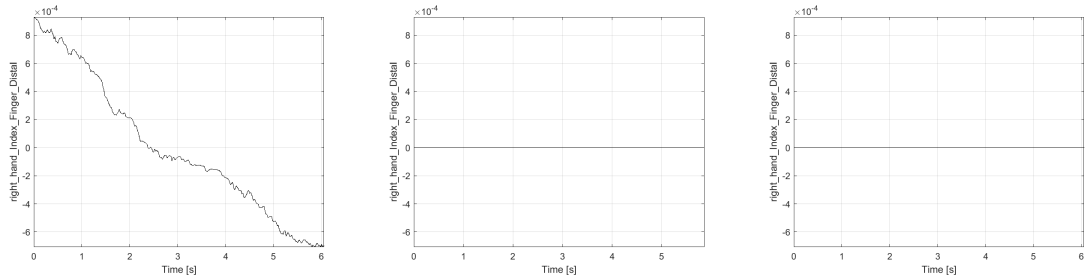
(a) (b) (c)

Figure 5.8: Graphical representation of right_hand_Thumb_Flexion joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



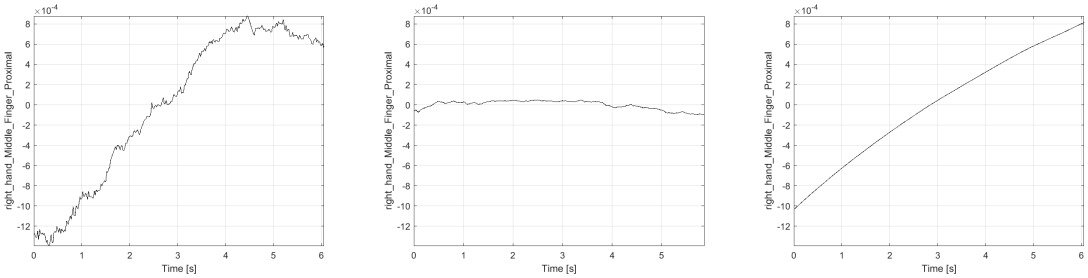
(a) (b) (c)

Figure 5.9: Graphical representation of right_hand_Index_Finger_Proximal joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



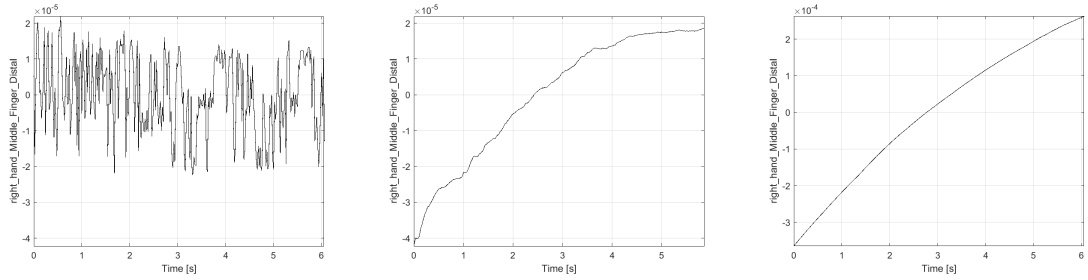
(a) (b) (c)

Figure 5.10: Graphical representation of right_hand_Index_Finger_Distal joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



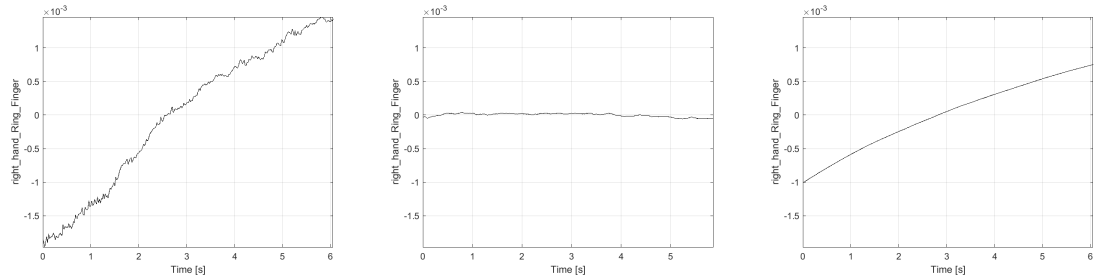
(a) (b) (c)

Figure 5.11: Graphical representation of right_hand_Middle_Finger_Proximal joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



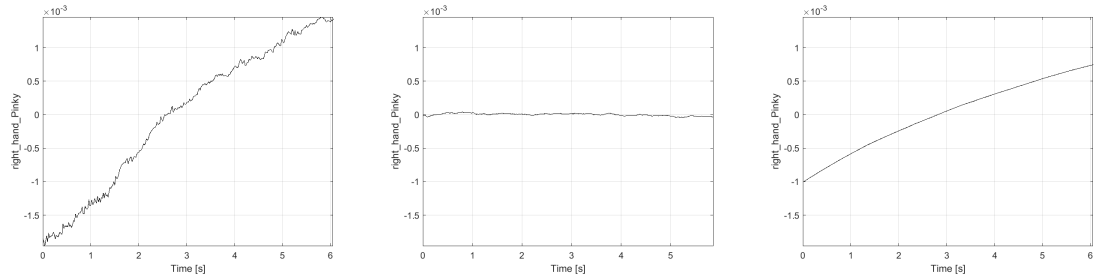
(a) (b) (c)

Figure 5.12: Graphical representation of right_hand_Middle_Finger_Distal joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



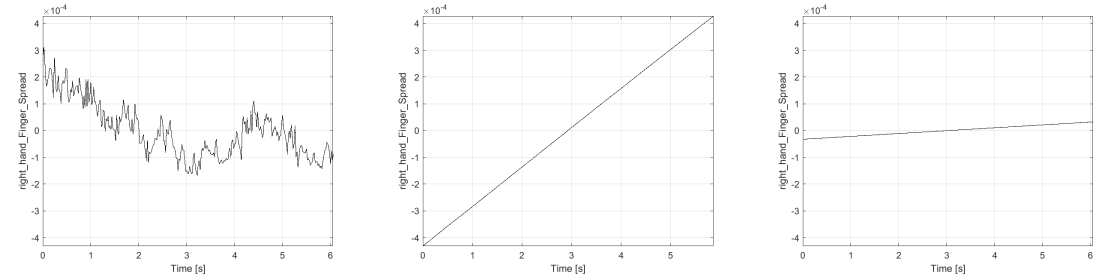
(a) (b) (c)

Figure 5.13: Graphical representation of right_hand_Ring_Finger joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



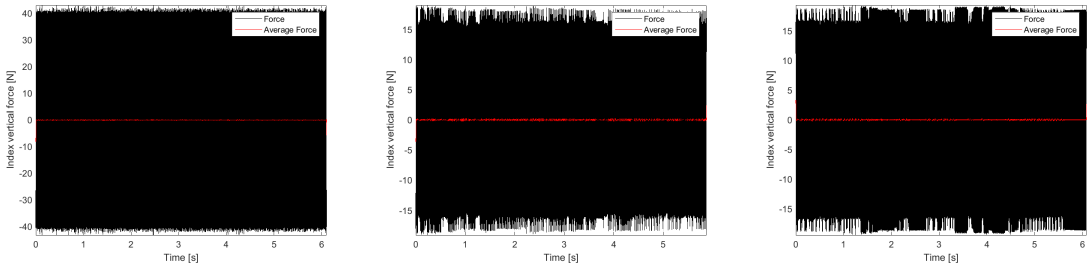
(a) (b) (c)

Figure 5.14: Graphical representation of right_hand_Pinky joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



(a) (b) (c)

Figure 5.15: Graphical representation of right_hand_Finger_Spread joint position for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$



(a) (b) (c)

Figure 5.16: Graphical representation of index tip contact force (black) and moving mean of the contact force (red) for inertias of a) $0.001 kg\,m^2$, b) $0.01 kg\,m^2$ and c) $0.1 kg\,m^2$

Interpretation of the results up to this point confirm that Gazebo has in fact numerical stability issues when simulating articulated bodies with small inertias under contact. Figures 5.1 to 5.15 show a very unstable behavior of the joint states and this is highlighted by the analysis in Table 5.1. With the increase of inertia in the simulated articulated bodies the average MAD shows a reduction in its value which reflects the associated decrease in instability also visually evident in the graphical representations of Figures 5.1 to 5.15.

The effect that the apparent vibration in the joints has over the force applied over an object demonstrated in Figure 5.16 it to be expected as the contact is not constant but intermittent. High oscillation in the force is therefore observed leading to a moving mean of the contact force that is approximately zero. Application of normal forces to the surface of an object is absolutely essential for gripping and manipulation tasks as these create the friction required to move the object, hence Gazebo seems to not be a very adequate solution for the realistic simulation problem in the scope of this project.

Table 5.1: Gazebo joint position dispersion assessment

Joint	inertia = $0.001 kg\,m^2$	inertia = $0.01 kg\,m^2$	inertia = $0.1 kg\,m^2$
"Finger_Spread"	0.0240e-3	0.01541e-4	0.00111e-4
"Index_Finger_Distal"	0.0097e-3	0.00091e-4	0.00141e-4
"Index_Finger_Proximal"	0.0204e-3	0.00151e-4	0.00161e-4
"Middle_Finger_Distal"	0.0072e-3	0.00171e-4	0.01111e-4
"Middle_Finger_Proximal"	0.0196e-3	0.03041e-4	0.03261e-4
"Pinky"	0.0222e-3	0.02871e-4	0.03111e-4
"Ring_Finger"	0.0221e-3	0.03011e-4	0.03121e-4
"Thumb_Flexion"	0.0471e-3	0.01221e-4	0.00331e-4
"Thumb_Opposition"	0.4379e-3	0.05771e-4	0.00531e-4
"shoulder_lift_joint"	0.0151e-3	0.10011e-4	0.12251e-4
"shoulder_pan_joint"	0.0149e-3	0.02301e-4	0.01531e-4
"elbow_joint"	0.0173e-3	0.15331e-4	0.17551e-4
"wrist_1_joint"	0.0474e-3	0.34761e-4	0.38181e-4
"wrist_2_joint"	0.0603e-3	0.00921e-4	0.01131e-4
"wrist_3_joint"	0.0118e-3	0.06101e-4	0.04361e-4
Average	5.1799e-05	5.8178e-06	5.7914e-06

5.2 MuJoCo Contact Behavior

Similarly, for analysis of the MuJoCo realistic simulation environment, joint states of the robot when the index is in contact with a surface and the force that is exerted over said surface were captured. Time series of joint states are displayed in Figure 5.18 while force readings are in Figure 5.17.

The same metodology used to process the states in the Gazebo case was also applied here producing the scrutiny presented in Table 5.2.

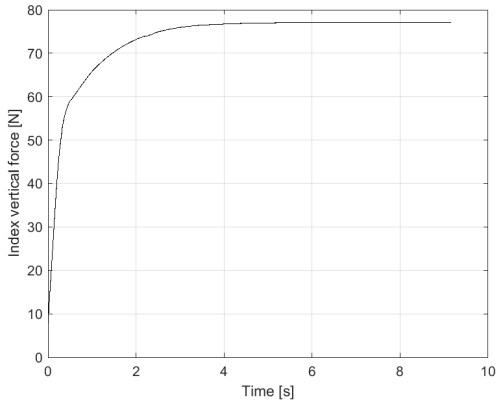
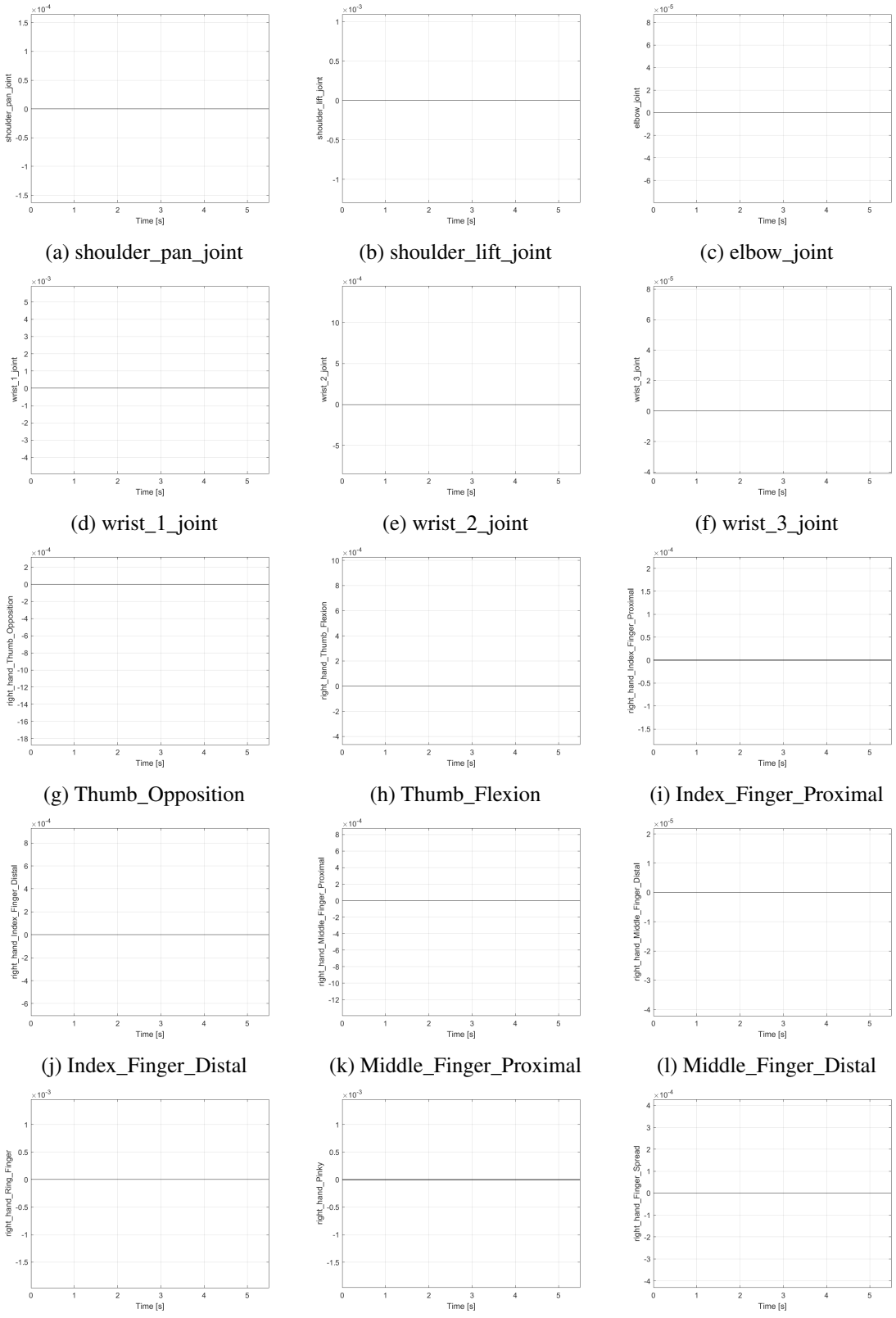


Figure 5.17: Graphical representation of index tip contact force in MuJoCo

MuJoCo presents a more stable operation that can be verified not only visually from Figures 5.17 and 5.18 but also analytically from the results obtained in Table 5.2. The joint states mostly stabilize in a final value when under contact which translates into an average MDA that is more that 10 orders of magnitude lower when compared to the equivalent test in Gazebo. The contact force exerted by the tip of the index finger shows a response much more similar to what was expected from a realistic simulation of the contact. Consequently, MuJoCo can be considered a valid candidate to perform simulation of the robotic system that is the protagonist to this project, when applied to grasping and manipulation tasks.



(a) shoulder_pan_joint (b) shoulder_lift_joint (c) elbow_joint

(d) wrist_1_joint (e) wrist_2_joint (f) wrist_3_joint

(g) Thumb_Opposition (h) Thumb_Flexion (i) Index_Finger_Proximal

(j) Index_Finger_Distal (k) Middle_Finger_Proximal (l) Middle_Finger_Distal

(m) Ring_Finger (n) Pinky (o) Finger_Spread

Figure 5.18: Graphical representation of joint positions in MuJoCo

Table 5.2: MuJoCo joint position dispersion assessment

Joint	inertia=0.1
"Finger_Spread"	0.0000
"Index_Finger_Distal"	0.0000
"Index_Finger_Proximal"	0.0000
"Middle_Finger_Distal"	0.0354e-15
"Middle_Finger_Proximal"	0.0555e-15
"Pinky"	0.0330e-15
"Ring_Finger"	0.0000
"Thumb_Flexion"	0.0000
"Thumb_Opposition"	0.0000
"shoulder_lift_joint"	0.0422e-15
"shoulder_pan_joint"	0.0845e-15
"elbow_joint"	0.1082e-15
"wrist_1_joint"	0.1400e-15
"wrist_2_joint"	0.1561e-15
"wrist_3_joint"	0.0001e-15
Average	4.3664e-17

Chapter 6

Conclusion and Future Work

There are multiple robotic simulation tools, each one with its advantages, which makes them well suited for different applications. In this work, two were explored regarding their adequacy for realistic simulation of a dexterous robotic hand and manipulator system for highly detailed tasks. The aforementioned exploration involved the development of description models compatible with Gazebo and MuJoCo, as well as support applications for the latter.

The results allow to conclude that Gazebo, in spite of its higher user-friendliness and integration with the ROS framework, is not well adapted to use cases such as the one presented in this work. MuJoCo shows promising results in the tested setting and, through the developed virtual hardware interface, the same benefits of ROS integration can now also be taken advantage of when using this alternative simulation platform.

Aside from the presented and analysed results in this document, this dissertation project resulted in the publication of a conference article and an open-source simulation suite for the described robotic system. The published article focused on "Reinforcement learning techniques applied to the motion planning of a robotic manipulator" and is included in Appendix A for further analysis. The models produced to execute the simulations as well as the code corresponding to the developed control pipeline can be found at https://github.com/franciscombr/SVH-simulation-suite.

Some external factors, such as the delay in the arrival of the SVH hand, led to the adaptation of the work's objectives. Moreover, there were also some requirements that were simplified in the planning phase, and were implemented after, such as the need for a stable simulation platform, which shifted the focus of the efforts to completing these requirements.

Future work on this subject can include the realistic physical parameters' modelling of the SVH hand, once the hardware becomes available. This will allow to reach simulation with a higher degree of fidelity to reality. Another step can be exploring the capabilities of the MoveIt motion planner as well as the algorithms mentioned in the literature review to complete pick and place tasks. The transition to this process will now be faster due to the developed simulation pipeline in the course of this dissertation project.

Regarding experimenting with the real hardware, the pipeline can be adapted to write and read

values to topics published and subscribed to not by the simulation environment but by a real robot and future work will include this adaptation.

Appendix A

Published Articles

2022 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
Santa Maria da Feira, Portugal – April 29-30, 2022

2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) | 978-1-6654-8217-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICARSC55462.2022.9784814

Reinforcement learning techniques applied to the motion planning of a robotic manipulator

Francisco M. Ribeiro*†, Vítor H. Pinto*†
* FEUP - Faculty of Engineering of University of Porto, Portugal,
Email: up201809215@edu.fe.up.pt, vitorpinto@fe.up.pt
† SYSTEC (DIGI2)
Research Center for Systems and Technologies
(Digital and Intelligent Industry Lab), Porto, Portugal

***Abstract*—Throughout this article the execution of the motion planning for a robotic manipulator by means of Reinforcement Learning methods is studied. Towards this, an implementation based on a "Wire and loop" game is used as an example case to be solved. The loop is controlled in a single plane as the end-effector of the manipulator. The modeling of the problem and the process of training the agent is detailed. This allowed for the verification of the capacity of a learning based method, having produced, under the considered abstractions, satisfying results by gaining the capability of completing the path imposed by the wire in 23 seconds.**

***Index Terms*—Robotic Manipulator, Reinforcement Learning, Industry 4.0, Motion Planning, Simulation**

I. Introduction

Across the Industry 4.0 standards, considered to be the industrial revolution of modern times, Cyber-Physical Production Systems (CPPS) are seen as the great drivers for the increase in productivity that is needed to correspond to today's economical demands. From CPPS some specific features are expected for them to become useful, such as their robustness facing different situations, security, autonomy and efficiency [1].

From the necessity of increasing industrial productivity levels, arises the utilization of robotic manipulators which have been becoming more and more popular in different applications [2]. With the associated technological developments, these have been gaining increasingly advanced capabilities that allow them to reach human level of performance in some assembly line tasks such as welding, painting, assembling and machining, among others [3]. Many of these tasks can be seen as the execution of a continuous motion over a defined trajectory which leads to less complicated solutions as is the case of programming the trajectory in a rigid way. However, this form of planning that does not allow for any amount of flexibility does not converge with the the expectations established for current CPPS, namely the adaptability to different conditions and autonomy [4]. Therefore, the need for endowing robotic manipulators with algorithms capable of dealing with environment variations is highlighted. They should have the capability of adapting their planning according to these variations.

Deep Reinforcement Learning (DRL) techniques have been evaluated as a possible robotic control solution capable of solving complicated manipulation tasks, as in the case of pick and place functionality [5]. Although results are promising, having these methods shown results that validate their use on industrial manipulators, because they directly process data for the automatic detection of important feature, they become computationally very expensive. The time required for training of DRL agents is reported to be in the order of days even when multiple cores of processing are used [5]. Therefore, there is space for improvement when it comes to the application of learning-based methods for the motion planning of robotic manipulators.

Having this, the main contribution of this paper is the modeling of the learning problem at hand and the application of a Reinforcement Learning (RL) algorithm as the solution. The utilization of a discrete action controller is proposed using the typical Markov Decision Process (MDP) scheme associated to the wire and loop game. In this game, the objective is to guide the loop around the wire from the beginning to the end without ever making contact between both. The presented solutions are implemented in MATLAB.

Initially a brief overview of the main theoretical aspects that are the core of this work is conducted in Section II. The way that these elements are applied is described in Section III. Finally results and some comments regarding the present work are presented in Sections IV and V, respectively.

II. Background

A. Markov Decision Processes

MDP are a class of sequential stochastic decision processes whose reward and state transition functions depend only on the current state and action. A MDP consists of

$M(S, A, R, \gamma, H, P)$ [6], and a graphic representation of a simple example of such process can be observed in Figure 1. $S_t \in S$ represents the state of the agent at each time step, A_t is

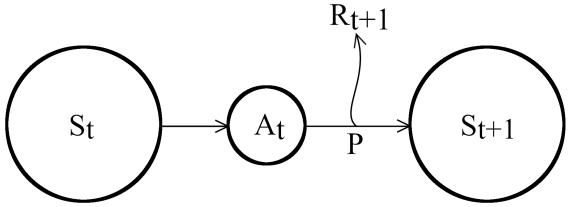


Fig. 1: Schematic representation of a MDP

the action that the agent takes from a state S_t and R is related to the reward associated with that action. The value γ is a constant between 0 and 1 and defines the relative importance of recent rewards compared to the delayed ones. H indicates the maximum number of steps that can be taken inside a single episode and T is a function of the current state and action that gives the probability of transitioning to a certain following state.

In decision processes such as the previously described example, there is a decision maker that, at each instant that an option must be made, observes the state of the system and focusing on this information it picks an action among a set of possibilities. From the made decision the state transitions to a new state with an associated probability and also an associated reward. These models have been applied to a wide range of research topics from queuing models to inventory control [7] and to represent problems in a suitable way for RL application [4].

The end goal of the decision maker is to select the actions that will optimize the operation of the controlled system, i.e. to maximize the received rewards. This concept of optimization is introduced by the Bellman equation [8]:

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma\, max_{a'}\{Q(s',a')\} - Q(s,a)) \quad (1)$$

where Q expresses the quality of an action a taken from state s. Reward r is observed and the transition to state s' occurs. The value of α corresponds to the learning rate which indicates the weight of a single action has regarding its quality index. On the other hand, the value of γ is a constant knows in the literature as the discount factor and it allows for the tuning of the importance of subsequent actions.

Assuming that the Q function exists and is representative of the state-action pairs of a given system, the usage of a policy expressed by

$$\pi(s) = arg\, max_a\{Q(s,a)\} \quad (2)$$

will lead to the choice, in any state, of the action that has a higher associated Q value which is the action that is deemed optimal. Therefore, the problem can be reduced to the estimation of this very same function and can be solved by means of RL in a Q-learning approach.

B. Reinforcement Learning

RL is a Machine Learning (ML) method based on the exploratory nature common to the psychology of many living beings [9]. This method utilizes iterative interaction of an agent with its environment in a way that allows it to refine the agent's behavior by considering, as feedback, the reward associated with the corresponding state-action pair, as it is schematically represented in Figure 2.

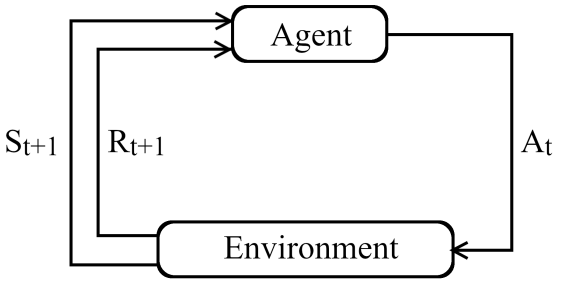


Fig. 2: Interaction schematic between agent and environment in RL paradigm. Adapted from [10]

The Q-learning algorithm enables a optimal choice policy to be found for any finite MDP, through the accumulated experience at each of the iterations. A policy is defined as a decision function that specifies what the agent is going to execute given the perceived environment conditions. The experience allows for the inference of the quality Q of all the actions by applying Equation 1 and the consequent policy defined upon that quality using Equation 2. This process is named Temporal Difference Learning (TDL).

Note that this process suffers from the problem of converging in local minima points from which it can't evolve any further. As a solution to this characteristics a variant of this algorithm designated as ϵ-greedy Q-learning is typically used. In the case of the latter, during the exploration process it is picked with probability ϵ a random action reinforcing the exploratory nature of the algorithm. With probability $1 - \epsilon$ an action is chosen by using the calculated policy.

III. METHODS

A. Simulation Environment

As it can be observed from Figure 3, the real environ-

Fig. 3: Experimental configuration of the robotic wire and loop game [4]

ment in which the robot actuates is quite complex, therefore some abstractions are made towards a simpler environment of simulation. Firstly, the movement of the robot's end-effector is considered to be reduced to only two degrees of freedom (DOF), confided in the horizontal plane that contains the wire. In reality, the information regarding the state of the robot would be acquired by means of a camera mounted over the robot. The acquisition and processing of the image captured by the camera is outside the scope of the work presented here.

Having this, a representation of the environment based on the layer structure of an image in the Red-Green-Blue (RGB) color space is considered. Each of the layers represents the wire, the limits of the loop and the objective, respectively. The stratified representation is illustrated in Figure 4.

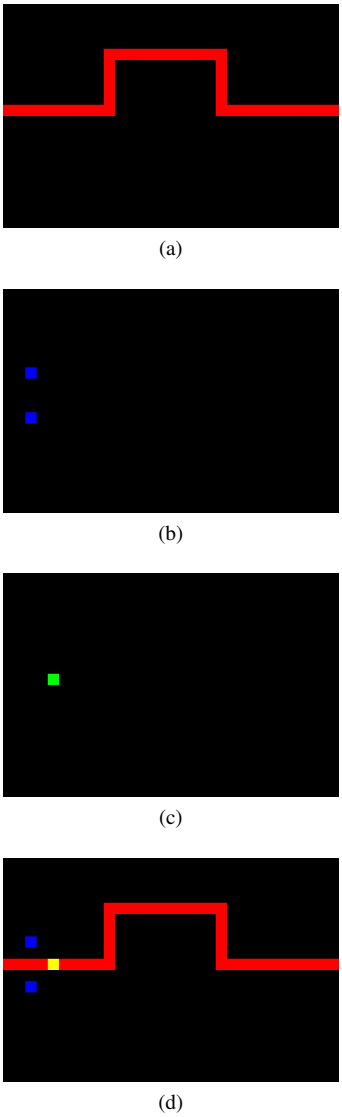
In the sense of modelling the problem as a MDP there is the need to define the action space and the features that compose a state from the point of view of the agent. For the action space six possibilities are considered:

- Up
- Down
- Left
- Right
- Rotate clockwise
- Rotate counter-clockwise.

For the state definition, the following features are extracted from the five pixel width square centered in the middle point of the loop:

- Positions of the path
- Position of the objective
- Rotation of the loop.

This limitation imposed on the information that the agent has for the current state allows it to generalize its knowledge to path configurations different from the one that was used to train it. This is valid as long as the new paths can be decomposed in segments that are found in the agent's look up table, therefore the choice of a training path with a high diversity of features is important.



(a) (b) (c) (d)

Fig. 4: (a) Wire Layer. (b) Loop limits layer. (c) Objective Layer. (d) Combination of the layers into a single map.

B. *Training process*

As the exploration of the environment by the agent progresses, information regarding identified states and the rewards associated with each of the taken actions should be kept. This information will compose the look up table of the Q values that will be updated during the training process.

To make this process as efficient as possible, each episode

of training is divided into two separate parts, a training phase and a testing phase. In Algorithm 1 this separation can be observed by the horizontal line that isolates the two phases. Initially, in the training phase, the agent explores its environment by maintaining a balance between random action selection and selecting actions based on past experiences building more knowledge about the surroundings. In this phase, every time a state that hasn't been observed before is verified then it is initialized in the look up table. The dynamic initialization of states eliminates the need to previously analyze all the possibilities of states that will exist and reduces the memory used in states that will actually never be verified in the training process.

Algorithm 1 ϵ-*greedy* Q-learning algorithm

initialize $Q(s_{init}, a)$
for cycle from 1 to N_c **do**
  if cycle = 1 **then**
    $init^{pos}_{cycle}$ set to starting-point of the wire
  else
    $init^{pos}_{cycle}$ set to $stable^{pos}_{cycle-1}$
  end if
  $\epsilon = \epsilon_{init}$
  for episode from 1 to N_e **do**
    while true **do**
      observe current state s
      if $n_{random} < \epsilon$ **then**
        select random action a
      else
        select $a = argmax_{a'}\{Q(s, a')\}$
      end if
      Apply action a to environment and observe new state s' and reward r
      if s' is unknown **then**
        initialize $Q(s', a')$
      end if
      Update Q with equation 1
      Update state s = s'
      if end condition found **then**
        break
      end if
    end while
    if $\epsilon > \epsilon_{min}$ **then**
      $\epsilon = \epsilon - 1/N_e$
    end if
  end for
  Initialize environment to $init^{pos}_{cycle}$
  while true **do**
    select $a = argmax_{a'}\{Q(s, a')\}$
    perform action a and observe new state s' and reward r
    if s' is stable state **then**
      $stable^{pos}_{cycle} = current_{pos}$
    end if
    Update Q with equation 1
    Update state s = s'
    if end condition found **then**
      break
    end if
  end while
end for

In the testing phase actions are chosen purely based on what has been previously learned in order to evaluate the judgement of the agent. This evaluation consists in observing how far the agent can take the loop along the wire without raising any stopping condition. The following training cycle in initiated in the position that is considered stable, therefore avoiding that the portion of the wire that has already been learned to be traveled again. A stable position corresponds to the last position of the loop when an objective was reached immediately before a stop condition.

A training cycle set of from the stable position found in the previous testing phase, or from the initial position if it is the first cycle of the process. Actions are taken in a ϵ-greedy fashion until a stop condition is met. This randomized behavior guarantees that a minimum level of exploration is achieved, independently of how good the policy of the agent may have become. It is therefore avoided that, with the convergence of the Q function, a greedy strategy is implemented, leading to a convergence in a possible local minima where it wouldn't be able to evolve any further.

Considered stop conditions are the following:

- Overlap of the loop and the wire
- Loop leaves the defined simulation area
- Path is completed
- Current objective leaves the 5 pixel wide square associated with the current state

Regarding rewards, both the situation of contact between the loop and the wire and leaving the simulation area incur in a negative reward. Reaching an objective is reflected by a positive reward in order to positively reinforce this behavior.

C. Objective update

When an objective is reached its location needs to be updated further down the wire. This update process is done automatically using only the information that can be perceived from the environment in order to guarantee that there are no implicit path defined by previously planned positions of the objective.

Considering the state square, the considered candidates are the points that form a cross centered in the middle point of the loop (Figure 5b). The candidates that don't coincide with the path are removed (Figure 5c). Then all the points in the direction that is opposite to the last direction of movement

of the loop are disregarded. In the case of Figure 5d it is considered that the loop moved to the right therefore candidates to the left are removed. Finally, from the remaining candidates the point that is further from the current objective is selected as it is seen in Figure 5e.

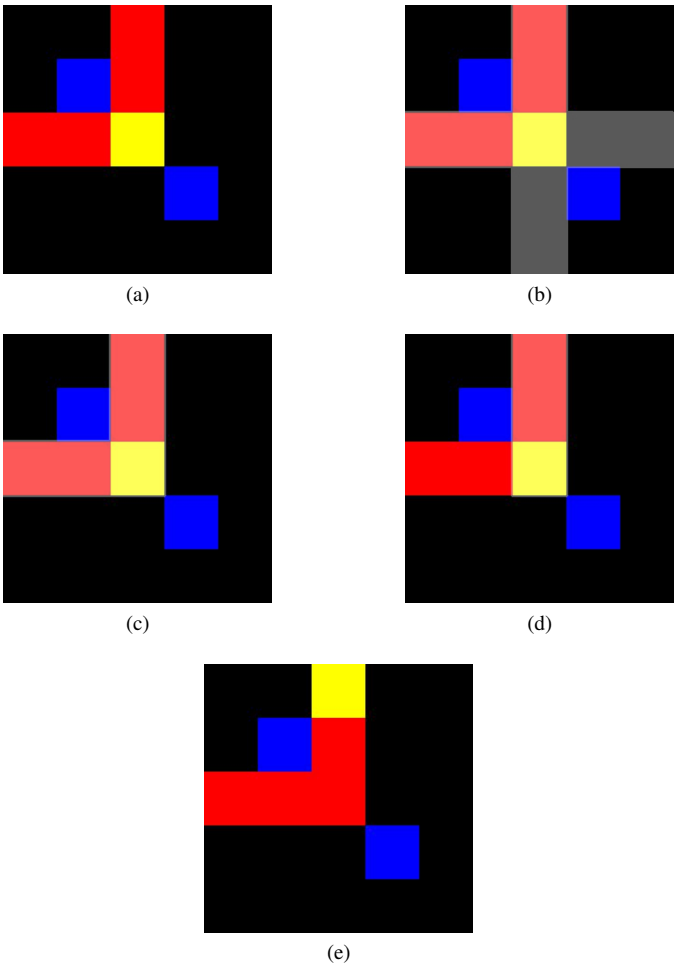


Fig. 5: Objective update algorithm steps. (a) Initial condition. (b) Initial candidates. (c) Path intersection candidates. (d) Direction based candidate filtering. (e) Final objective placement.

IV. Results and Discussion

The proposed methods were implemented and tested using 50 training episodes with 200 exploration cycles each. The constants α and γ both were set to 0.5. The training path is represented in Figure 4d. The process was completed in $22.7s$ in MATLAB R2018a running on Ubuntu 21.10 using a AMD Ryzen 9 processor. These configurations allowed for the agent to learn how to solve the path autonomously.

The progression of the agent's capacity can be observed in Figure 1. It can be verified that in 24 training cycles the agent is capable of solving the wire and loop game. It is also worth noting the horizontal sections of the graph where the distance stabilizes. These sections correspond to the parts of the wire where there is a curve and are therefore harder to learn for the agent. To verify the knowledge acquired by the agent, it is used

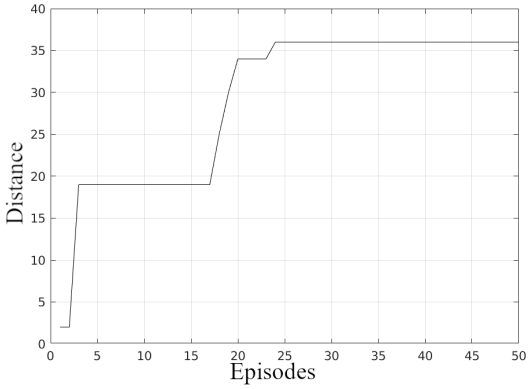


Fig. 6: Evolution of the covered distance with respect to the completed training cycles

to solve the same path used for the training and a different one. In Figure 7 four iterations of the verification process on the training path can be observed. Figure 8 is the analog but for a different path. Both were completed successfully e presented a behavior very similar to what was expected particularly in curvature points of the wire, as it can be observed in the mentioned examples.

The verification process takes on average $110ms$ which means that an action is selected based on the look up table in under $10ms$ each. This indicates that the described process could very well be used for real-time applications, as is the case of the control of an actual robotic manipulator.

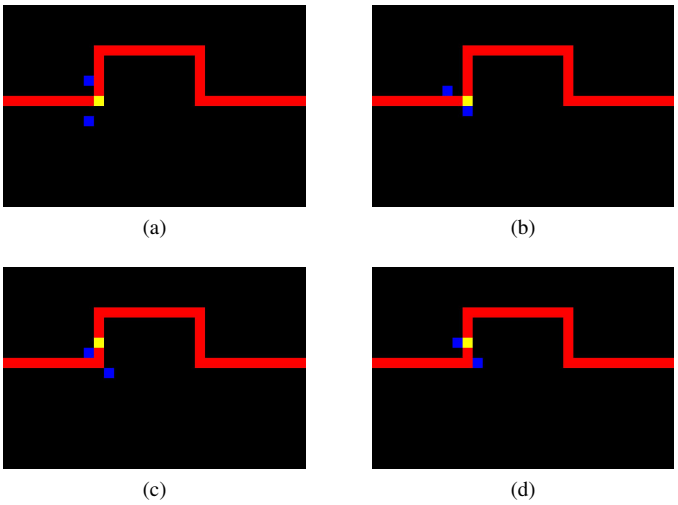


Fig. 7: Iterations of the verification process with training path

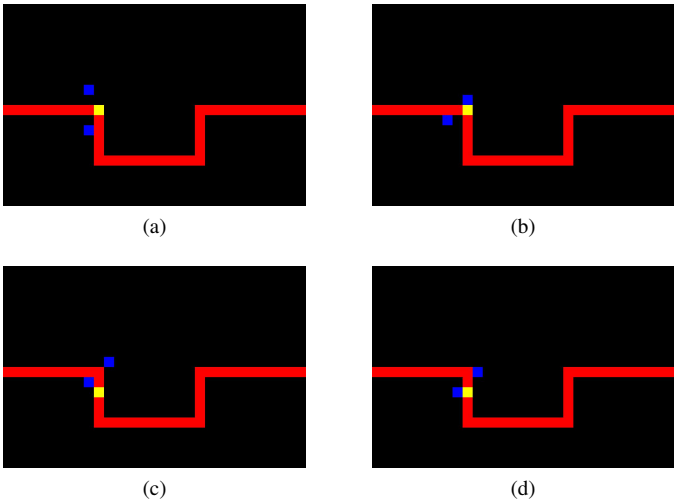


Fig. 8: Iterations of the verification process with path different from the training

V. Conclusion

Concluding, the application of RL algorithms to the motion planning of robotic manipulators is a viable solution with the correct formulation of the learning problem. This approach provides manipulators with the flexibility that is needed to endure variations in production processes. The presented solution, being directed to the following of a generic trajectory, can be applied to a multitude of implementations, namely welding, gluing, cutting, manipulation, among others.

Note that the considered paths are 4-connected which imposes a limitation on the complexity of the paths that can be considered. To extend this solution to more intricate problems, the simulation environment would have to be changed accordingly. This being the case, it would be beneficial to use more complete physical simulation platforms such as *Gazebo*, *NVIDIA Isaac* or *MuJoCo*. This would also allow for the extension of the problem to three dimensions instead of the two that are seen here.

References

[1] L. Monostori, "Cyber-physical production systems: Roots, expectations and r&d challenges," *Procedia CIRP*, vol. 17, pp. 9–13, 2014, Variety Management in Manufacturing, ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2014.03.115. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827114003497.

[2] W. H. Sunada and S. Dubowsky, "On the Dynamic Analysis and Behavior of Industrial Robotic Manipulators With Elastic Members," *Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 105, no. 1, pp. 42–51, Mar. 1983, ISSN: 0738-0666. DOI: 10.1115/1.3267343. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/105/1/42/5749154/42_1.pdf. [Online]. Available: https://doi.org/10.1115/1.3267343.

[3] A. Grau, M. Indri, L. L. Bello, and T. Sauter, "Industrial robotics in factory automation: From the early stage to the internet of things," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017, pp. 6159–6164. DOI: 10.1109/IECON.2017.8217070.

[4] R. Meyes, H. Tercan, S. Roggendorf, *et al.*, "Motion planning for industrial robots using reinforcement learning," *Procedia CIRP*, vol. 63, pp. 107–112, 2017, Manufacturing Systems 4.0 – Proceedings of the 50th CIRP Conference on Manufacturing Systems, ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2017.03.095. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221282711730241X.

[5] Y. Zhu, Z. Wang, J. Merel, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," Feb. 2018. DOI: 10.15607/rss.2018.xiv.009. [Online]. Available: https://arxiv.org/abs/1802.09564v2.

[6] R. Sutton, "Learning to predict by the method of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, Aug. 1988. DOI: 10.1007/BF00115009.

[7] M. L. Puterman, "Chapter 8 markov decision processes," in *Stochastic Models*, ser. Handbooks in Operations Research and Management Science, vol. 2, Elsevier, 1990, pp. 331–434. DOI: https://doi.org/10.1016/S0927-0507(05)80172-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050705801720.

[8] R. Bellman, "On the theory of dynamic programming," *Proceedings of the National Academy of Sciences*, vol. 38, no. 8, pp. 716–719, 1952, ISSN: 0027-8424. DOI: 10.1073/pnas.38.8.716. eprint: https://www.pnas.org/content/38/8/716.full.pdf. [Online]. Available: https://www.pnas.org/content/38/8/716.

[9] R. S. Sutton, "Introduction: The challenge of reinforcement learning," in *Reinforcement Learning*, R. S. Sutton, Ed. Boston, MA: Springer US, 1992, pp. 1–3, ISBN: 978-1-4615-3618-5. DOI: 10.1007/978-1-4615-3618-5_1. [Online]. Available: https://doi.org/10.1007/978-1-4615-3618-5_1.

[10] Z. Ding, Y. Huang, H. Yuan, and H. Dong, "Introduction to reinforcement learning," in *Deep Reinforcement Learning: Fundamentals, Research and Applications*, H. Dong, Z. Ding, and S. Zhang, Eds. Singapore: Springer Singapore, 2020, pp. 47–123, ISBN: 978-981-15-4095-0. DOI: 10.1007/978-981-15-4095-0_2.

References

[1] S. Choi, W. J. Eakins, and T. A. Fuhlbrigge, "Trends and opportunities for robotic automation of trim & final assembly in the automotive industry," *2010 IEEE International Conference on Automation Science and Engineering, CASE 2010*, pp. 124–129, 2010. DOI: 10.1109/COASE.2010.5584524.

[2] J. Frohm, V. Lindström, M. Winroth, and J. Stahre, "The industry's view on automation in manufacturing," *IFAC Proceedings Volumes*, vol. 39, no. 4, pp. 453–458, Jan. 2006, ISSN: 1474-6670. DOI: 10.3182/20060522-3-FR-2904.00073.

[3] A. H. Quispe, H. B. Amor, and H. I. Christensen, "Combining arm and hand metrics for sensible grasp selection," *IEEE International Conference on Automation Science and Engineering*, vol. 2016-November, pp. 1170–1176, Nov. 2016, ISSN: 21618089. DOI: 10.1109/COASE.2016.7743537.

[4] J. R. Napier, "The prehensile movements of the human hand," *The Journal of bone and joint surgery. British volume*, vol. 38-B, no. 4, pp. 902–913, 1956, ISSN: 0301-620X. DOI: 10.1302/0301-620X.38B4.902. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/13376678/.

[5] M. R. Cutkosky, "On Grasp Choice, Grasp Models, and the Design of Hands for Manufacturing Tasks," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 269–279, 1989, ISSN: 1042296X. DOI: 10.1109/70.34763.

[6] S. A. Pertuz, C. H. Llanos, and D. M. Munoz, "Development of a Robotic Hand Using Bioinspired Optimization for Mechanical and Control Design: UnB-Hand," *IEEE Access*, vol. 9, pp. 61 010–61 023, 2021, ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3073010.

[7] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous Robots*, vol. 38, no. 1, pp. 65–88, Jan. 2015, ISSN: 15737527. DOI: 10.1007/S10514-014-9402-3/TABLES/2. [Online]. Available: https://link.springer.com/article/10.1007/s10514-014-9402-3.

[8] D. Prattichizzo and J. C. Trinkle, "Grasping," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 671–700, ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_29.

[9] M. A. Roa and R. Suarez, "Computation of independent contact regions for grasping 3-d objects," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 839–850, 2009. DOI: 10.1109/TRO.2009.2020351.

[10] I. Kao, K. Lynch, and J. W. Burdick, "Contact modeling and manipulation," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 647–669, ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_28.

[11] N. Rojas and A. M. Dollar, "Classification and Kinematic Equivalents of Contact Types for Fingertip-Based Robot Hand Manipulation," *Journal of Mechanisms and Robotics*, vol. 8, no. 4, Mar. 2016, 041014, ISSN: 1942-4302. DOI: 10.1115/1.4032865. eprint: https://asmedigitalcollection.asme.org/mechanismsrobotics/article-pdf/8/4/041014/6254030/jmr_008_04_041014.pdf. [Online]. Available: https://doi.org/10.1115/1.4032865.

[12] K. Harada, M. Kaneko, and T. Tsuji, "Rolling based manipulation for multiple objects," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 4, 2000, 3887–3894 vol.4. DOI: 10.1109/ROBOT.2000.845337.

[13] M. Ciocarlie, C. Lackner, and P. Allen, "Soft finger model with adaptive contact geometry for grasping and manipulation tasks," in *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*, 2007, pp. 219–224. DOI: 10.1109/WHC.2007.103.

[14] B. León, A. Morales, and J. Sancho-Bru, "Robot grasping foundations," in *From Robot to Human Grasping Simulation*. Cham: Springer International Publishing, 2014, pp. 15–31, ISBN: 978-3-319-01833-1. DOI: 10.1007/978-3-319-01833-1_2.

[15] E. Rimon and J. Burdick, "On force and form closure for multiple finger grasps," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, 1996, 1795–1800 vol.2. DOI: 10.1109/ROBOT.1996.506972.

[16] K. Lakshminarayana, "Mechanics of form closure," *ASME Paper*, vol. 78-DET-32, 1978.

[17] R. M. Murray, S. S. Sastry, and L. Zexiang, "Multifingered hand kinematics," in *A Mathematical Introduction to Robotic Manipulation*, 1st. USA: CRC Press, Inc., 1994, pp. 211–259, ISBN: 0849379814.

[18] D. Guo, F. Sun, J. Zhang, and H. Liu, "A grasp synthesis and grasp synergy analysis for anthropomorphic hand," *2013 IEEE International Conference on Robotics and Biomimetics, ROBIO 2013*, pp. 1617–1622, 2013. DOI: 10.1109/ROBIO.2013.6739698.

[19] M. Schieber, "Muscular production of individuated finger movements: The roles of extrinsic finger muscles," *Journal of Neuroscience*, vol. 15, no. 1, pp. 284–297, 1995, ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.15-01-00284.1995. eprint: https://www.jneurosci.org/content/15/1/284.full.pdf.

[20] M. Santello, M. Flanders, and J. F. Soechting, "Postural Hand Synergies for Tool Use," *Journal of Neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, Dec. 1998, ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.18-23-10105.1998.

[21] I. T. Jolliffe, "Introduction," in *Principal Component Analysis*. New York, NY: Springer New York, 2002, pp. 1–9, ISBN: 978-0-387-22440-4. DOI: 10.1007/0-387-22440-8_1.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[23] R. S. Sutton, "Introduction: The challenge of reinforcement learning," in *Reinforcement Learning*, R. S. Sutton, Ed. Boston, MA: Springer US, 1992, pp. 1–3, ISBN: 978-1-4615-3618-5. DOI: 10.1007/978-1-4615-3618-5_1.

[24] Z. Ding, Y. Huang, H. Yuan, and H. Dong, "Introduction to reinforcement learning," in *Deep Reinforcement Learning: Fundamentals, Research and Applications*, H. Dong, Z. Ding, and S. Zhang, Eds. Singapore: Springer Singapore, 2020, pp. 47–123, ISBN: 978-981-15-4095-0. DOI: 10.1007/978-981-15-4095-0_2.

[25] R. Sutton, "Learning to predict by the method of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, Aug. 1988. DOI: 10.1007/BF00115009.

[26] M. Saeed, M. Nagdi, B. Rosman, and H. H. Ali, "Deep Reinforcement Learning for Robotic Hand Manipulation," *Proceedings of: 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering, ICCCEEE 2020*, Feb. 2021. DOI: 10.1109/ICCCEEE49695.2021.9429619.

[27] S. W. Ruehl, C. Parlitz, G. Heppner, A. Hermann, A. Roennau, and R. Dillmann, "Experimental evaluation of the schunk 5-finger gripping hand for grasping tasks," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, 2014, pp. 2465–2470. DOI: 10.1109/ROBIO.2014.7090710.

[28] *Schunk svh ros driver*. [Online]. Available: http://wiki.ros.org/schunk_svh_driver.

[29] F. Ficuciello, A. Federico, V. Lippiello, and B. Siciliano, "Synergies evaluation of the schunk s5fh for grasping control," *Springer Proceedings in Advanced Robotics*, vol. 4, pp. 225–233, 2018, ISSN: 25111264. DOI: 10.1007/978-3-319-56802-7_24/FIGURES/6. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-56802-7_24.

[30] A. Cloutier and J. Yang, "Grasping force optimization approaches for anthropomorphic hands," *Journal of Mechanisms and Robotics*, vol. 10, no. 1, Feb. 2018, ISSN: 19424310. DOI: 10.1115/1.4038684/377208.

[31] I. A. Gal, D. Bucur, and L. Vladareanu, "DSmT Decision-Making Algorithms for Finding Grasping Configurations of Robot Dexterous Hands," *Symmetry 2018, Vol. 10, Page 198*, vol. 10, no. 6, p. 198, Jun. 2018, ISSN: 20738994. DOI: 10.3390/SYM10060198.

[32] A. Oliver, S. Kang, B. Wünsche, and B. Macdonald, "Using the kinect as a navigation sensor for mobile robotics," Nov. 2012, pp. 509–514. DOI: 10.1145/2425836.2425932.

[33] Y. Fan, T. Tang, H. C. Lin, and M. Tomizuka, "Real-Time Grasp Planning for Multi-Fingered Hands by Finger Splitting," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4045–4052, Dec. 2018, ISSN: 21530866. DOI: 10.1109/IROS.2018.8594369. arXiv: 1804.00050.

[34] N. Vahrenkamp, E. Koch, M. Wachter, and T. Asfour, "Planning High-Quality Grasps Using Mean Curvature Object Skeletons," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 911–918, Apr. 2018, ISSN: 23773766. DOI: 10.1109/LRA.2018.2792694. arXiv: 1710.02418.

[35] S. Yao, M. Ceccarelli, G. Carbone, and Z. Dong, "Grasp configuration planning for a low-cost and easy-operation underactuated three-fingered robot hand," *Mechanism and Machine Theory*, vol. 129, pp. 51–69, Nov. 2018, ISSN: 0094-114X. DOI: 10.1016/J.MECHMACHTHEORY.2018.06.019.

[36] Z. Zhao, W. Shang, H. He, and Z. Li, "Grasp prediction and evaluation of multi-fingered dexterous hands using deep learning," *Robotics and Autonomous Systems*, vol. 129, p. 103 550, Jul. 2020, ISSN: 0921-8890. DOI: 10.1016/J.ROBOT.2020.103550.

[37] H. Xue, S. Wen, C. Yang, and H. Liu, "Model based reinforcement learning for robot grasping trajectory generation," *2019 4th IEEE International Conference on Advanced Robotics and Mechatronics, ICARM 2019*, pp. 720–726, Jul. 2019. DOI: 10.1109/ICARM.2019.8834254.

[38] P. Jia, L. Wu, G. Wang, W. N. Geng, F. Yun, and N. Zhang, "Grasping Torque Optimization for a Dexterous Robotic Hand Using the Linearization of Constraints," *Mathematical Problems in Engineering*, vol. 2019, pp. 1–17, Nov. 2019, ISSN: 1024-123X. DOI: 10.1155/2019/5235109.

[39] C. Islek and E. Ozdemir, "Design of a fuzzy safety margin derivation system for grip force control of robotic hand in precision grasp task:," vol. 18, no. 3, May 2021, ISSN: 17298814. DOI: 10.1177/17298814211018055.

[40] Z. Deng, Y. Jonetzko, L. Zhang, and J. Zhang, "Grasping Force Control of Multi-Fingered Robotic Hands through Tactile Sensing for Object Stabilization," *Sensors 2020, Vol. 20, Page 1050*, vol. 20, no. 4, p. 1050, Feb. 2020, ISSN: 14248220. DOI: 10.3390/S20041050. [Online]. Available: https://www.mdpi.com/1424-8220/20/4/1050/htm%20https://www.mdpi.com/1424-8220/20/4/1050.

[41] C. Rosales, R. Suárez, M. Gabiccini, and A. Bicchi, "On the synthesis of feasible and prehensile robotic grasps," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 550–556, 2012, ISSN: 10504729. DOI: 10.1109/ICRA.2012.6225238.

[42] Z. Liu, Z. Wu, T. Dong, X. Zhu, and K. Xu, "Reach-to-Grasp Planning for a Synergy-Controlled Robotic Hand Based on Grasp Quality Prediction," *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*, pp. 778–783, Jul. 2018. DOI: 10.1109/ROBIO.2018.8665139.

[43] N. S. Pollard, "Closure and quality equivalence for efficient synthesis of grasps from examples," *The International Journal of Robotics Research*, vol. 23, no. 6, pp. 595–613, 2004. DOI: 10.1177/0278364904044402.

[44] F. Ficuciello, "Synergy-Based Control of Underactuated Anthropomorphic Hands," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1144–1152, Feb. 2019, ISSN: 15513203. DOI: 10.1109/TII.2018.2841043.

[45] C. Mizera, T. Delrieu, V. Weistroffer, C. Andriot, A. Decatoire, and J. P. Gazeau, "Evaluation of Hand-Tracking Systems in Teleoperation and Virtual Dexterous Manipulation," *IEEE Sensors Journal*, vol. 20, no. 3, pp. 1642–1655, Feb. 2020, ISSN: 15581748. DOI: 10.1109/JSEN.2019.2947612.

[46] L. Jiang, B. Liu, S. Fan, and H. Liu, "A Posture Planning Method in Clustered Synergy Subspace for HIT/DLR Hand II," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11745 LNAI, pp. 134–145, Aug. 2019, ISSN: 16113349. DOI: 10.1007/978-3-030-27529-7_12. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-27529-7_12.

[47] A. Sadun, J. Jalani, and J. Sukor, "An overview of active compliance control for a robotic hand," Dec. 2014.

[48] A. S. Sadun, J. Jalani, J. Abdul Sukor, and F. Jamil, "Force control for a 3-Finger Adaptive Robot Gripper by using PID controller," *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation, ROMA 2016*, Feb. 2017. DOI: 10.1109/ROMA.2016.7847807.

[49] N. Morita, H. Nogami, E. Higurashi, and R. Sawada, "Grasping Force Control for a Robotic Hand by Slip Detection Using Developed Micro Laser Doppler Velocimeter," *Sensors 2018, Vol. 18, Page 326*, vol. 18, no. 2, p. 326, Jan. 2018, ISSN: 14248220. DOI: 10.3390/S18020326. [Online]. Available: https://www.mdpi.com/1424-8220/18/2/326/htm%20https://www.mdpi.com/1424-8220/18/2/326.

[50] F. Veiga, B. Edin, and J. Peters, "Grip Stabilization through Independent Finger Tactile Feedback Control," *Sensors 2020, Vol. 20, Page 1748*, vol. 20, no. 6, p. 1748, Mar. 2020, ISSN: 14248220. DOI: 10.3390/S20061748. [Online]. Available: https://www.mdpi.com/1424-8220/20/6/1748/htm%20https://www.mdpi.com/1424-8220/20/6/1748.

[51] D. Ortenzi, U. Scarcia, R. Meattini, G. Palli, and C. Melchiorri, "Synergy-Based Control of Anthropomorphic Robotic Hands with Contact Force Sensors," *IFAC-PapersOnLine*, vol. 52, no. 15, pp. 340–345, Jan. 2019, ISSN: 2405-8963. DOI: 10.1016/J.IFACOL.2019.11.698.

[52] S. Yajima, T. Shimono, T. Mizoguchi, and K. Ohnishi, "Automatic grasping position adjustment for robotic hand by estimating center of gravity using disturbance observer," *International Conference on Advanced Intelligent Mechatronics, AIM*, vol. 2020-July, pp. 595–600, Jul. 2020. DOI: 10.1109/AIM43001.2020.9159000.

[53] S. A. Pertuz, C. Llanos, C. A. Pena, and D. Munoz, "A Modular and Distributed Impedance Control Architecture on a Chip for a Robotic Hand," *31st Symposium on Integrated Circuits and Systems Design, SBCCI 2018*, Nov. 2018. DOI: 10.1109/SBCCI.2018.8533266.

[54] V. R. Garate, M. Pozzi, D. Prattichizzo, and A. Ajoudani, "A bio-inspired grasp stiffness control for robotic hands," *Frontiers Robotics AI*, vol. 5, no. JUN, p. 89, 2018, ISSN: 22969144. DOI: 10.3389/FROBT.2018.00089/BIBTEX.

[55] V. R. Garate, M. Pozzi, D. Prattichizzo, N. Tsagarakis, and A. Ajoudani, "Grasp Stiffness Control in Robotic Hands Through Coordinated Optimization of Pose and Joint Stiffness," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3952–3959, Oct. 2018, ISSN: 23773766. DOI: 10.1109/LRA.2018.2858271.

[56] J. Camilo Vasquez Tieck, K. Secker, J. Kaiser, A. Roennau, and R. Dillmann, "Soft-Grasping with an Anthropomorphic Robotic Hand Using Spiking Neurons," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2894–2901, Apr. 2021, ISSN: 23773766. DOI: 10.1109/LRA.2020.3034067.

[57] F. Song, Z. Zhao, W. Ge, W. Shang, and S. Cong, "Learning Optimal Grasping Posture of Multi-Fingered Dexterous Hands for Unknown Objects," *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*, pp. 2310–2315, Jul. 2018. DOI: 10.1109/ROBIO.2018.8665277.

[58] H. Merzic, M. Bogdanovic, D. Kappler, L. Righetti, and J. Bohg, "Leveraging contact forces for learning to grasp," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 3615–3621, May 2019, ISSN: 10504729. DOI: 10.1109/ICRA.2019.8793733. arXiv: 1809.07004.

[59] D. E. Kim, A. Li, and J. M. Lee, "Stable grasping of robotic hand using objective function," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, vol. 2019-July, pp. 538–543, Jul. 2019. DOI: 10.1109/AIM.2019.8868615.

[60] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, 2000, 348–353 vol.1. DOI: 10.1109/ROBOT.2000.844081.

[61] A. Ohev-Zion and A. Shapiro, "Grasping of deformable objects applied to organic produce," in *Towards Autonomous Robotic Systems*, R. Groß, L. Alboul, C. Melhuish, M. Witkowski, T. J. Prescott, and J. Penders, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 396–397, ISBN: 978-3-642-23232-9.

[62] L. Zaidi, J. A. Corrales, B. C. Bouzgarrou, Y. Mezouar, and L. Sabourin, "Model-based strategy for grasping 3D deformable objects using a multi-fingered robotic hand," *Robotics and Autonomous Systems*, vol. 95, pp. 196–206, Sep. 2017, ISSN: 0921-8890. DOI: 10.1016/J.ROBOT.2017.06.011.

[63] F. Nadon and P. Payeur, "Automatic selection of grasping points for shape control of non-rigid objects," *IEEE International Symposium on Robotic and Sensors Environments, ROSE 2019 - Proceedings*, Jun. 2019. DOI: 10.1109/ROSE.2019.8790383.

[64] F. Nadon and P. Payeur, "Grasp selection for in-hand robotic manipulation of non-rigid objects with shape control," *SYSCON 2020 - 14th Annual IEEE International Systems Conference, Proceedings*, Aug. 2020. DOI: 10.1109/SYSCON47679.2020.9275929.

[65] A. Montaño and R. Suárez, "Dexterous Manipulation of Unknown Objects Using Virtual Contact Points," *Robotics 2019, Vol. 8, Page 86*, vol. 8, no. 4, p. 86, Oct. 2019, ISSN: 22186581. DOI: 10.3390/ROBOTICS8040086. [Online]. Available: https://www.mdpi.com/2218-6581/8/4/86/htm%20https://www.mdpi.com/2218-6581/8/4/86.

[66] A. Montaño and R. Suárez, "Dexterous Manipulation of Unknown Objects Using Virtual Contact Points," *Robotics 2019, Vol. 8, Page 86*, vol. 8, no. 4, p. 86, Oct. 2019, ISSN: 22186581. DOI: 10.3390/ROBOTICS8040086. [Online]. Available: https://www.mdpi.com/2218-6581/8/4/86/htm%20https://www.mdpi.com/2218-6581/8/4/86.

[67] K. Higashi, K. Koyama, R. Ozawa, K. Nagata, W. Wan, and K. Harada, "Functionally divided manipulation synergy for controlling multi-fingered hands," *IEEE International Conference on Intelligent Robots and Systems*, pp. 9190–9197, Oct. 2020, ISSN: 21530866. DOI: 10.1109/IROS45743.2020.9341766. arXiv: 2003.11699.

[68] A. Nagabandi, K. Konoglie, S. Levine, V. Kumar, and G. Brain, "Deep Dynamics Models for Learning Dexterous Manipulation," Sep. 2019, ISSN: 2331-8422. arXiv: 1909.11652. [Online]. Available: https://arxiv.org/abs/1909.11652v1.

[69] A. Nair, D. Chen, P. Agrawal, *et al.*, "Combining self-supervised learning and imitation for vision-based rope manipulation," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2146–2153, Jul. 2017, ISSN: 10504729. DOI: 10.1109/ICRA.2017.7989247. arXiv: 1703.02018.

[70] D. Jain, A. Li, S. Singhal, A. Rajeswaran, V. Kumar, and E. Todorov, "Learning deep visuomotor policies for dexterous hand manipulation," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 3636–3643, May 2019, ISSN: 10504729. DOI: 10.1109/ICRA.2019.8794033.

[71] O. A. M. Andrychowicz, B. Baker, M. Chociej, *et al.*, "Learning dexterous in-hand manipulation:" *https://doi.org/10.1177/0278364919887447*, vol. 39, no. 1, pp. 3–20, Nov. 2019, ISSN: 17413176. DOI: 10.1177/0278364919887447. arXiv: 1808.00177. [Online]. Available: https://journals.sagepub.com/doi/10.1177/0278364919887447.

[72] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 3651–3657, May 2019, ISSN: 10504729. DOI: 10.1109/ICRA.2019.8794102. arXiv: 1810.06045.

[73] P. Rivera, E. V. Añazco, and T. S. Kim, "Object Manipulation with an Anthropomorphic Robotic Hand via Deep Reinforcement Learning with a Synergy Space of Natural Hand Poses," *Sensors 2021, Vol. 21, Page 5301*, vol. 21, no. 16, p. 5301, Aug. 2021, ISSN: 14248220. DOI: 10.3390/S21165301. [Online]. Available: https://www.mdpi.com/1424-8220/21/16/5301/htm%20https://www.mdpi.com/1424-8220/21/16/5301.

[74] P. Ruppel and J. Zhang, "Learning object manipulation with dexterous hand-arm systems from human demonstration," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5417–5424, Oct. 2020, ISSN: 21530866. DOI: 10.1109/IROS45743.2020.9340966.

[75] J. Orbik, A. Agostini, and D. Lee, "Inverse reinforcement learning for dexterous hand manipulation," *IEEE International Conference on Development and Learning, ICDL 2021*, Aug. 2021. DOI: 10.1109/ICDL49984.2021.9515637.

[76] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[77] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: A moveit! case study," Apr. 2014.

[78] *Moveit motion planning framework*. [Online]. Available: https://moveit.ros.org/.

[79] M. Quigley, K. Conley, B. Gerkey, *et al.*, "Ros: An open-source robot operating system," vol. 3, Jan. 2009.

[80] P. Ruppel, N. Hendrich, S. Starke, and J. Zhang, "Cost functions to specify full-body motion and multi-goal manipulation tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3152–3159. DOI: 10.1109/ICRA.2018.8460799.

[81] M. Körber, J. Lange, S. Rediske, S. Steinmann, and R. Glück, "Comparing popular simulation environments in the scope of robotics and reinforcement learning," Mar. 2021.

[82] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," Jun. 2016.

[83] 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.

[84] E. Drumwright, J. Hsu, N. Koenig, and D. Shell, "Extending open dynamics engine for robotics simulation," Nov. 2010, pp. 38–50, ISBN: 978-3-642-17318-9. DOI: 10.1007/978-3-642-17319-6_7.

[85] E. Coumans, "Bullet physics simulation," Jul. 2015, p. 1. DOI: 10.1145/2776880.2792704.

[86] J. Lee, M. X. Grey, S. Ha, *et al.*, "Dart: Dynamic animation and robotics toolkit," *Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018. DOI: 10.21105/joss.00500. [Online]. Available: https://doi.org/10.21105/joss.00500.

[87] M. Sherman, A. Seth, and S. Delp, "Simbody: Multibody dynamics for biomedical research," *Procedia IUTAM*, vol. 2, pp. 241–261, Dec. 2011. DOI: 10.1016/j.piutam.2011.04.023.

[88] J. M. Hsu and S. C. Peters, "Extending open dynamics engine for the darpa virtual robotics challenge," [Online]. Available: http://osrfoundation.org.

[89] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[90] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51 416–51 431, 2021. DOI: 10.1109/ACCESS.2021.3068769.

[91] S. Chitta, E. Marder-Eppstein, W. Meeussen, *et al.*, "Ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017. DOI: 10.21105/joss.00456. [Online]. Available: http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf.