POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale

Corso di Laurea Magistrale in Ingegneria Meccanica

A methodological approach to re-engineer the experience of consumer products based on the use of haptic devices

Relatore:
Ing. Francesco FERRISE
Correlatore:
Ing. Serena GRAZIOSI

Tesi di:
Guilherme PHILLIPS FURTADO
Matr. 764112

Anno Accademico 2012–2013

POLITECNICO DI MILANO

Sommario

Corso di Laurea Magistrale

A methodological approach to re-engineer the experience of consumer products based on the use of haptic devices

by Guilherme PHILLIPS FURTADO
Matr. 764112

Oggigiorno non esiste una maniera sistematica per catturare le preferenze tattili di un utente che interagisce con interfacce di prodotti di consumo quali cassetti, porte o pulsanti, né un metodo per tradurre tali preferenze in un meccanismo in grado di generarle. L'obiettivo del lavoro di tesi è di proporre un metodo che permetta alle compagnie che operano nel settore dei prodotti di consumo di re-ingegnerizzare il feedback tattile di tali prodotti sulla base delle preferenze degli utenti. Questo metodo si basa sull'uso di prototipi virtuali interattivi con i quali l'utente può interagire attraverso sistemi haptic a ritorno di forza. Tale ritorno di forza può essere modificato in base alle preferenze dell'utente grazie ad un modello di forze che controlla l'interfaccia haptic, completamente parametrizzato. Questo permette di trasformare una serie di feedback percettivi puramente qualitativi in specifiche quantitative. Attraverso algoritmi di ottimizzazione queste specifiche quantitative sono successivamente tradotte in un meccanismo reale. La porta di una lavastoviglie è stata utilizzata come caso di studio per dimostrare l'efficacia del metodo e identificarne i limiti.
keywords: haptics, tecnologia haptic, user experience design, virtual prototype, stima dei parametri

POLITECNICO DI MILANO

Abstract

Corso di Laurea Magistrale

A methodological approach to re-engineer the experience of consumer products based on the use of haptic devices

by Guilherme PHILLIPS FURTADO
Matr. 764112

Currently, there is no systematic way to evaluate users' tactile preference when interacting with parts of consumer' products such drawers, doors or buttons through the use of interactive virtual prototypes, nor a way to infer the specifications of the mechanism that would be responsible for the desired haptic feedback. The aim of this work is to describe a methodological approach that enables companies, operating in the consumer goods market, to re-engineer the haptic feedback from the interaction with such products according to users' preferences, by creating an easily modifiable virtual prototype that is interacted with through a haptic interface. The haptic feedback provided by the virtual prototype is readily modified at the user's request through the use of a parametric model that controls the haptic interface, responsible for transforming perceptual qualitative feedbacks into quantitative specifications. These quantitative specifications are transformed into the technical specifications of a real mechanism through the solution of an optimization problem. A dishwasher door has been used as case study in order demonstrate the effectiveness of the proposed methodology and to identify the limits.

keywords: haptics, haptic technology, user experience customization, virtual prototype, parameter estimation, mechanisms

Contents

Sommario **i**

Abstract **ii**

List of Figures **v**

List of Tables **vii**

1 Introduction **1**
- 1.1 Technology Overview . . . 2
- 1.2 Related Works . . . 3

2 Methodological Approach **5**

3 Physical System Modelling: Dishwasher Door **11**
- 3.1 Simulation Software . . . 11
- 3.2 Description of the Dishwasher Door . . . 12
- 3.3 Dynamic model in AMESim . . . 14
  - 3.3.1 Articulated mechanism . . . 14
    - 3.3.1.1 Subsystem DOOR . . . 18
    - 3.3.1.2 Subsystem PLATE . . . 18
    - 3.3.1.3 Subsystem RELATIVESPEED . . . 20
    - 3.3.1.4 Subsystem FRICTIONFORCE . . . 24
  - 3.3.2 Latch mechanism . . . 25
- 3.4 Measurements . . . 27

4 Haptic rendering **34**
- 4.1 Haptic Device . . . 34
- 4.2 Coding language and API . . . 36
  - 4.2.1 H3D API . . . 37
  - 4.2.2 Python . . . 37
  - 4.2.3 X3D . . . 37
- 4.3 Haptic Model . . . 37
  - 4.3.1 Transformation of Coordinates . . . 39
  - 4.3.2 Constraining the end-effector . . . 41
    - 4.3.2.1 First method: Radial direction stiffness . . . 41
      - Example: . . . 43
    - 4.3.2.2 Second method: Restitution force based on distance minimization . . . 43

4.3.3 Haptic Behavior Modelling . . . 49
4.3.3.1 Activation functions . . . 50
4.3.3.2 Unlocking Phase . . . 52
4.3.3.3 Locking phase . . . 53
4.3.3.4 Transition between locked and unlocked state . . . 54
4.3.3.5 Moving Phase . . . 54
4.4 Implementation . . . 57
Code Description . . . 59
X3D file . . . 59
Python file . . . 61
4.5 Experimental results . . . 63

5 Optimization **66**
5.1 Designing the problem: Dynamic system parameter estimation . . . 67
5.2 Solving the problem: Genetic Algorithm . . . 69
Chromosomes or Individuals . . . 69
Selection . . . 70
Recombination . . . 70
Mutation . . . 70
Methodology . . . 70
5.2.1 Parameter estimation of the system from measurements . . . 71
5.2.2 Parameter estimation of the haptic model from measurements 72
5.2.3 From haptic model to technical specification . . . 74
5.2.4 Redesigning the dishwasher mechanism . . . 76

6 Conclusion **80**

A Appendix A: X3D file for haptic rendering **82**

B Appendix B: Python file for haptic rendering **89**

Bibliography **103**

List of Figures

2.1 Methodological Approach 7

3.1 Description of components used from AMESim 13
3.2 Mechanical components of the Door 14
3.3 Sketch of the articulated mechanism 16
3.4 Dry friction as an hyperbolic tangent 17
3.5 Block diagram of the system 17
3.6 Block diagram of the subsystem DOOR 19
3.7 Block diagram of the subsystem PLATE 20
3.8 Diagram of the plate 21
3.9 Block diagram of the subsystem RELATIVESPEED 23
3.10 Block diagram of the subsystem FRICTIONFORCE 24
3.11 System representation of the latch mechanism 25
3.12 Block diagram for the latch mechanism 26
3.13 Force applied by the latch mechanism on AMESim model due to the spring 27
3.14 Stiffness of the latch mechanism in function of position on AMESim model 28
3.15 Sensor positioning for the measurements 30
3.16 Measured force when unlocking the door in function of time 31
3.17 Measured position when unlocking the door in function of time . . 31
3.18 Measured force when locking the door in function of time 32
3.19 Measured angular speed in function of time 33
3.20 Measured input force in function of time 33

4.1 Workspace of HapticMaster 35
4.2 Model for the spring acting on the radial direction of a curve . . . 42
4.3 Elliptic trajectory theoretically followed by the end effector 44
4.4 Force applied by the user at the end effector for the elliptical trajectory 44
4.5 Force magnitude necessary to maintain the end effector on the elliptic trajectory 45
4.6 Circular trajectory followed by the system 50
4.7 Magnitude of the force required to maintain the end effect on the circular trajectory 51
4.8 Kinetic energy comparison 51
4.9 Activation function 52
4.10 Analysis of force required to unlock the door 53
4.11 Asymmetry of dry friction 56

4.12 Variable dry friction 57
4.13 Velocity of the haptic model and the actual device - no friction . . 64
4.14 Force applied at the end effector - no friction 65

5.1 Optimization model 72
5.2 Comparison between the behavior of the system according on the estimated parameters 73
5.3 Optimization model for the haptic behavior 73
5.4 Haptic model in AMESim 76
5.5 Comparison between the response of the system and haptic model (const. friction) 77
5.6 Value of the coefficient $A(\theta)$ according to the position of the door . 79
5.7 Comparison between the response of the system and haptic model (var. friction) 79

List of Tables

5.1 Genetic algorithm parameters 74
5.2 Measured parameters 74
5.3 Estimated parameters of the system 75
5.4 Technical specifications obtained from the chosen haptic behavior . 78

Chapter 1

Introduction

There are numerous factors that influence the decision of people to purchase a product. It has been argued by researchers that what a product feels like can influence whether or not people will end up buying. For that reason, the people who are responsible for taking strategic decisions about new products, at the marketing division of companies, are urging design teams to give priority to the design the user experience, later focusing at the functional and technical features. Touching a product, in particular, plays an important role in our evaluation and appreciation of different products(Spence and Gallace [1]).

Currently, when designing the user experience of new products, companies may perform comparative tests involving potential customers as well as marketing experts, and taking as samples their own products or the products of competitors (Bhutta and Huq [2]). The result of these tests defines the expected behaviour of the new product, which becomes a sort of combination of the most appealing features of the tested products. Once the desired behaviour is defined, marketing experts must translate it into project targets, and then into design specifications with the help of research and development engineers. However, a methodological approach encompassing all these steps is yet to be implemented, being one of the issues how to translate these design specifications, which are mainly qualitative, into quantitative specifications that would allow engineers to recreate the product envisioned by the marketing experts. Another issue that emerges from this process is that the initial phase of testing is based only on products already available on the market, limiting the range of possible user experiences that could be tested by the experts [3].

1.1 Technology Overview

This work aims to address these issues on the process of designing the user experience by taking advantage of current virtual reality technologies that allow the interaction of users with virtual prototypes, focusing on the tactile experience. This way, it becomes possible for end users and marketing experts interact with the product on the initial phases of design, while allowing engineers to assert their preferences into quantitative parameters that govern the behaviour of the virtual prototype.

Virtual prototyping is a technique in the process of product development that involves mainly the use computer-aided design (CAD) and computer-aided engineering (CAE) software to simulate the behavior of the product in the real world, validating the design before committing to making a physical prototype. The flexibility offered by a virtual prototype allows designers to more quickly evaluate and explore different design alternatives, which can lead to improved quality and performance. They also allow the reduction of development costs and time required to bring the product to the market. It is possible for a person to physically interact with a virtual prototype by the use of virtual reality technologies.

Virtual reality is a term referring to computer-simulated environments capable of simulating physical presence in places in the real world, as well as in imaginary worlds. To create a believable experience, virtual reality environments have to take advantage of our sensory capabilities, particularly by rendering visual, auditory and tactile sensory information through the use of appropriate devices. Visual experiences are usually enabled by computer screens or stereoscopic displays, while auditory experiences can be created by sounds reproduced through speakers or headphones. Tactile information, also known as force feedback, can be generated through the use of haptic systems [4].

Haptic technology is a tactile feedback technology which takes advantage of the sense of touch by applying forces, vibrations, or motions to the user on a specified manner through the use of what is called haptic interface or haptic device. This mechanical stimulation can be used to assist in the creation of virtual objects in a computer simulation and to control such virtual objects. It can be said the relation of haptic systems with the sense of touch is the equivalent of the relation between computer graphics and vision and of audio speakers and headphones to the hearing senses.

By exploiting haptic system capabilities of interacting with virtual prototypes, it is possible to facilitate the communication between collaborators of different disciplinary backgrounds when designing a new product, such as allowing engineers to quantify the physical sensations described by marketing experts and end-users. It also allow a greater range of experiences to be tested, as the characteristics of the virtual prototype do not need to be limited by the characteristics of existing products.

This project will focus on developing a method to design a tactile interactive virtual prototype, which behavior is defined by mathematical equations that correspond to the forces that the real product would exert when the user interacts with it, forces that can be changed in real time and are rendered through a haptic interface, later translated into real technical specifications of the mechanical system that would enable the desired behavior to be reproduced into the real world. A dishwasher door is used as a case study, and the existing mechanism is re-designed to reproduce the specified behavior defined by marketing experts on the virtual prototype.

1.2 Related Works

Numerous studies have explored the interface between research and development teams and marketing teams and its role on the process of developing new products. Among many issues, difficulties in communication between these two tend to emerge due to difference on their working domains and their different models for the product representation [5, 6]. Another important issue is that customers' requirements tend to be described into imprecise and non-technical terms [7].

The design of multisensory aspects of traditional consumer products is becoming a common practice in different product areas [1], such as food, cars and domestic appliances and can help marketing people understand the demands of the customers, where the test of the perceptual feedback is usually based on the use of physical prototypes.

The use of physical prototypes is necessary when tests require the physical interaction with the product through the senses of touch and smell. With the technological advances in virtual reality, it is possible to simulate both visual appearance and the sound emitted by products faithfully through the use of virtual prototypes. The use of virtual prototypes is gaining interest in industry, due to their flexibility and growing fidelity, aside from the advantages brought by reduction of development cost and time. Traditionally, most of the tests performed on Virtual Prototypes have been purely visual, aiming at evaluating the aesthetic aspects of a new product. More recently, a number of haptic devices have been developed to simulate the haptic feedback with products and their components. Shin et al. [8], for example, developed custom haptic devices to simulate the physical interaction with refrigerator, while Strolz et al. [9] developed them to simulate car doors. The aim of these haptic-based virtual prototypes is testing design solutions concerning the haptic behavior, and eventually review the design of the product.

In the works of Bordegoni and Ferrise [10][11] it has been demonstrated the effective use of a multisensory environment based on touch, hearing and vision to capture users' preferences, already in the concept design phase. These initial studies have demonstrated the potentiality of Virtual Prototyping in capturing users' preferences, and have highlighted the necessity to change the way new products are designed[12]. Their use can improve the communication between marketing teams, engineering teams[3] and customers, and is focused on the present study.

Chapter 2

Methodological Approach

In [12], the authors outline some advantages and requirements for an interactive virtual prototype to become the digital substitute of the physical prototype. The ones addressed on the present work are:

- **It should be based on functional model for each technical domain** – that is, the virtual prototype is a functional model that characterizes and simulates the domain (or the domains) of the real product to evaluate and test. In this work, the domain addressed is related to the force feedback the real product should give to the user.
- **Sharable among different stakeholders** – an advantage of using virtual prototypes is that they can be shared over the network and accessed by different stakeholders at the same time.
- **Modifiable and parametric** – one of the most important advantages of Virtual Prototypes is that they can be easily modifiable and can be even parametric. By using the physical prototype of an object, an individual can generally only express an opinion as to whether he likes or dislikes it, but cannot easily test variants of the object. By using a parametric virtual model instead the user can ask to make changes of the prototype until he is satisfied.
- **Real-time feedback** – The interactive virtual prototype should react to user's actions in real- time (from the user's perception point of view). The simulation algorithms should be fast enough to grant a real-time feedback. If this is not feasible, simplified algorithms should be used. The simplifications should still give an adequate perception of the object. In that case, the problem moves from the simulation of the ideal physics- based behavior, to the simulation of the faithfully perceived physics- based behavior. In this work, we call such simplified model the "haptic model", and will be discussed in greater detail on

the appropriate section. It is highlighted that the most important aspect of the interactive Virtual Prototype is not the complexity of the simulation of the product, but how the results of the overall simulation is perceived by the humans.

- **Sharable among different users located around the world** – Sometimes testing activities on the same product must be performed in different cultural contexts or geographical regions. Interactive virtual prototypes enable such testing without shipping or reconstructing a physical prototype on these locations.

We describe the methodological approach that we believe enables a company to re-engineer the haptic feedback of a door (in this case a dishwasher door) on the basis of the optimization of the product experience, described on [13]. The method is schematically illustrated in Fig. 2.1. To this aim, virtual prototypes are adopted as means a user can adapt on the basis of his preferences until he gets his favourite experience. In this work, only the haptic feedback is adaptable, while the hearing and visual experiences are maintained fixed, thus the multisensory experience is created. The design of the multisensory product experience through interactive virtual prototypes has some open issues:

- How similar is the virtual prototype to the real product?
- How it can be made adaptable to user preferences and how it is possible to correlate and easily translate those changes into new product specifications?

To address these issues, we identify two stages that come respectively before and after the testing of the multisensory product experience within a virtual environment:

- An initial stage, where a parametric virtual replica of the tactile feedback of product in question is created and adjusted;
- A final stage to transform the desired behaviour of the virtual prototype (obtained by implementing the changes requested by the user)

into the technical/physical specifications required to create the product in the real world.

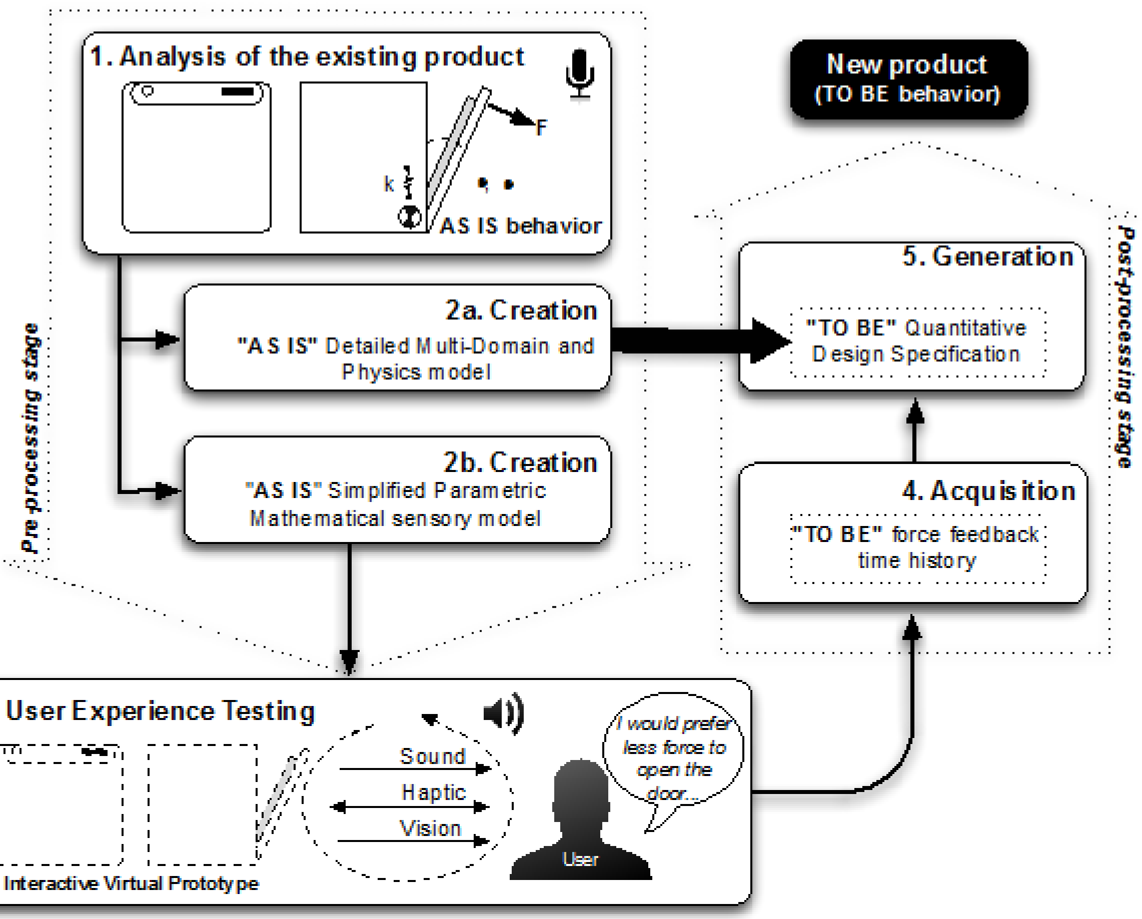
1. Analysis of the existing product
k
F
AS IS behavior
2a. Creation
"AS IS" Detailed Multi-Domain and Physics model
2b. Creation
"AS IS" Simplified Parametric Mathematical sensory model
Pre-processing stage
New product (TO BE behavior)
5. Generation
"TO BE" Quantitative Design Specification
4. Acquisition
"TO BE" force feedback time history
Post-processing stage
3. User Experience Testing
Sound
Haptic
Vision
Interactive Virtual Prototype
User
I would prefer less force to open the door...

FIGURE 2.1: The methodological approach described in this section [13].

The first stage starts with the analysis of the existing product (the dishwasher door) and its components or sub-systems (e.g. the door opening system) that affects the interaction. These components will be re-engineered in order to meet the perception desired by the user. A mathematical model of the door, with its sub-systems components, is derived, and its parameters will be related to the technical specifications necessary to build the system (the physical model). To create such a mathematical model, it is necessary to analyse the mechanism responsible for locking the door, and also the hinge mechanism that controls the door position (e.g. links, spring and damping effects should be analysed). Therefore, the aim is to come out with the list of variables that control and determine the behaviour of the product. The resources necessary to perform this step can vary, depending on the amount of information already available.

The original behaviour of the door ("AS IS") is characterized by

- The sounds that the door emits, in particular, the opening and closing sound, that can be recorded and reproduced by the interactive virtual prototype;
- Its visual appearance, that can be represented by a CAD model and displayed at a computer screen or a projector, for example;
- The force it applies to the user when interacting with it: the force required to unlock/lock and to move along its trajectory. These can be obtained through appropriate measures.

The last point can be addressed by creating another, simplified, mathematical model, whose behaviour is similar to the physical system, but it actually represents perceived physical behavior or the main characteristics of the door (haptic model), not its physical components, which is used to render the force feedback on a haptic interface. Such a step is fundamental in order to come out with an easy-adaptable representation of the system behaviour on which is possible to carry out the simulation in real time in a way that becomes easy to change the behaviour of the virtual prototype.

While it is possible to make the haptic model exactly the same as the physical model, certain problems might emerge:

- If the equations of the physical system are too complex, it becomes difficult to know how each parameter translates into the final behavior of the door, so proper real-time evaluation and testing in a user-friendly manner might be compromised;

- If the physical system was modelled without using explicit equations, such as block diagram environments, it will either have to be rewritten explicitly, or additional tools will have to be used or a computer program will have to be developed to allow interaction between this model and the haptic interface;
- The equations representing the physical system might be too complex to be rendered in real time on the haptic interface. The stiffness of the equations, or the necessity of solving them numerically might create unacceptable delays.

For these reasons, the haptic model on this study is a simplified version of the physical system that aims captures the main perceived sensations, defined by the parameters of the equations.

By measuring the force applied by the user to lock/unlock/move the door simultaneously with an appropriate state of the system, such the speed, acceleration and/or displacement, in function of time, we can say that the dynamical behavior of the original door is described. This behaviour is then used to adjust the parameters that govern the haptic model, in order for its behavior to become as close as possible to the one from original product. The adjustment is made through the use of an optimization algorithm that aims to minimize the difference between the output of the haptic model and the measured states, given the measured force as an input. The same procedure can be applied to tune the mathematical model of the physical system and find the original technical specifications.

The next step is to test the virtual prototype, where its visual appearance, based on a CAD model, can be projected on a screen, and the haptic model is rendered by a haptic interface, allowing user to interact with the virtual prototype. The position of the end-effector of the haptic interface is synchronized with the displacement of the model of the door. Then, for example, the user can try to lock the virtual door and then request a smaller effort, or he can move the door and request to it to be more soft.

Once the desired behaviour has been identified, the final stage of the methodology can start. First, it is necessary to acquire the new (i.e. the "TO

BE") behaviour of the door. This can be done by acquiring the behaviour of the haptic interface, which is measured internally with logs being generated, or registering the values of parameters governing the haptic model. While the most convenient way is to register the values of the parameters governing the haptic model, due to the lack of transparency of the haptic interface, it might not truly represent the perceived effort of the user. The process to obtain the technical specification consists on minimizing the difference between the haptic model response and the physical system model response, given the same input. This time, the parameters governing the physical model will be adjusted to minimize that difference through the use of a optimization algorithm.

Chapter 3

Physical System Modelling: Dishwasher Door

The system that will be used to evaluate the proposed methodology is the interaction with a dishwasher door. A mathematical model of the mechanism that controls the rotation of the door of dishwasher and the locking mechanism for closing is created, where the technical specifications (such as the value of the spring stiffness) are parameters of the mathematical model. Modelling the system enable us to estimate these parameters to accomplish a desired behavior.

3.1 Simulation Software

The dynamic system is developed and solved by the use of the simulation software LMS-AMESim. AMESim is a commercial simulation software for the modelling and analysis of multi-domain systems. The software is a suite of tools used to model, analyze and predict the performance of mechatronics systems. Models are described using time-dependent analytical equations that represent the system's behaviour, which can be any combination of hydraulic, pneumatic, thermal, electric or mechanical systems. To create a simulation model for a system, the set of standard libraries available from the commercial version is used, they contain pre-defined components for different physical domains. The icons in the system have to be connected and for this purpose each icon has ports, which have several inputs and outputs, according to the system they are representing[14]. The process for designing the system is similar to other simulation tools for modelling and analysing multidomain dynamic systems, such as the block diagrams of Scicos/Scilab. AMESim also comes with tools for ease use optimization algorithms such as genetic optimization and non-linear programming by quadratic Lagrangian. The libraries used are Mechanical, Planar Mechanical and Signal, Control, and a brief description of the main component used for the model is given in Fig. 3.1.

3.2 Description of the Dishwasher Door

A dishwasher door provided by Indesit Company is used as the case study. The movement of the door is defined by a planar semi-circular trajectory, restricted to 90º. Depending on how the door and its opening mechanism have been designed, the force required to open it can vary. To open the door, typically, the user has to pull hard enough to unlock a spring-loaded mechanism. The latch mechanism used to lock the door, and specifically the component that clips into the locking mechanisms, is a plastic piece that can be represented as a leaf spring.

Once the door is no longer locked, a mechanism is responsible to maintain it stable, controlling the required force to move it. It consists of an articulated mechanism with a plate connecting all joints, a spring, three friction pieces, one rotating joint and one slotted link that allows vertical translation and rotation 3.2. All movements occur at the same plane.

The door is attached to the front side of the cabinet by means of a hinge placed at the bottom part of the door. The hinge provides an opportune balancing force, generated by the cumulative effects of the spring and of the frictions that interacts with the articulated mechanism, in order to guarantee the stability of the doors during its movement from the vertical to the horizontal position.

The plate has the purpose of transmitting a force, generated by the compression of the spring and the reaction of the frictions, on a desired manner, to the door, and also limiting its course to 90°. It has 4 extremities, denoted as A, B, C and H. The extremity A is connected at the door through a rotating joint. Extremity B is connected to the cabinet by a slotted link, where friction is also present, affecting the vertical displacement of the joint. Extremity C is connected to the spring and another friction. The extremity H has the sole purpose of stopping the door when it is fully open, by colliding

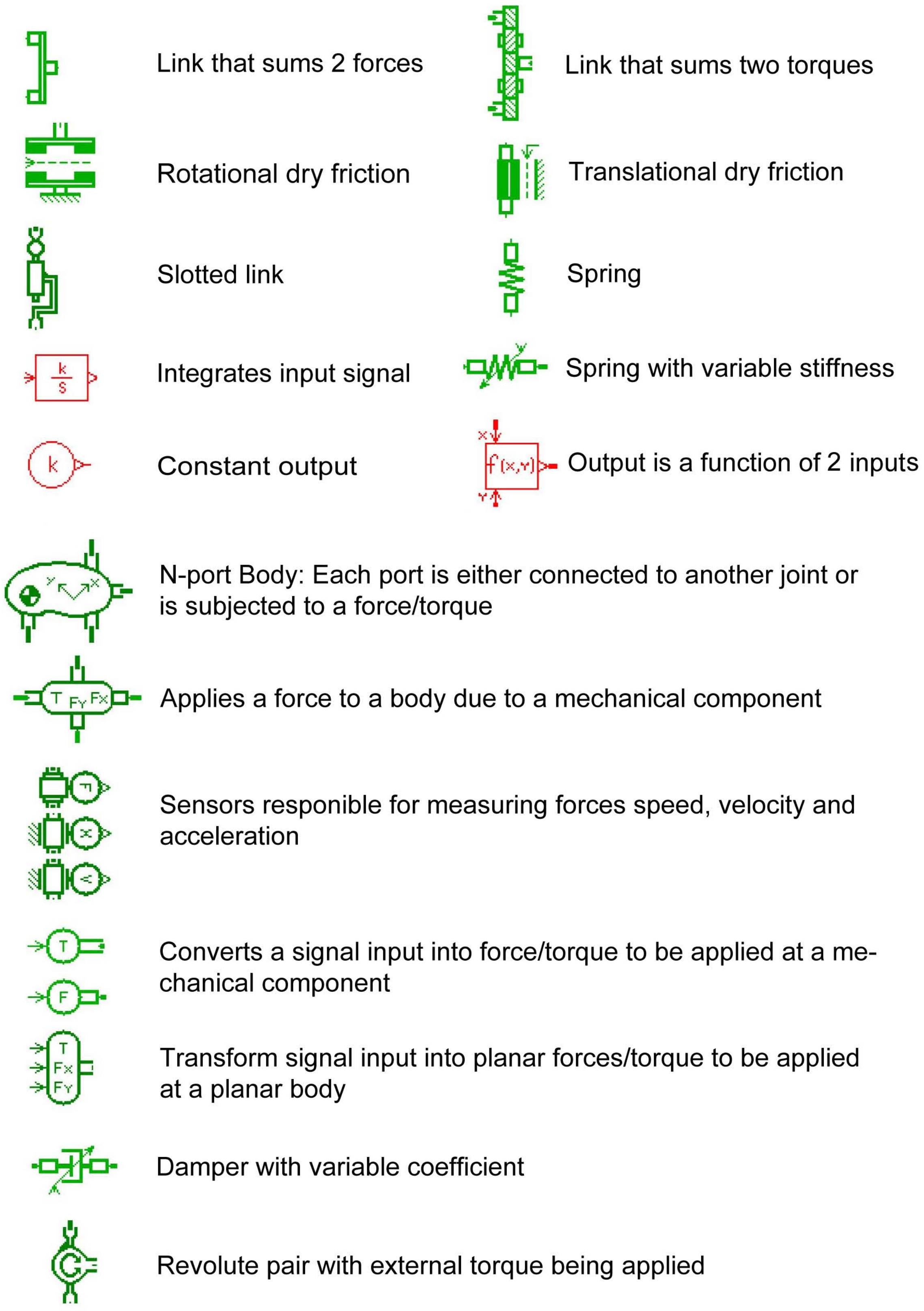
Link that sums 2 forces
Link that sums two torques
Rotational dry friction
Translational dry friction
Slotted link
Spring
k/s
Integrates input signal
Spring with variable stiffness
k
Constant output
f(x,y)
Output is a function of 2 inputs
N-port Body: Each port is either connected to another joint or is subjected to a force/torque
T Fy Fx
Applies a force to a body due to a mechanical component
Sensors responible for measuring forces speed, velocity and acceleration
T
F
Converts a signal input into force/torque to be applied at a mechanical component
T Fx Fy
Transform signal input into planar forces/torque to be applied at a planar body
Damper with variable coefficient
Revolute pair with external torque being applied

FIGURE 3.1: A brief description of the components used to develop the model of the dishwasher door.

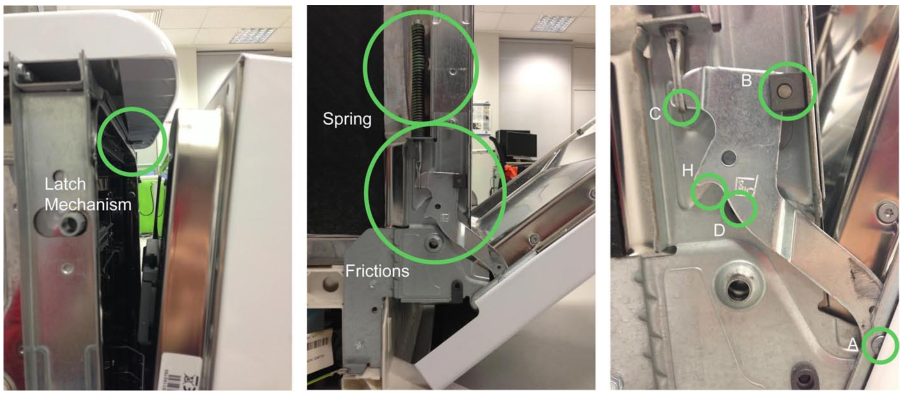

FIGURE 3.2: Mechanical components responsible for controlling the door's motion

with the metal protuberance at the cabinet side. Also, at the cabinet, there is a static friction piece where the plate rubs while moving, at point D (Fig. 3.2).

3.3 Dynamic model in AMESim

3.3.1 Articulated mechanism

Some hypothesis are made to derive the dynamic model of the system without the latch mechanism, that is, only the door with the articulated mechanism:

- all components are infinitely rigid (except the spring);
- the dissipative forces are only caused by dry friction purposefully placed to interact with the articulated mechanism;
- only the door has significant mass and moment of Inertia;
- dry friction is assumed constant, unless the normal force is resulted from the joint reactions, in which case the Coulomb model is adopted;
- the spring has constant stiffness;

- both kinetic and static friction coefficients are the same
- the forces due to friction are caused when there is relative speed between components that are in contact, acting at the geometrical center of area where both surfaces are in contact.

A sketch of the dynamic model without the latch mechanism is displayed at Fig. 3.3. The bar OAE represents the door, the triangle ABC is subject to the forces responsible for controlling the door, which are transmitted via the revolute joint A. The user force is applied at point E. At point B there is a sliding joint with friction, at point C there is friction applying a force on the vertical direction, where the spring is also connected. The effect of the force due to the plate sliding at the fixed friction at point D at the cabinet is transferred to point B, where it is transformed into an external torque and a vertical force.

The model adopted for dry friction is in the shape of a hyperbolic tangent, where both the static and dynamic coefficient are the same. The friction torque/force developed at the contact has the value:

$$F_{fr} = F_{mag}\ tanh(a\ V_{rel}) \tag{3.1}$$

where F_{mag} is the magnitude of the force, a is the sensitivity factor and V_{rel} is the relative speed between the two surfaces, whose shape changes according to the value of a (Fig. 3.4).

The block diagram of the system, resulted from the connection between each subsystem is represented at Fig. 3.5 where each subsystem is connected.

Each of the blocks represent a subsystem of the model: the lock mechanism (LOCK) the door (DOOR), the plate (PLATE), the block that calculates the relative speed between the plate and the static friction at point D, as a function of the vertical speed and rotation of point B (RELATIVESPEED) and the resulting force transmitted at point B due to that force plus the friction already acting at point B (FRICTIONFORCE). At a point of the door, representing the handle, an input force is given by the

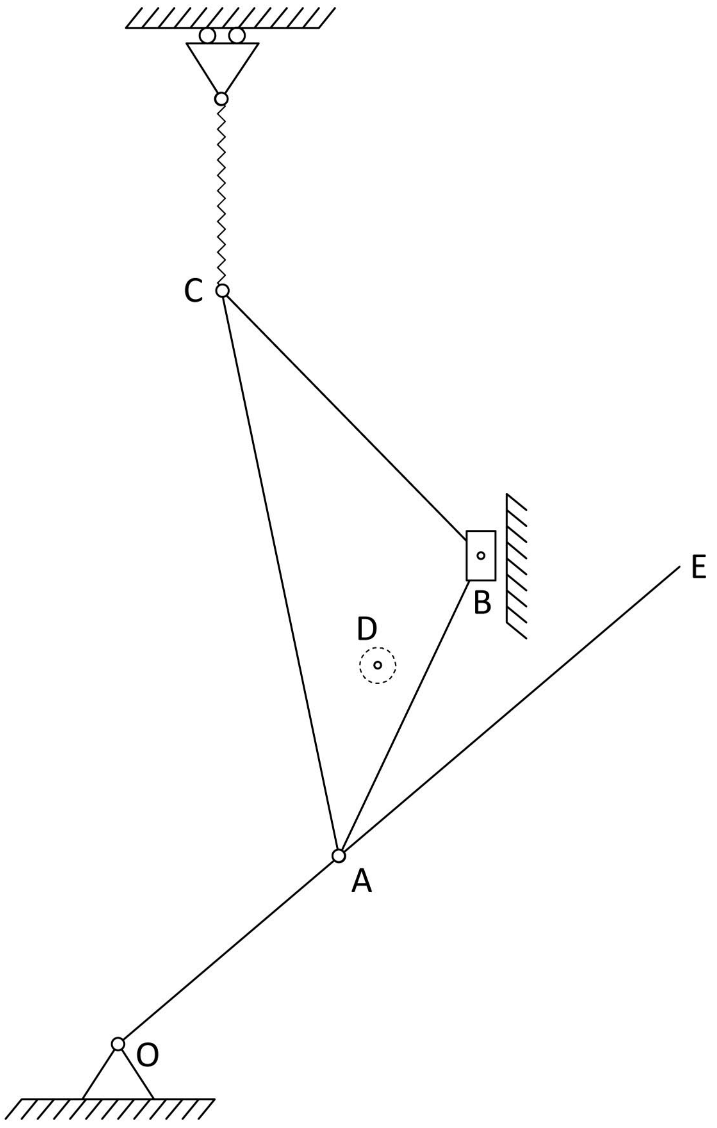

FIGURE 3.3: Sketch of the articulated mechanism

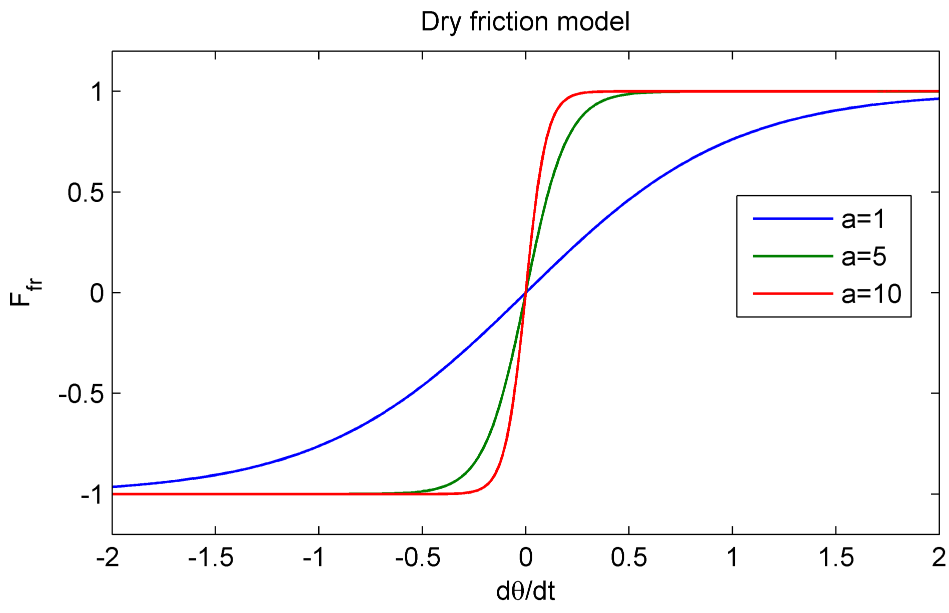

FIGURE 3.4: The higher the value of a, the closer F_{fr} is to the behavior of the ideal dry friction

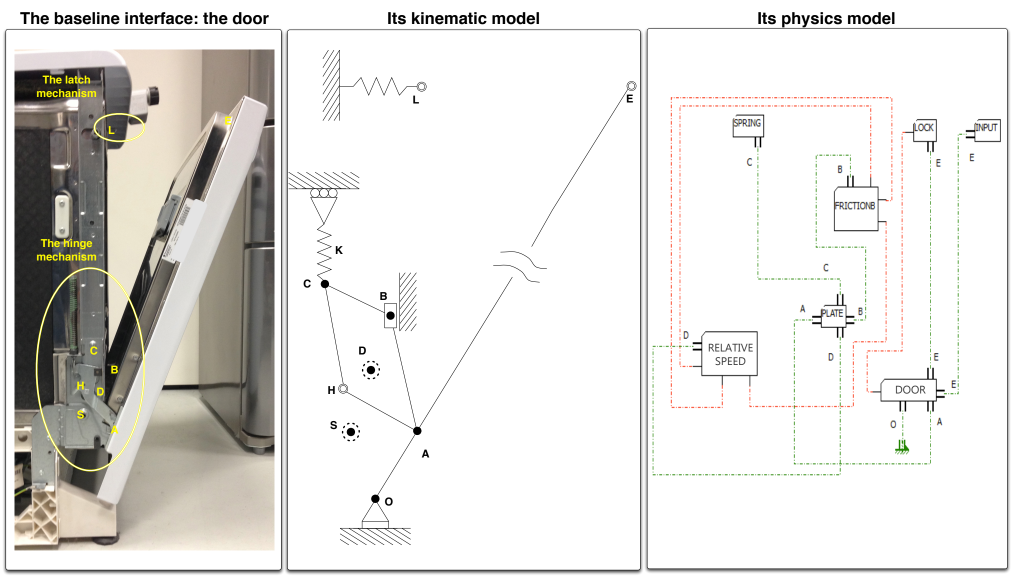

FIGURE 3.5: Block diagram connecting each subsystem, relating to the physical system

user. The green line indicates mechanical linking (such as force application, state constraints), while the red line indicates dimensionless signals (state measures, numerical inputs, etc.) The door is linked to the plate, the friction applied at the plate while it slides is calculated and act as an external force, along with the additional friction already present, at point B. The parameters that define our system are the technical specifications of the product, they include the dimensions and position of each relevant component, the spring stiffness, the force required to lock or unlock the door and the forces generated by each friction. The standard values for the variables occurs when the door is fully opened.

3.3.1.1 Subsystem DOOR

The door is modelled as a rod connected to the ground by a revolute joint. At the revolute joint, a constant dry friction is applied. At point 2, user force is applied; at point 3 the forces from the lock mechanism are applied, and at point 4 the angular position is measured and sent as an output and at point 1 the door is linked with the plate (Fig. 3.6). The important parameters are:

- L – length of the door;
- T_{at_O} – torque acting at joint O, due to friction;
- M – mass of the door;
- I_z – moment of inertia of the door at its barycentre;
- (G_x,G_y) – centre of mass of the door;
- (L_{OA}) – distance between points A and O.

3.3.1.2 Subsystem PLATE

The plate is a body connected at the door (point 7) by a revolute joint at point A, connected to the ground by a sliding joint at point B and to a spring at point C (Fig. 3.7).

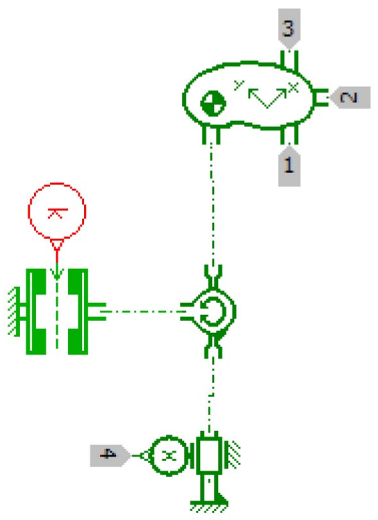

FIGURE 3.6: Block diagram of the subsystem DOOR

Points 1-5 measure the normal force acting on the cabinet the vertical position, the rotation, the vertical speed and the rotational speed, respectively. They are used to calculate the friction at B, which enters as a force at 6.

At C a spring acts, along with friction, using only the vertical displacement and speed of point C. The important parameters are:

- F_{at_C} – force acting at point C due to friction;
- T_{at_A} – torque acting at joint A, due to friction;
- L_{B_x} – horizontal position of point B;
- L_{B_y} – initial vertical position of point B;
- L_{C_x} – initial horizontal position of point C;
- L_{C_y} – initial vertical position of point C;
- K – spring stiffness.
- F_{load} – spring pre-load

3.3.1.3 Subsystem RELATIVESPEED

This subsystem uses the vertical position and rotation measurements of the plate in relation to point B, and calculates the relative speed between the plate and the fixed friction at point D of the cabinet, which is then transformed into a torque and a force. It is denoted y the vertical position of point B in relation to point D, dy/dt its vertical speed, D_x the horizontal

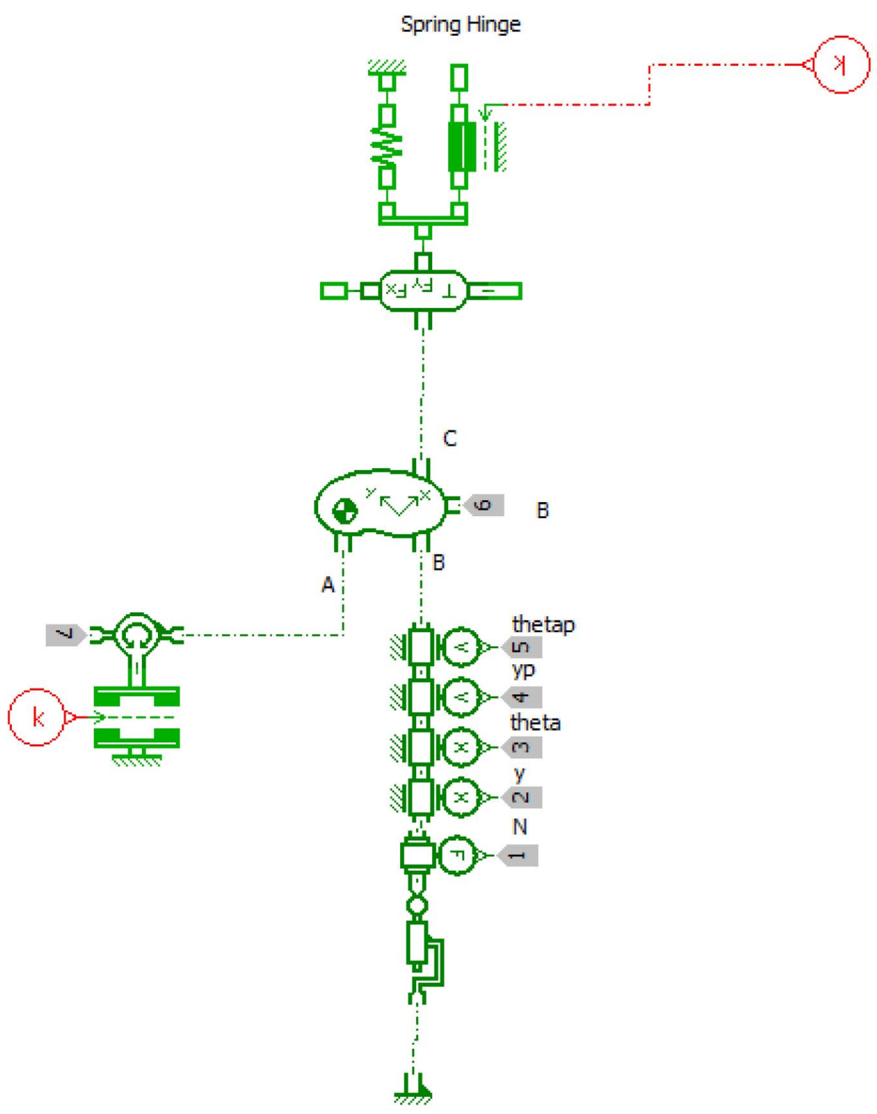

FIGURE 3.7: Block diagram of the subsystem PLATE

position of point B in relation to D (which remains constant), D_y the vertical position of point B in relation to D at the initial state, θ the angular rotation of the body, with an initial value of $atan(D_y, D_x)$ and $d\theta/dt$ the angular rotation and speed of the plate in relation to B. The relative speed V_D between point D and the plate can be thought as the vertical speed of point B plus the distance R between D and B multiplied by the angular speed of the plate. That is:

$$R = \sqrt{y^2 + D_x^2} \tag{3.2}$$

$$V_{D_x} = \frac{d\theta}{dt}\ R\ (-sin(\theta)) \tag{3.3}$$

$$V_{D_y} = \frac{d\theta}{dt}\ R\ cos(\theta) + dy/dt \tag{3.4}$$

The angle formed between V_D and the horizontal direction is

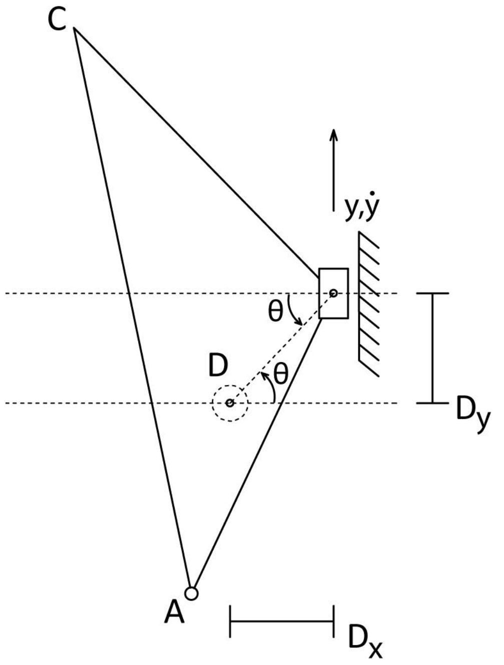

FIGURE 3.8: Diagram used to derive the effects of the fixed friction on point D

$$\alpha_{V_D} = atan2(V_{D_y}, V_{D_x}) \tag{3.5}$$

Once the relative speeds are computed, the forces applied due to the friction at point D can be computed:

$$F_{D_x} = F_{at_D}\ tanh(a\ V_{D_x})\ cos(\alpha_{V_D}) \tag{3.6}$$

$$F_{D_y} = F_{at_D}\ tanh(a\ V_{D_y})\ sin(\alpha_{V_D}) \tag{3.7}$$

These forces can be transferred to point B, where they are transformed into a vertical force and a torque acting at point B. Since point B cannot move horizontally, the force at the horizontal direction is neglected. The resulting torque is:

$$T = D_x\ F_{D_y} + y\ F_{D_x} \tag{3.8}$$

On the diagram (Fig. 3.9), the points 1-6 correspond to the following signals:

- Point 1 – dy/dt
- Point 2 – T
- Point 3 – D_y
- Point 4 – $d\theta/dt$
- Point 5 – y
- Point 6 – θ

The most important parameters on this subsystem are:

- F_{at_D} – Friction force magnitude acting at the fixed point H from the cabinet;
- (D_x, D_y) – The position of point B in relation to D at the initial state;

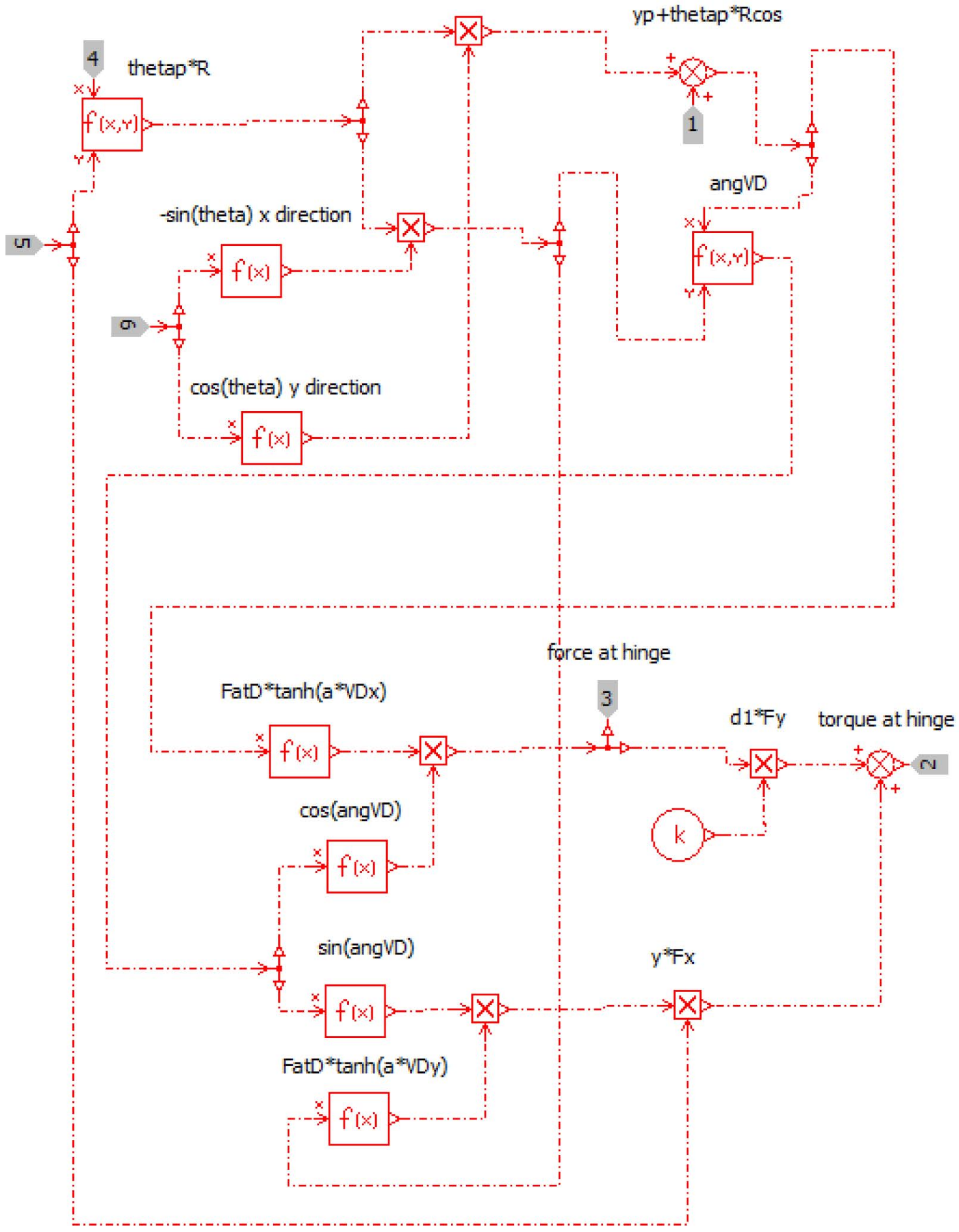

FIGURE 3.9: Block diagram of the subsystem RELATIVESPEED

3.3.1.4 Subsystem FRICTIONFORCE

This subsystem (3.10) transform the signals of the subsystem RELATIVESPEED into a mechanical force and sums it with the friction already present at point B.

At point 1 enters the force signal and at 4 enters the torque signal, respectively, from the subsystem RELATIVESPEED. At point 2 enters the normal force applied by the slotted link. The torques and forces are summed, and then applied at the plate at the output 3.

The important parameters on this subsystem are:

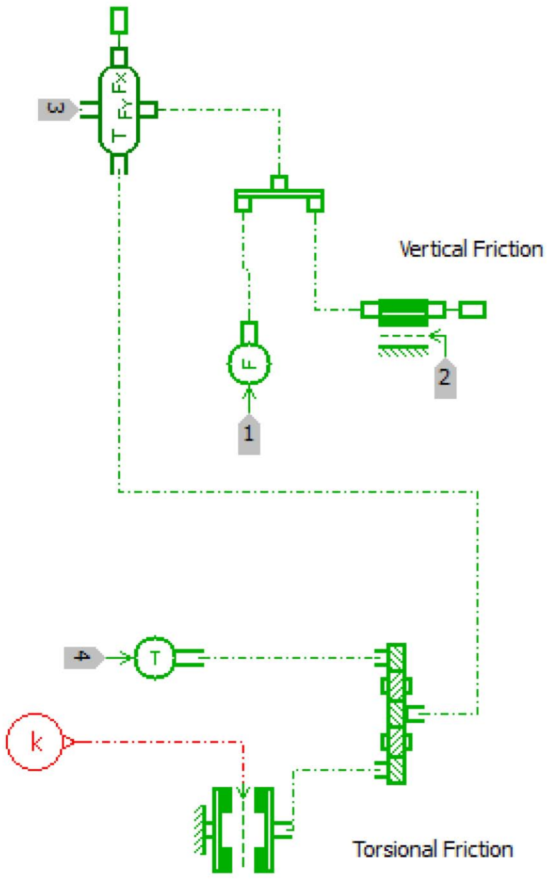

FIGURE 3.10: Block diagram of the subsystem FRICTIONFORCE

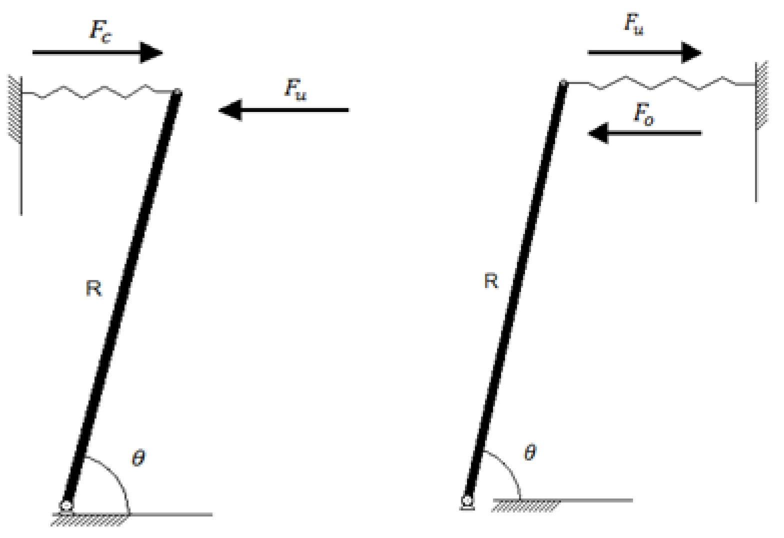

FIGURE 3.11: System representation of the latch mechanism: the stiffness of the spring depends whether the system is being opened or closed

- μ – coefficient for the force of the vertical friction;
- T_{at_B} – force acting at the slotted link, due to friction;

3.3.2 Latch mechanism

The latch mechanism resists to the force the user applies both when the user is trying to open and when trying to close. The deformation cannot be clearly felt by the user; instead, once a force threshold is crossed, the door opens (or closes). It can be thought as a very stiff spring that goes under compression, both during opening and closing. Therefore, we have a high stiffness spring/damper system where the force changes direction depending on whether the user is opening or closing the door.

The model is composed by 3 springs that are active only inside a limited interval: one for the opening phase, another for the closing phase, and another to represent the wall. The wall also has a damping term to dissipate the kinetic energy once it is touched. The wall is active for $\theta < 0$, while the opening and closing phase are restricted at a custom range.

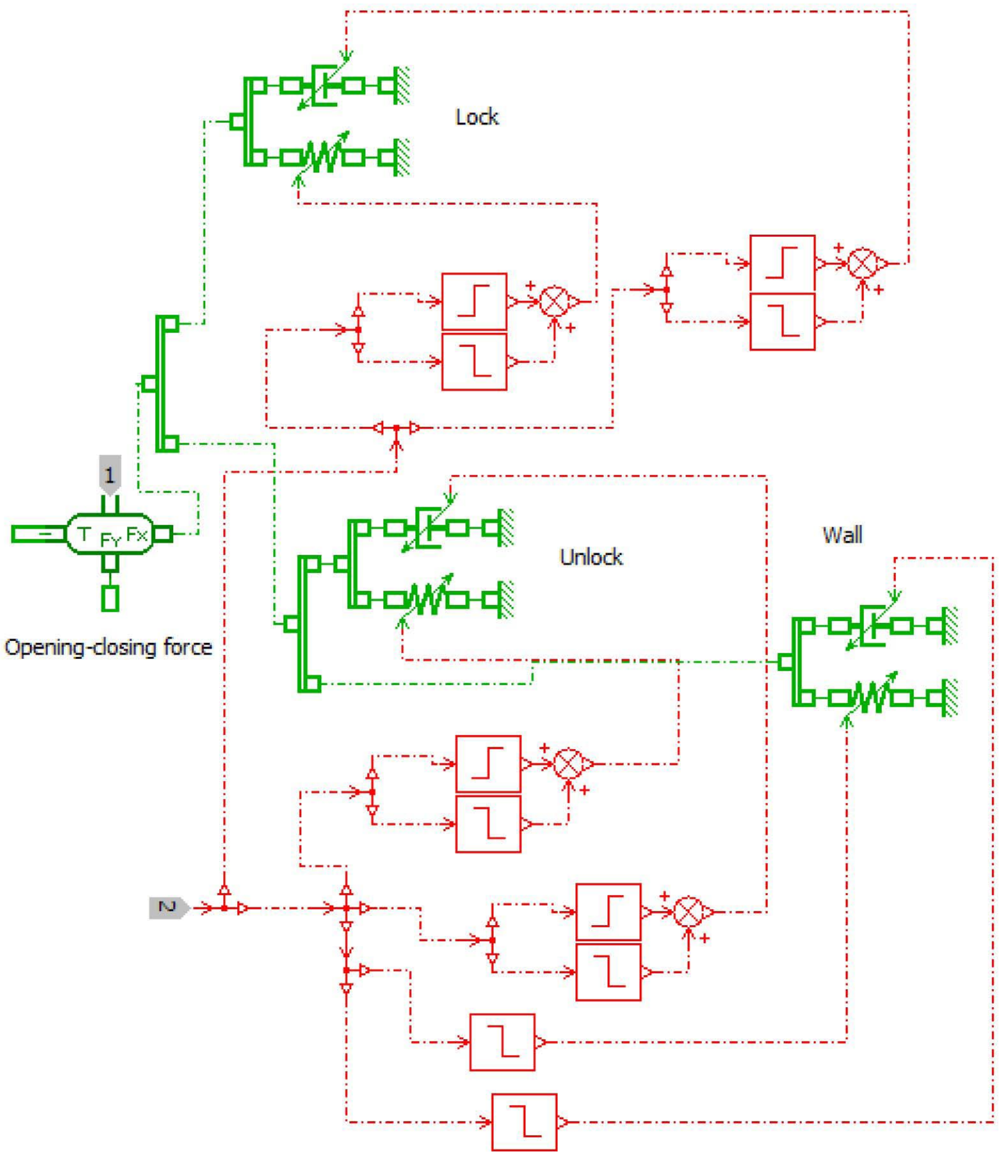

FIGURE 3.12: Block diagram for the latch mechanism

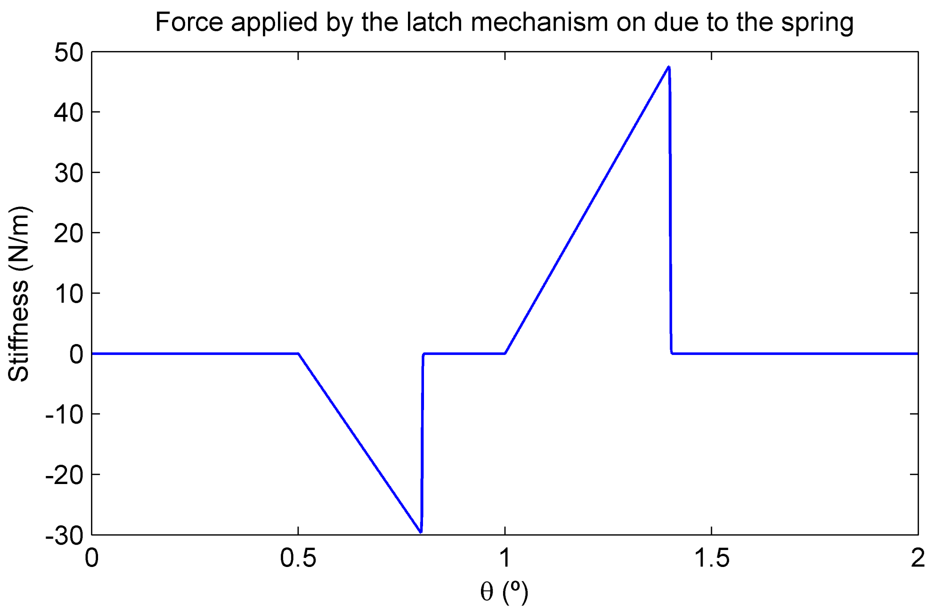

FIGURE 3.13: The force applied by the latch mechanism in function of the position of the door due to the spring. The force changes direction depending whether the user is trying to lock or unlock the door

Looking at the block diagram (Fig. 3.12, at point 1 the force is applied at the door, while at point 2 the position of the door is used to define the values for the stiffness and damping values. In Fig. 3.13 and Fig. 3.14, the graphics illustrate how the spring stiffness changes according to the position and the force applied by the latch mechanism in function of the position.

3.4 Measurements

The estimation of the relevant parameters of the dynamic model developed requires quantitative analysis of the dimensions and behavior of the real system. The relevant distances were obtained by means of direct measurement while the remaining parameters were estimated by an optimization process that is explained on Chapter 5. The development of the virtual replica of the product also takes into consideration the qualitative behavior of the real product; simplifications and assumptions can be made

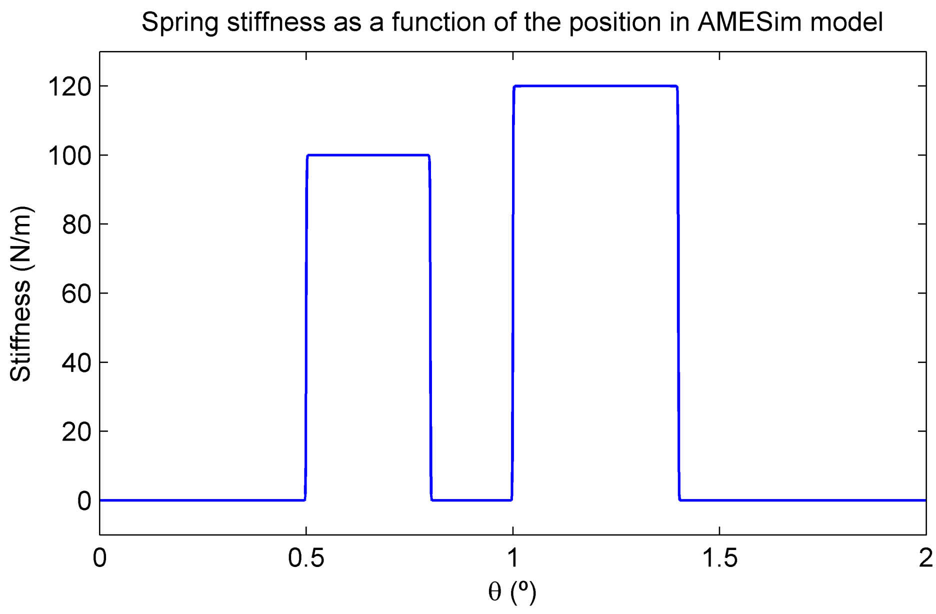

FIGURE 3.14: The stiffness of the latch mechanism changes according to the position of the door, they are related to the force required to lock or unlock the door

based on the measurements. If it is considered that the our dynamic system has the following form:

$$\begin{aligned} \dot{x}(t) &= f(x(t), p) + g(x(t), p)\ F_u(t) \\ x(0) &= x_0 \end{aligned} \tag{3.9}$$

where x is the state of the system, x_0 is the initial condition, $p = \{p_1, p_2, ..., p_i\}$ is the vector that contains the parameters to be estimated and $F_u(t)$ the input given by the user, the solution of the system can be written on the following form:

$$x(t) = h(p, F_u(t), x_0, t) \tag{3.10}$$

If we can measure $x(t)$ and $F_u(t)$, knowing the initial condition x_0, we should be able to find a set p that satisfies Eq. 3.9 as $x(t)$, $F_u(t)$ and the structure of f and g are known. Therefore, to estimate p, we measured and estimated the state and the input $F_u(t)$. It is important to note, however, that there might be more than one set p that satisfies the imposed condition.

Therefore, while the set p found might not be equivalent to the real values that characterize the system, it is a set that generates an equivalent system, and satisfies the intended purpose.

The state $x(t)$ is composed by the angular speed ($\dot{\theta}$) and angular position (θ). The entire state $x(t)$ does not need to be fully measured, since:

$$\theta(t) = \int \dot{\theta}(t)dt \tag{3.11}$$

The measurement of the force applied by the user has been performed using a compression donut load cell (FUTEK model LTH300, www.futek.com) with a maximum detection load of $445N$. The load cell has been mounted between participants' hand and the door handle (as illustrated in Fig. 3.15) in order to measure the required opening force as a function of time. The estimation of the force required to lock and unlock the door can be readily obtained from the measured force: since no noticeable movement is perceived, measuring the state becomes unnecessary. However, when the door is locked, there is a range where some movement is allowed; it was measured through an inclinometer (Columbia Research Laboratories model SI-701B), with range of $\pm 10°$. The estimation of the state of the system was performed through measuring the velocity in function of time with a gyroscope (British Aerospace Systems and Equipment unipolar gyroscope, able to detect angular velocities in the range of $\pm 100°/s$), simultaneously measured with the force applied by the user. The signals have been acquired through the National Instruments NI cDAQ-9172 and NI 9125 analog input modules (www.ni.com) and processed through LabVIEW SignalExpress. All the signals have been acquired with a frequency of $1024Hz$. Three users have been asked to open the door by applying different forces with the sensors configuration showed in Fig. 3.15. Subsequently the load cell has been glued on the external part of the door and the same users have been asked to close the door by applying different forces.

The sensors have been coupled in the following ways: first the load cell plus the inclinometer were used to detect what happened before the door is opened (the zero corresponding to the door closed) and then the

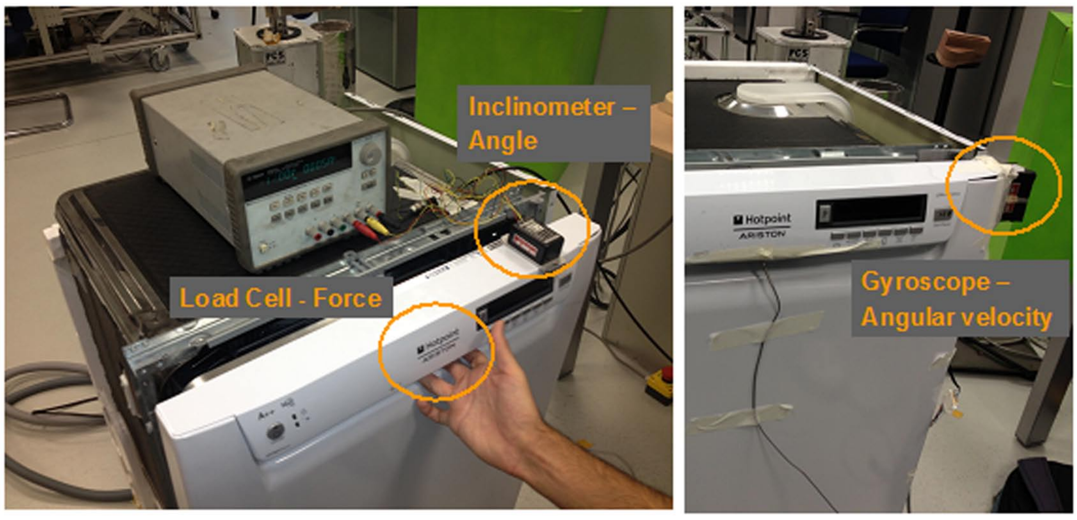

FIGURE 3.15: Location of sensors for the measurements performed on the commercial dishwasher [13].

load cell plus the gyroscope to understand the dynamic behaviour of the door. As it can be seen from Fig. 3.16 and Fig. 3.17, the is no detectable movement after 0.5°, when the door is locked, until a threshold of force is crossed; the force required to unlock can be considered simply the peak of the graphic.

The behavior when locking the door is identical; the peak is taken as the force required to lock the door, the dynamic behavior is neglected.

When the door was fully open, an impulse is given and the angular speed is measured by the gyroscope. By looking at the graphics (Fig. 3.19 and 3.20), that correspond to one trial, measured by the gyroscope and load cell, it is possible to observe that the angular speed decreases at approximately a constant rate once a force is no longer applied, thus it can be inferred that the main factor responsible for the deceleration comes from a constant dry friction.

It is noted that since the forces are by humans, the input of the system will always be significantly different for every trial and as a consequence, the output of the system will also be different for every trial. While the pair input/output are changes every trial, both in magnitude and shape, they are

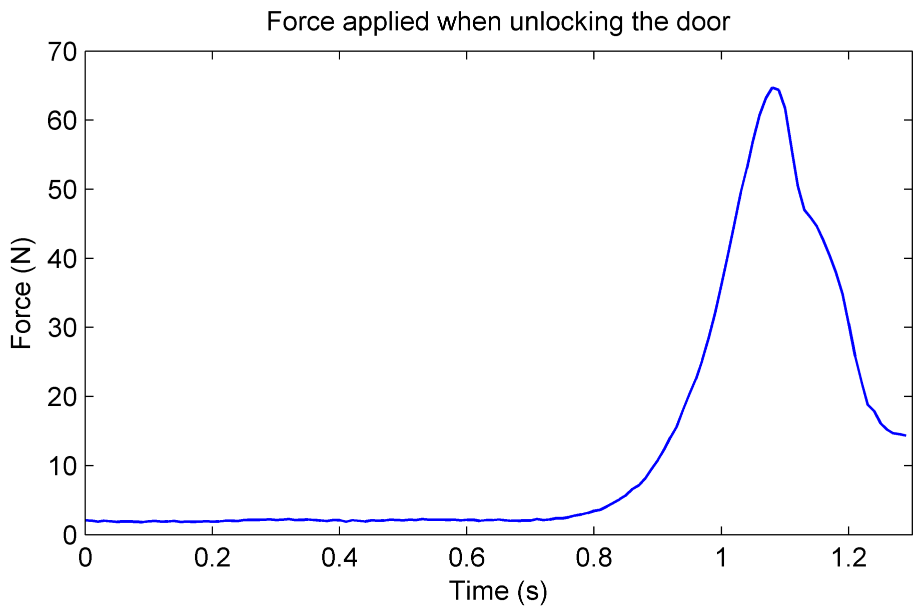

FIGURE 3.16: Measured force when unlocking the door in function of time: the force required to unlock the door is characterized by the peak of the graphic

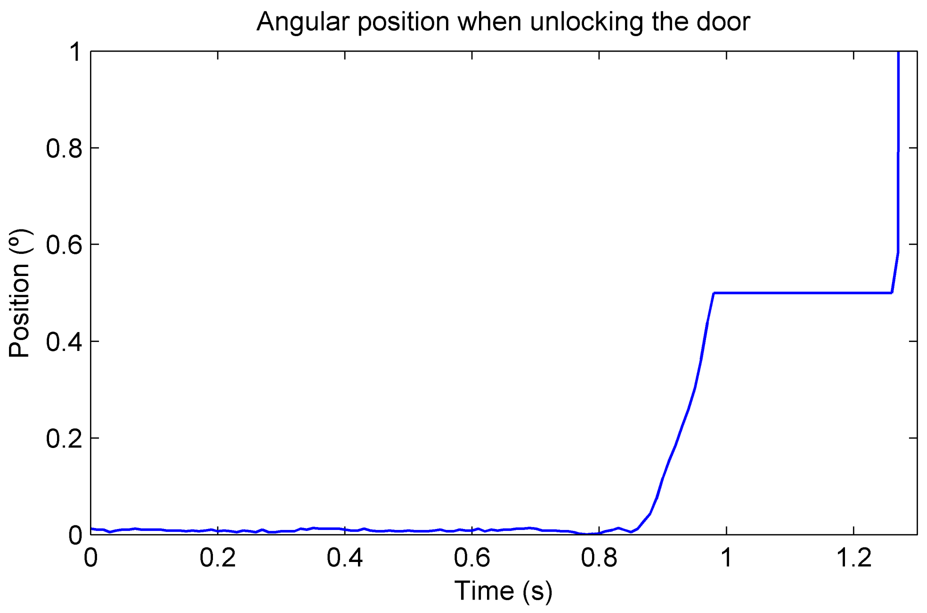

FIGURE 3.17: Measured position when unlocking the door in function of time: no noticeable displacement until the force cross a threshold

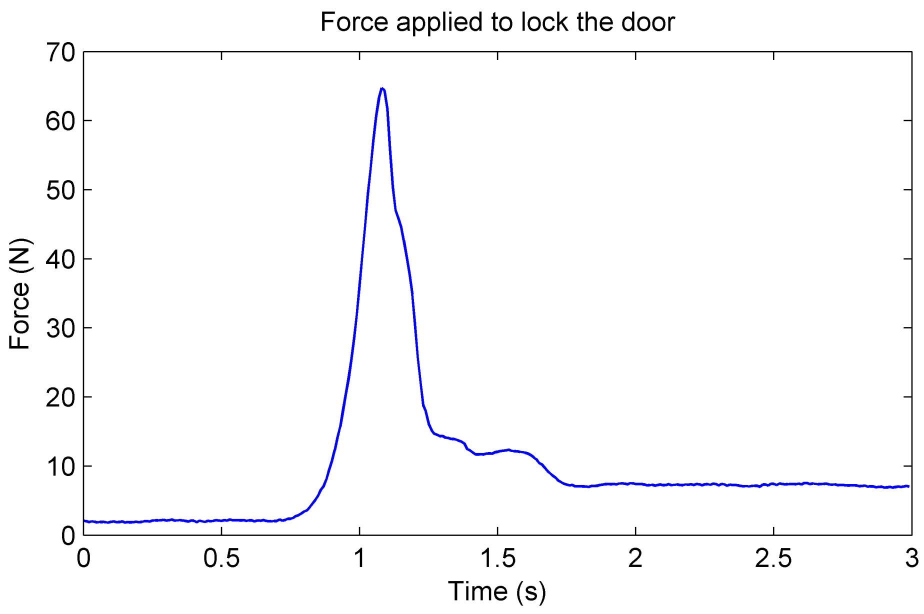

Figure 3.18: Measured force when locking the door in function of time: the peak of the graphic characterizes the force required to lock the door

still measuring the response of the same system, and so it is expected that the estimation of the parameters of the system does change significantly when done for each pair. The estimation is discussed in Chapter 5.

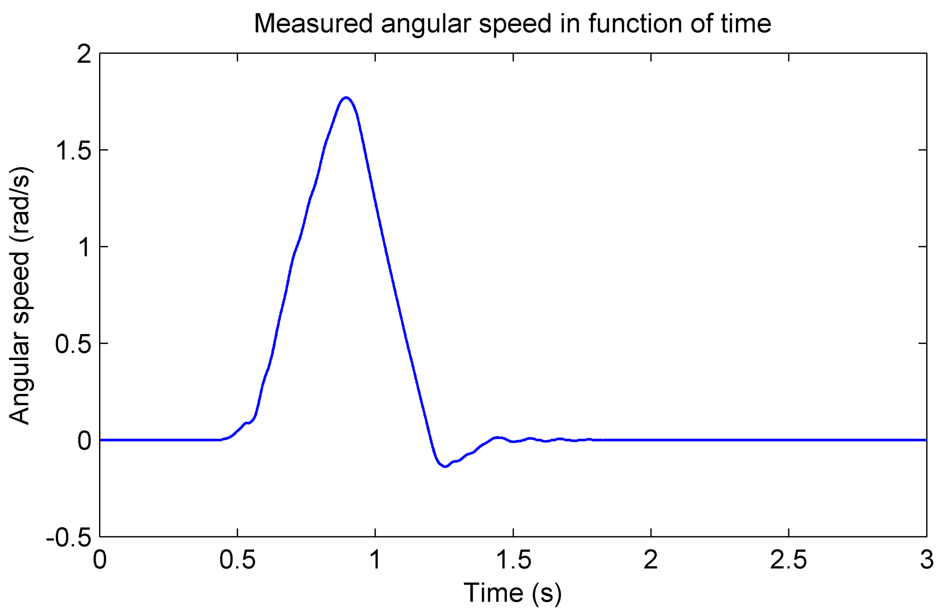

FIGURE 3.19: The measured angular speed in function of time shows that the angular speed decreases linearly, which indicates dry friction is the predominant dissipative force

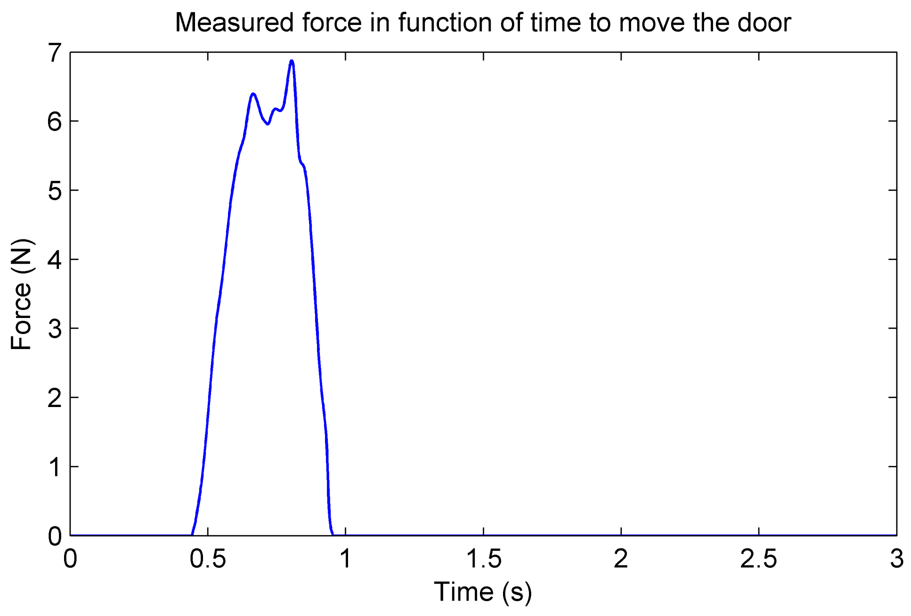

FIGURE 3.20: Measured input force in function of time: an impulse is given when the door is fully open

Chapter 4

Haptic rendering

In this chapter, it is described the process of creating the haptic virtual prototype, detailing the device used, the adopted programming language and the strategies to mimic the tactile sensations derived from the dishwasher door.

4.1 Haptic Device

There are plenty of haptic interfaces commercially available today. Commercially, haptic devices come in two distinct classes impedance controlled devices, and admittance controlled devices [15]. A well-known example of impedance control devices are the Phantom devices from Sensable. The paradigm on impedance control is this: the user moves the haptic device, and the device will react with a force when the virtual object is met. From the haptic device point of view, displacement becomes the input, and the force is the output. The user will feel the mass and friction of the actual device, though they can be made reduced through mechanical design. Impedance controlled devices tend to be naturally light and highly backdrivable, typically cable driven by DC motors. In contrast with impedance control, in admittance control the paradigm is this: the user exerts a force on the haptic device, and the device will react with the proper displacement. That is, force applied by the user becomes the input, while the displacement is the output. Admittance control allows backlash and tip inertia to be eliminated, giving considerable freedom on the design of the device. This results in increased robustness, where high stiffness and high forces are possible. The device used for this work is based on the admittance control paradigm, the MOOG's HapticMaster. While impedance control device tends to be lightweight, backlash free, but generally able to render low masses. Their performance is lacking if high stiffness or higher forces are required to be rendered. Admittance control devices, in contrast, are more suitable for

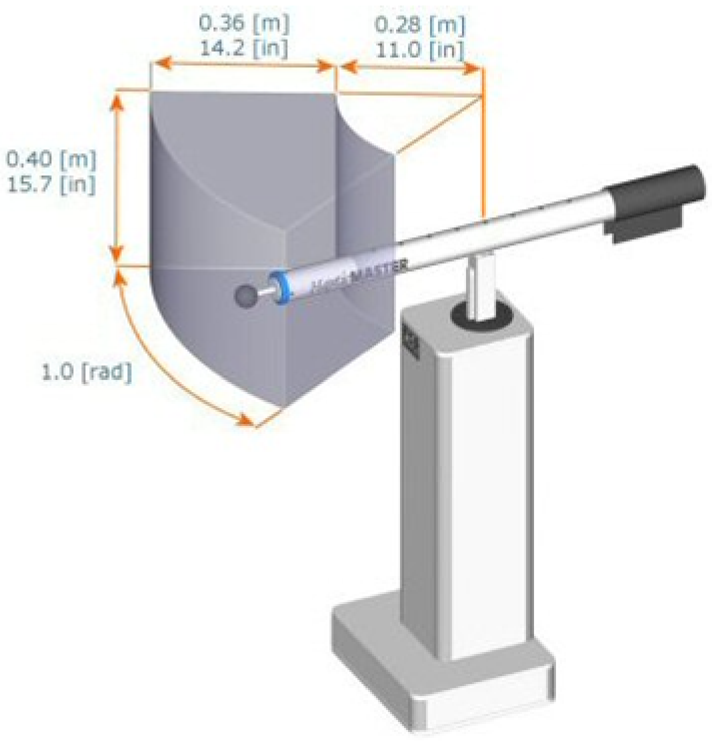

FIGURE 4.1: The workspace of the HapticMaster [16]

higher forces and stiffness, though they are not well-suitable for rendering low mass. The choice of using an admittance control haptic device comes due to the fact that our case study requires both high stiffness and higher forces . The HapticMaster measures the force exerted by the user, and an internal model calculates the position, velocity and acceleration which a virtual object touched in space would get as a result of this force [16]. This vector is commanded to the robot, which makes the movement by means of a conventional control law, rendered at $2500Hz$, and can apply a forces up to $200N$. A residual mass will always be present. The workspace of the HapticMaster is depicted in Fig. 4.1. It allows one rotation around its base and two translations, resulting in 3 degrees of freedom for the end effector, spanning a volumetric workspace. The HapticMaster allows the end effector to be changed for another, allowing us to mount the handle of the door at the end of the robot arm [16].

Some important considerations must be mentioned. One of the main problems encountered when using this haptic device in conjunction with H3D API was related to its stability when higher stiffness was required to be rendered with explicit stiff functions. Even though the haptic device used for the application was designed for high stiffness, it still wasn't able to properly render the forces when the user tried to lock/unlock the virtual door (when using explicit functions): the device became highly unstable once a threshold of stiffness was crossed, fairly inferior to the required one to properly reproduce the act of unlocking, and specially, locking the door. Additionally, there was a limit on the stiffness used to constrain the end effector to follow a specific trajectory the way we modelled. While it was not as stiff as a real dishwasher door, it still allowed to reproduce the movement with an acceptable fidelity. Regarding the smoothness, we detected that it did not translate so well when we tried to render dry friction as an hyperbolic tangent. Not only some background vibration could be sensed when moving the device around, but higher friction values (above 10N) would render the device highly unstable. Another limitation was related to the size of the workspace: it was not large enough, therefore the door had to be scaled down until it could fit entirely on the workspace. Unfortunately, none of the devices available had a workspace large enough. Even though these issues were present, and the virtual dishwasher door behavior did not fully equate with the real dishwasher door, the concept proposed on this study was successfully validated, and a better result would be obtained with a haptic device developed for this purpose.

4.2 Coding language and API

In order to control the haptic manipulation and then propose to the user different behaviors, the scripting language Python combined with H3DAPI [17] has been used for developing the interaction model and to retrieve the values necessary to quantitatively describe the interaction phase in terms of displacement, acceleration and velocity. The H3D API is used as an interface to communicate the instructions from the computer to the haptic device, while Python scripts are responsible for calculating the force applied to the user by the end-effector. For contextualization, a brief description of H3D API and Python are given.

4.2.1 H3D API

SenseGraphics H3D API is an open source software development platform for multi-sensory applications, using the open standards X3D, OpenGL and SenseGraphics haptics in a unified scenegraph taking, where both the haptic and graphic rendering are taken care of [17]. It considerably accelerates the development time, and can use Python scripting or C++ programming code for additional flexibility. It was chosen because not only it is compatible with HapticMaster, but also because it combines haptic-graphics into a single plataform.

4.2.2 Python

Python is a powerful dynamic, general-purpose, interpreted high-level programming language that is used in a wide variety of application domains [18]. Its design philosophy emphasizes code readability and allows the expression of concepts in fewer lines of code than other lower level programming languages, such as C. It supports multiple programming paradigms, including object-oriented, imperative and functional programming styles.

4.2.3 X3D

X3D is a open standards file format and run-time architecture to represent and communicate 3D scenes and objects using XML. It provides a system for the storage, retrieval and playback of real time graphics content embedded in applications [19].

4.3 Haptic Model

The haptic model is the mathematical model responsible for computing the forces that the end-effector is subjected, the appropriate model should effectively mimic the desired behaviour. The equations used have the form:

$$\begin{aligned} X &= [x, y, z] \\ F_h &= (F_{hx}, F_{hy}, F_{hz}) = F(X, \dot{X}) \end{aligned} \tag{4.1}$$

where F_h is the vector representing the forces that are applied to the end-effector, and X is the state of the system: the spatial coordinates of the end-effector and $\dot{X}$ their derivatives.

In order to correspond the haptic model with the act of opening the door, the following points are addressed:

1. Describe the position of the door both in Cartesian coordinates and polar coordinates relating it with the description of internal coordinates of the end-effector by H3D API;
2. Constrain the movement of the end-effector to the same trajectory allowed by the door;
3. Evaluate what are the main effects influencing the tactile haptic sensation and describe them into equations.

The first point is addressed simply by evaluating the Cartesian coordinates displayed when the tracker position is requested, and then making an appropriate translation. The second point is addressed by defining the trajectory as an intersection between two surfaces, one in which the boundaries have a circular shape with the same radius defined by the movement of the door (e.g. a sphere or a cylinder), and the other defined by a plane that contains the trajectory. The intersection constrains the end-effector by applying an attracting force proportional to a specified distance between the end-effector and the surfaces. As the proportional force tends to create undesired vibrations, damping was also applied.

In the third point, it should be noted that the main effects will be tuned in real-time. Therefore it is important to describe them in an "intuitive" way: functions should represent sensations, not the dynamic parameters from the real mechanism. Instead of using functions where the parameters correspond to a distance between two pivots or the specific friction occurring on one of the sliding components, the effect perceived by the user should be represented, i.e. the parameters should correspond to the "easiness" or "smoothness" of moving the door. This way, the operator can more easily understand how modifying a value affects his perception, e.g. to affect the "smoothness", the operator modifies the global friction, instead of the distance between two pivots. To do so, the equations responsible for calculating these forces were described directly in function of the angular position, and the parameters are changed to alter the magnitude of these functions.

4.3.1 Transformation of Coordinates

Since the trajectory traversed by the door is a semi-circumference, it is convenient to describe the position of the end-effector in polar coordinates. Since the force applied to the haptic device uses the state of the end-effector in Cartesian coordinates, when programmed in x3d, and the forces used in the model are described in cylindrical coordinates, a conversion becomes convenient.

The internal coordinate system of H3D, that we call O_i, describing the origin of the end-effector, is located at the geometrical center of the workspace (Fig. 4.1). To avoid the use of an angular coordinate outside the range of $[0, \pi/2]$, the origin is translated one of the extremities of the workspace, from $(0, 0, 0)_i$ to $(0, -0.158, -0.138)_i$, which becomes the origin of the new coordinate system, O_m.

Denoting $(pos_x, pos_y, pos_z)_i$ the absolute position coordinates of the device on O_i the internal coordinate system, read by the software, the coordinates $(x, y, z)_m$ (whose index will from now on be suppressed) used for the equations are:

$$\begin{aligned} x &= pos_x \\ y &= 0.158 + pos_y \\ z &= 0.138 + pos_z \end{aligned} \tag{4.2}$$

The linear speed is thus:

$$\begin{aligned}
\frac{dx}{dt} &= \frac{dpos_x}{dt} \\
\frac{dy}{dt} &= \frac{dpos_y}{dt} \\
\frac{dz}{dt} &= \frac{dpos_z}{dt}
\end{aligned} \tag{4.3}$$

The force vectors are described as (Fx, Fy, Fz) on Cartesian coordinates.

The equivalent coordinates and their time derivatives on the cylindrical representation (x, r, θ) are:

$$\begin{aligned}
r &= \sqrt{y^2 + z^2} \\
\frac{dr}{dt} &= \frac{y\ \frac{dy}{dt} + z\ \frac{dz}{dt}}{\sqrt{y^2 + z^2}}
\end{aligned} \tag{4.4}$$

$$\begin{aligned}
\theta &= atan2(y, z) \\
\frac{d\theta}{dt} &= \frac{\frac{dy}{dt}\ z - \frac{dz}{dt}\ y}{y^2 + z^2}
\end{aligned} \tag{4.5}$$

On vectors, a rotation matrix operation is applied, being the matrix of rotation:

$$Rot(\theta) = \begin{pmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{pmatrix} \tag{4.6}$$

The radial force F_r and the tangential force F_θ are:

$$[F_r, F_\theta] = Rot(\theta)\ [F_z, F_y] \tag{4.7}$$

Conversely:

$$[F_z, F_y] = Rot(\theta)^T\ [F_r, F_\theta] \tag{4.8}$$

Note that the indexes y and z are inverted due to the way θ has been defined and due to how the rotation matrix has been assembled. The door is fully open when $\theta = 0$ and fully closed when $\theta = \pi/2$.

4.3.2 Constraining the end-effector

Here it is described generalized methods capable of describing the forces necessary to be applied the end-effector to restrict its movement to a given trajectory or surface. Two methods are developed: the first one for 2D curves described on polar coordinates, the second one for 3D continuous surfaces with continuous first derivatives. Both methods are based on the same idea: the end-effector is attached to a very stiff virtual spring-damper couple which equilibrium point is any point of the surface/curve. So, whenever the end-effector position is outside the allowed region, a restitution force will be applied to bring it back.

4.3.2.1 First method: Radial direction stiffness

The first method, applied to 2D curves described by polar coordinates, consists on defining a spring on the radial direction, whose equilibrium point is a function of the angular position (Fig. 4.2). We define:

- $P(x, y)$ a point representing the position of the end effector (on Cartesian coordinates, though not necessarily the ones mentioned before);
- $R(\theta)$ the function representing the curve, in polar coordinates, where the origin coincides with the Cartesian coordinate system;
- $Q(r, \theta)$ the closest point to P on the radial direction (on polar coordinates);
- k the spring stiffness;
- c the damping value.

We then have:

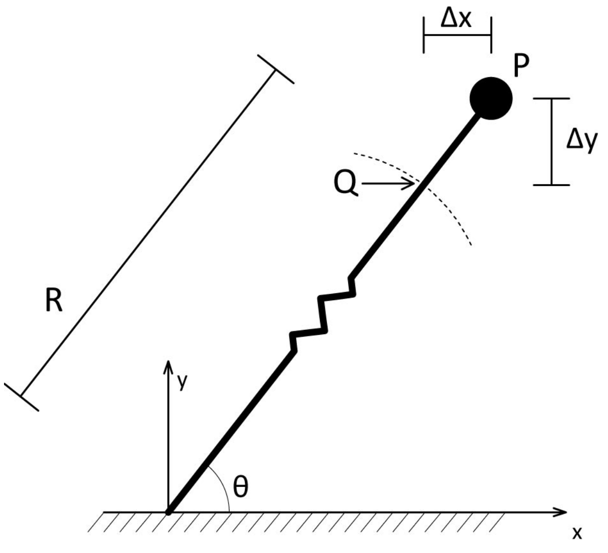

FIGURE 4.2: Model for the spring acting on the radial direction of a curve: the force will always be oriented in the radial direction, though the equilibrium point can change according to the curve we want the end effector to follow

- the angle that aligns P with Q and Q with the origin of the system:

$$\theta = atan2(y/x); \tag{4.9}$$

- the distance on x direction between P and Q:

$$\Delta x = x - r\ cos(\theta); \tag{4.10}$$

- the distance on y direction between P and Q:

$$\Delta y = y - r\ sin(\theta); \tag{4.11}$$

Therefore, the forces that bring the end-effector from point P to Q are

$$\begin{aligned} F_x &= -k\ \Delta x - c\ d\Delta x/dt \\ F_y &= -k\ \Delta y - c\ d\Delta y/dt \end{aligned} \tag{4.12}$$

If end-effector possess has a virtual mass m, and the user applies a force $F_u = [F_{ux}, F_{uy}]$, the differential equations that represents the movement of the end-effector are:

$$\begin{aligned} m\ddot{x} &= F_x + F_{ux} \\ m\ddot{y} &= F_y + F_{uy} \end{aligned} \tag{4.13}$$

The following example illustrates the dynamic behaviour of the end-effector numerically.

Example: Defining the trajectory as an ellipse on polar coordinates:

$$R(\theta) = \frac{ab}{(\sqrt{a^2\ cos(\theta)^2 + b^2\ sin(\theta)^2})} \tag{4.14}$$

Where:

$$\begin{aligned} a &= 0.2; \\ b &= 0.1; \\ c &= 20; \\ k &= 200. \end{aligned} \tag{4.15}$$

The end effector is set to have unitary mass and is set to begin at point $(0.3, 0.3, 0)$. Since the initial point does not belong to the trajectory, the end effector is brought to it by the forces (F_x, F_y) (Fig. 4.3). At $t = 3s$ a vertical force is applied at the end effector by the user (Fig. 4.4).

It can be seen that the trajectory is followed satisfactorily, and the absolute force applied by the haptic interface is acceptable (Fig. 4.5).

4.3.2.2 Second method: Restitution force based on distance minimization

The second method consists on finding the closest point $Q(a, b, c)$ of a surface to $P(x, y, z)$, and applying the a force at the direction $Q - P$

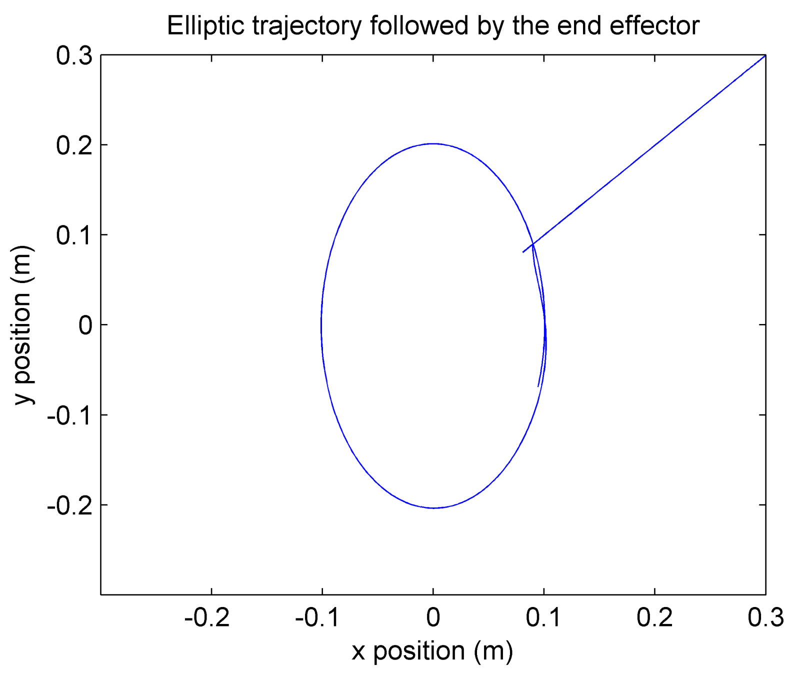

FIGURE 4.3: The end effector is brought to the trajectory from an initial position that does not belong to the ellipse

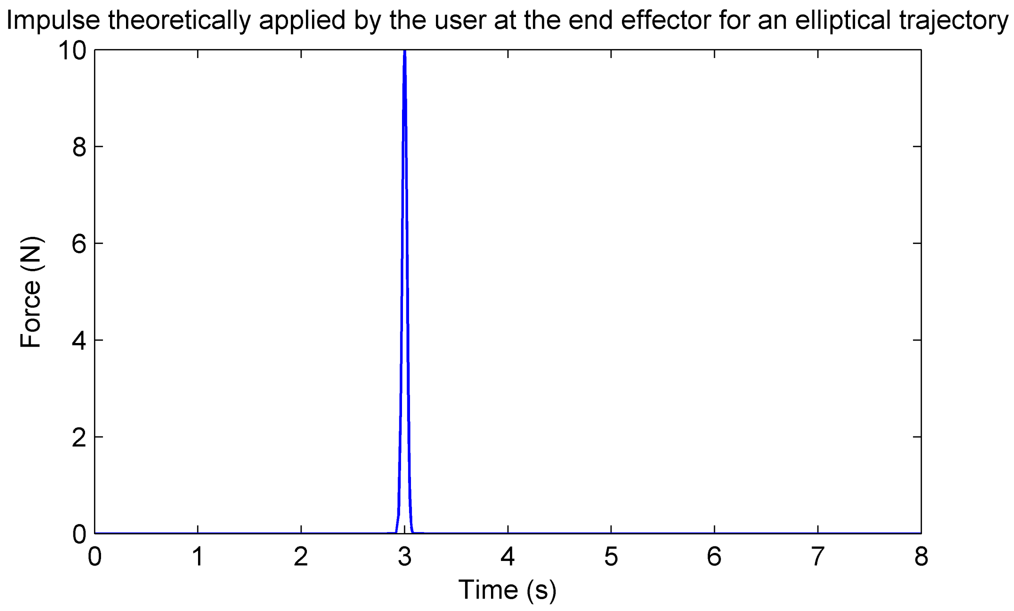

FIGURE 4.4: Impulse theoretically applied by the user at the end effector for the elliptical trajectory

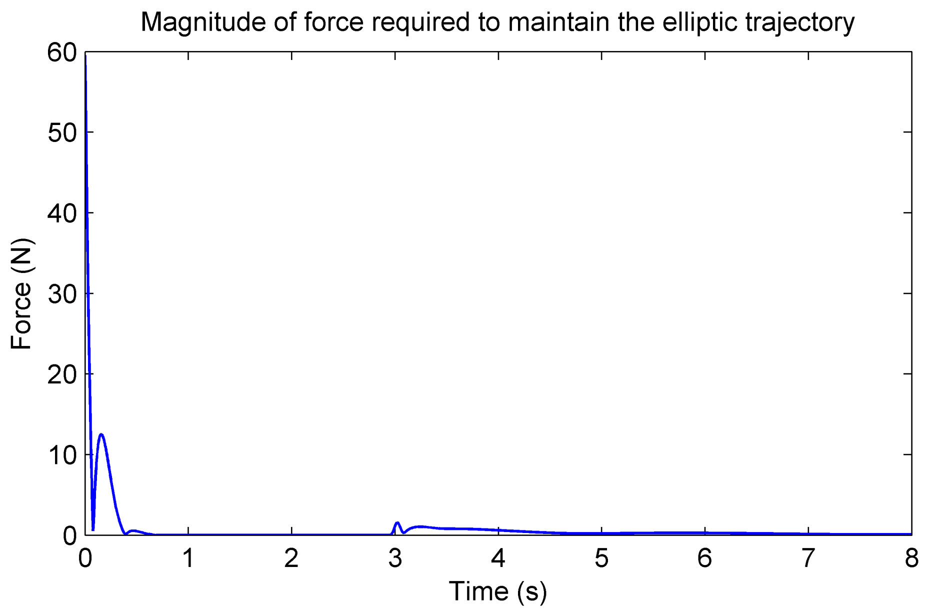

FIGURE 4.5: Force magnitude that the haptic device should apply at the end effector to maintain it on the elliptic trajectory

that brings P to Q, and is the method used on this work to constrain the end-effector into the desired trajectory. Let us define a surface of class C^2 on R^3: $g(a, b, c)$, and the distance d between $Q(a, b, c)$ and $P(x, y, z)$ as:

$$d(a, b, c)^2 = (x - a)^2 + (y - b)^2 + (z - c)^2 \tag{4.16}$$

The problem becomes:

$$\begin{aligned} \underset{(a,b,c)}{\text{Minimize}} \quad & d(a, b, c)^2 \\ \text{subject to} \quad & g(a, b, c) = 0 \end{aligned} \tag{4.17}$$

The optimization problem 4.17 can be solved by the Lagrangian method. The Lagrangian function L and the Lagrangian multiplier λ are introduced:

$$L = d^2 - \lambda * g(a, b, c) \tag{4.18}$$

The minimum occurs when:

$$\begin{aligned}\frac{\partial L}{\partial a} &= 0\\ \frac{\partial L}{\partial b} &= 0\\ \frac{\partial L}{\partial c} &= 0\\ \frac{\partial L}{\partial \lambda} &= 0\end{aligned} \tag{4.19}$$

The solution of the above system of equations gives us the coordinates of Q in function of the coordinates of P. Unfortunately, it can't always be solved explicitly. In that case, it would have to be solved numerically, which might or might not be possible to be done in real time. The force is:

$$F = -k*(P-Q)-c*(\dot{P}-\dot{Q}) = -k\ (\Delta x, \Delta y, \Delta z)-c\ (\Delta \dot{x}, \Delta \dot{y}, \Delta \dot{z}) \tag{4.20}$$

In our case study, the trajectory that the door describes is very simple; it can be represented by the intersection of a cylinder with a plane perpendicular to its main axes. The equation of the plane is $x = 0$, and the equation for the cylinder is $y^2 + z^2 = R^2$ or $r(\theta) = R$. The plane restricts the trajectory to $x = 0$, while the cylinder restricts it to a constant radius inside the plane.

It is evident that the minimum distance between a point and the plane $x = 0$ is exactly the value of the x coordinate. Therefore, force applied by the plane should be:

$$F_{k_x} = -k_x\ x - c_x\ \dot{x} \tag{4.21}$$

On the particular case of the cylinder, the radial distance coincides with the minimum distance, so both methods become equivalent. The equivalent physical example is a body connected to a spring attached to a revolute joint.

Solving Eq. 4.17 and 4.19 for the cylinder $g(a, b, c) = b^2+c^2-R^2 = 0$:

$$\begin{aligned}
\frac{\partial L}{\partial a} &= -2(a-x) = 0 \\
\frac{\partial L}{\partial b} &= -2b\lambda - 2(-b+y) = 0 \\
\frac{\partial L}{\partial c} &= -2c\lambda - 2(-c+z) = 0 \\
\frac{\partial L}{\partial \lambda} &= b^2 + c^2 - R^2 = 0
\end{aligned} \tag{4.22}$$

The group of equations (4.22) forms a non-linear system of algebraic equations. There are two solutions for this system in λ, a, b and c. The first solution is:

$$\begin{aligned}
&\lambda = 1 - \frac{\sqrt{y^2+z^2}}{R} \\
&a = x \\
&b = \frac{Ry}{\sqrt{y^2+z^2}} \\
&c = \frac{Rz}{\sqrt{y^2+z^2}}
\end{aligned} \tag{4.23}$$

The second solution is:

$$\begin{aligned}
&\lambda = 1 + \frac{\sqrt{y^2+z^2}}{R} \\
&a = x \\
&b = -\frac{Ry}{\sqrt{y^2+z^2}} \\
&c = -\frac{Rz}{\sqrt{y^2+z^2}}
\end{aligned} \tag{4.24}$$

If we define:

$$tan(\theta) = \frac{y}{z} \tag{4.25}$$

We can write:

$$\begin{aligned} b &= R\ sin(\theta) \\ c &= R\ cos(\theta) \end{aligned} \tag{4.26}$$

where both solutions are contemplated. The angle θ can be seen as the angle formed between the door and a horizontal plane.

The distance $P - Q$ (in relation to the cylinder) is:

$$P - Q = (x, y, z) - (a, b, c) = (0, y - R\ sin(\theta), z - R\ cos(\theta)) \tag{4.27}$$

If we define:

$$\Delta y = y - Rsin(\theta) \tag{4.28}$$

$$\Delta \dot{y} = \dot{y} - \dot{\theta}Rcos(\theta) \tag{4.29}$$

$$\Delta z = z - Rcos(\theta) \tag{4.30}$$

$$\Delta \dot{z} = \dot{z} + \dot{\theta}Rsin(\theta) \tag{4.31}$$

We have the forces acting on the end effector as:

$$F_{k_y} = -k_r\ (\Delta y) - c_r\ (\Delta \dot{y}) \tag{4.32}$$

$$F_{k_z} = -k_r\ (\Delta z) - c_r\ (\Delta \dot{z}) \tag{4.33}$$

Due to the damping term c_r, there will be dissipation of energy of the system while the end effector moves around the trajectory, reducing its speed over time, which should not happen. However, the effect can be negligible for appropriate values of c_r and k_r, or, if it is not negligible, energy can be added through a "negative damping" coefficient c_e creating a force that acts tangent to the trajectory, removing or greatly minimizing this effect.

In the case of a cylinder, the forces that compensates the dissipation of energy has the following form (with $c_e > 0$):

$$F_{y_e} = c_e R cos(\theta)\dot{\theta} \tag{4.34}$$

$$F_{ze} = -c_e R sin(\theta)\dot{\theta} \tag{4.35}$$

In Fig. 4.6, it is shown that the end effector remains on the desired trajectory, given an initial speed. In Fig. 4.7 we can see the force being applied by the haptic device, and on Fig. 4.8 we compare the variation of kinetic energy between using an appropriate coefficient c_e and not using it.

4.3.3 Haptic Behavior Modelling

This section describes the models that calculate the force feedback responsible for delivering the physical sensations during the act of opening the door. We divided into 4 phases: the unlocking phase, locking phase, the transition between the locked and unlocked state, and the phase where the door freely moves, which then are linked together into a single continuous equation. Part of the assessment of how the dominant forces behave has been done on Chapter 3, and it will be shown how the haptic behavior can be modelled after them.

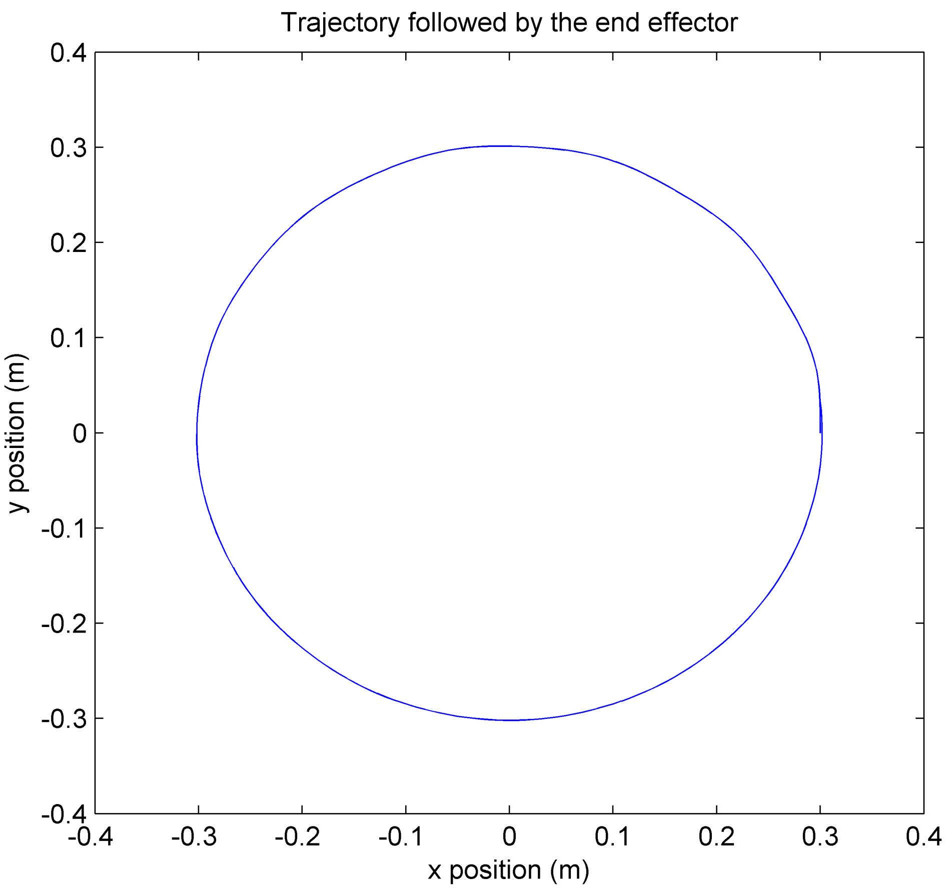

FIGURE 4.6: Circular trajectory followed by the system given an initial speed

4.3.3.1 Activation functions

In order to join discrete behaviours into a continuous equation, the use of activation functions was adopted, i.e. functions that assumes the value of 1 or 0 depending on the input. A wide variety of sigmoid functions (functions that have an "S" shape) can be used as activation functions, e.g. the hyperbolic tangent or the error function. The adopted function was based on the hyperbolic tangent:

$$S(\theta, \theta_0) = (tanh(a(\theta - \theta_0)) + 1)/2, \quad a > 0. \tag{4.36}$$

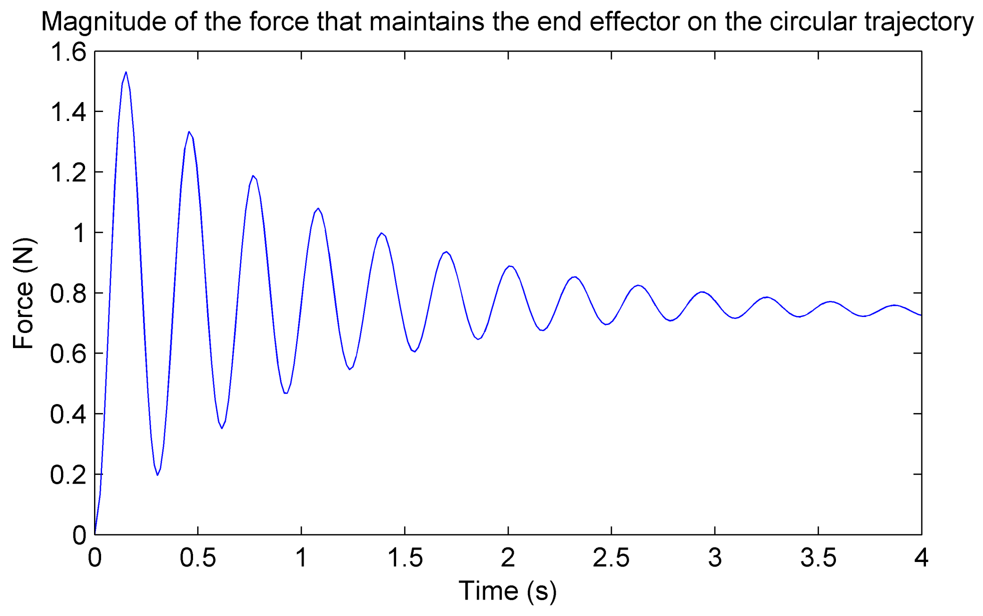

FIGURE 4.7: Magnitude of the force required to constrain the movement of the end effector inside the desired trajectory

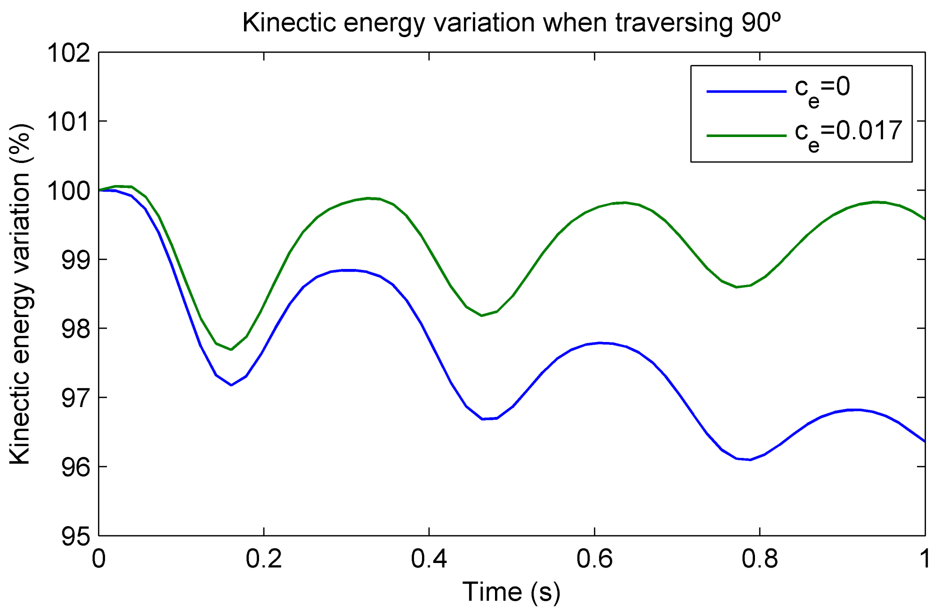

FIGURE 4.8: Kinetic energy variation when the end effector traverses 90°, which depends on the value of c_e

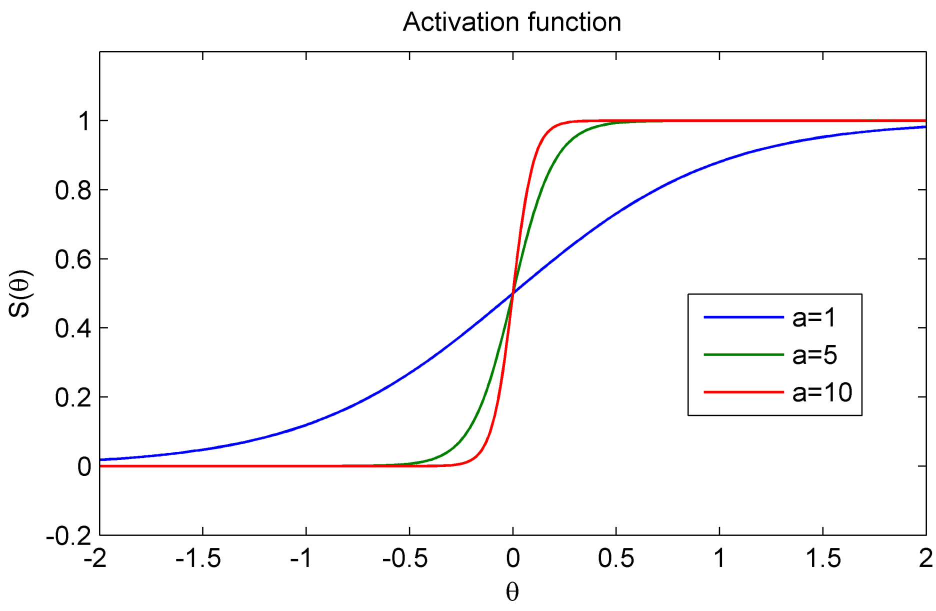

FIGURE 4.9: Effect of a coefficient on activation function

The above function tends to the step function as a increases. Therefore it is assumed:

$$S(\theta, \theta_0) \approx 1 \text{ for } \theta > \theta_0; \tag{4.37}$$

$$S(\theta, \theta_0) \approx 0 \text{ for } \theta < \theta_0. \tag{4.38}$$

4.3.3.2 Unlocking Phase

The unlocking phase corresponds to the situation where the door is being opened, where the force is applied by latch mechanism. The characteristics deemed relevant to represent the sensation were the faint pulling effect the door executes at the hand of the operator, on the region where some looseness can be felt (red square on Fig. (4.10)), and the maximum force necessary to unlock the door, on the rigid region (green circle on Fig. (4.10)).

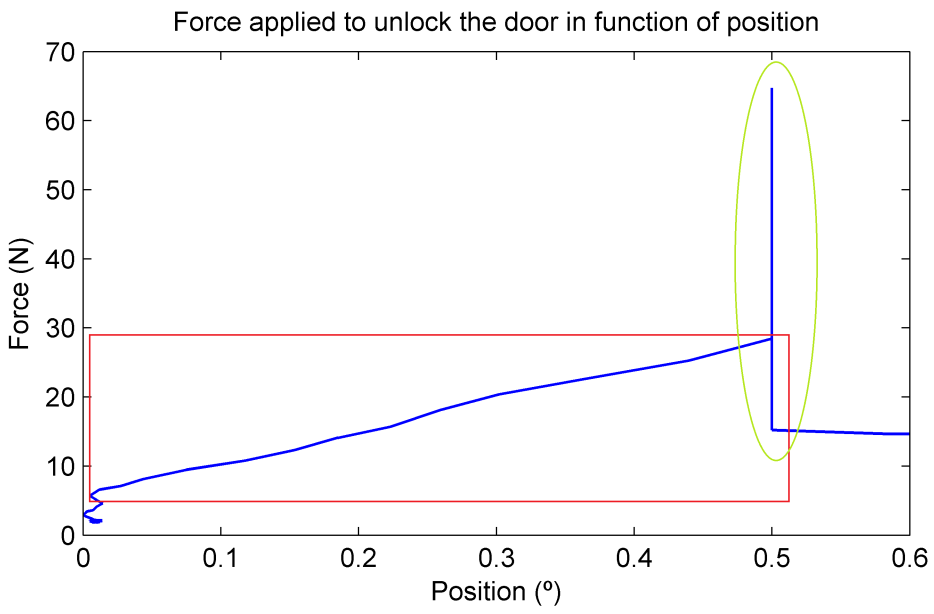

FIGURE 4.10: Analysis of force required to unlock the door

These distinctive behaviors were modelled as a spring: the red region was modelled as a spring with low stiffness k_l, while the green region was modelled as a spring with very high stiffness k_∞. The angle where the stiffness change is called θ_0, the angle where the door finally opens is called θ_1. The activation function S is used as a transition between those regions. It should be remembered that the value of the angle when the door is closed is at its maximum, $\pi/2$, and it decreases as the door opens, so $\theta_0 > \theta_1$. The equation representing the force applied by the end-effector at the user is:

$$F_o = [k_l(\theta - \pi/2) - k_{1\infty}(\theta - \theta_0)\ S(\theta, \theta_0)]\ S(\theta, \theta_1) \tag{4.39}$$

4.3.3.3 Locking phase

The locking phase corresponds to the situation where the door is being closed, where the force is being applied by the latch mechanism. The behavior is identical to the yellow region on the unlocking phase, and thus is modelled as a very rigid spring, though the force goes in the opposite direction. For practical implementations, such as reducing vibrations or eliminating elastic collisions, a viscous damping c_∞ is also added:

$$F_c = k_{2\infty}(\theta - \theta_3)\ [S(\theta,\theta_3) - S(\theta,\theta_2)] - c_\infty\dot{\theta}[S(\theta,\theta_3) - S(\theta,\theta_2)] \tag{4.40}$$

Where F_c is the force the haptic device applies, while θ_2 and θ_3 is the region where the locking phase is valid, being $\theta_2 > \theta_3$.

4.3.3.4 Transition between locked and unlocked state

Between the locked and unlocked state, there is a very small region where the door does not stay: it will either go back to the region where it is locked, or where it is unlocked. To force the haptic device to simulate that region, a negative viscous damping c_n is used: the system becomes unstable at that region, and is forced to move away. Therefore:

$$F_t = c_n\ \dot{\theta}\ [S(\theta,\theta_2) - S(\theta,\theta_1)] \tag{4.41}$$

Where F_t is the force the haptic device applies, while θ_1 and θ_2 is the region where the transition phase is valid, being $\theta_1 > \theta_2$.

Since during the locking/unlocking phase theta is very small, the only relevant component of the force is on z direction.

4.3.3.5 Moving Phase

The moving phase corresponds to the situation where the door can move freely around its point of rotation. The analysis of the mechanism that controls the door indicates that dry friction is the most relevant effect. Measures confirm it: the velocity decays at an approximate constant rate, which implies a constant force opposing the movement is at work (Fig. 3.19). Therefore characteristic deemed most relevant was the friction, which should correspond to degree of "smoothness". The equation tries capture different kinds of "smoothness", in this case represented by dry friction and the asymmetry of forces caused by whether the angle is decreasing or increasing in value, at different positions. Forces responsible for returning the device and viscous friction were also represented, even if they were not present on the original product. A force responsible to make the door return to an equilibrium position (such as automatically closing) is also present.

The term responsible for the viscous friction is:

$$F_v = -c_v\ \dot{\theta} \tag{4.42}$$

A return force can be caused by a spring force, where the equilibrium point is at an angle θ_k, and its equivalent stiffness can change in function of position, if desired, and can be broken into different components arbitrarily, and i defines which component is being referred. The magnitude would be:

$$F_{ret_i} = -k_{ret_i}(\theta)\ (\theta - \theta_k) \tag{4.43}$$

The dry friction is modelled as an hyperbolic tangent:

$$F_{at} = -c_{at} tanh(a\dot{\theta}) \tag{4.44}$$

The coefficient a defines the slope of the curve (on the same way of the activation function S) and c_at corresponds to the friction force magnitude.

We impose two different friction force magnitudes: one value when the door is closing ($\dot{\theta} > 0$) and another when the door is opening ($\dot{\theta} < 0$).

For $\dot{\theta} > 0$:

$$F^+_{dry_k} = -c^+_{at}\ tanh(a\ \dot{\theta}) \tag{4.45}$$

For $\dot{\theta} < 0$:

$$F^-_{dry_k} = -c^-_{at}\ tanh(a\ \dot{\theta}) \tag{4.46}$$

The dry friction force would become (Fig. 4.11):

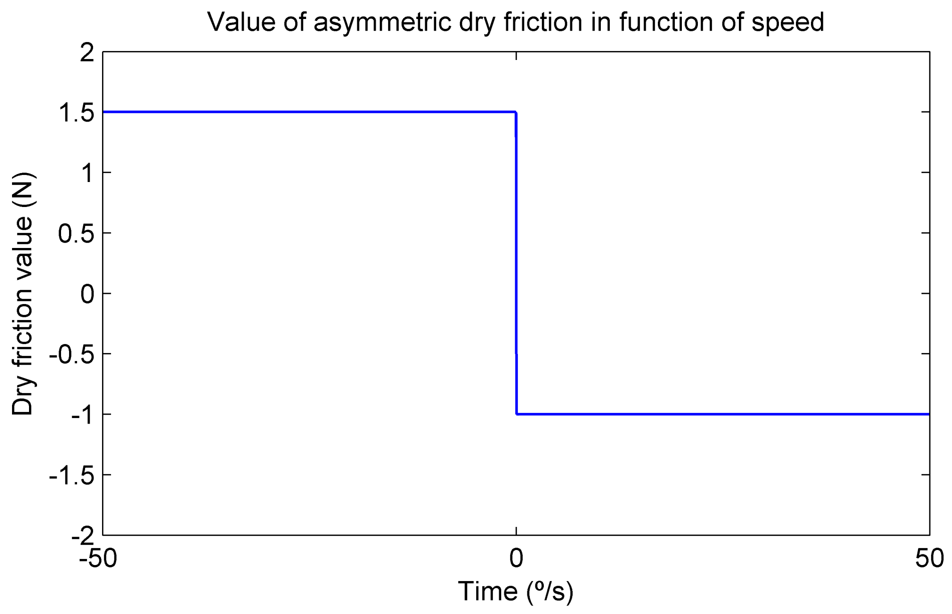

FIGURE 4.11: The values of the friction can be different depending on whether the door is being opened or closed. In this example, the value of the dry friction is higher when the value of the angle is decreasing (more force required to fully open)

$$F_{dry_k} = F^+_{dry_k} + (-F^+_{dry_k} + F^-_{dry_k})\ S(\dot{\theta}, 0) \tag{4.47}$$

Friction can also varies according to the angular position. In that case, activation functions are used to make the transition. For n transitions, we have:

$$F_{dry}(\theta) = \sum_{k=1}^{n} F_{dry_k}(S(\theta, \theta_{(k-1)}) - S(\theta, \theta_k)) \tag{4.48}$$

Where friction remains approximately constant on the interval $[\theta_{(k-1)}\ \theta_k]$. Exemplified on Fig. (4.12) is the case where $n = 2$ and the transition occurs at approximately 45°, with a modest coefficient a on the activation function.

The final dissipative force F_{dis} would be the sum of the one caused by dry friction and the one caused by viscous damping:

$$F_{dis}(\theta) = F_{dry}(\theta) + F_v \tag{4.49}$$

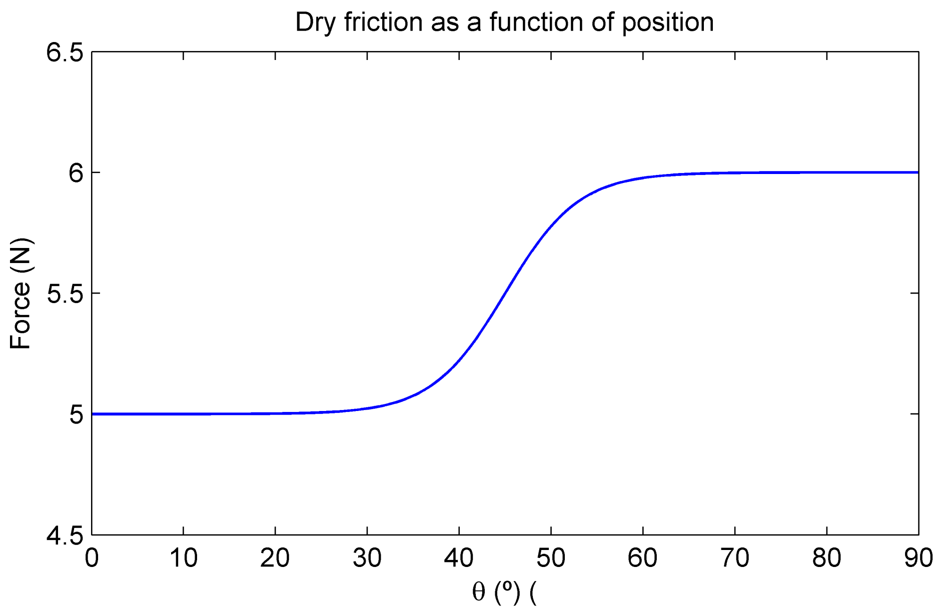

FIGURE 4.12: Variation of the dry friction magnitude according to the position of the door for $n = 2$, selecting the intermediate point at 45°

The vector of the force F_{dis} acts tangent to the trajectory, and can be transformed into Cartesian coordinates by the application of the rotation matrix (4.6):

$$\begin{pmatrix} F_{dis_z}(\theta) \\ F_{dis_y}(\theta) \end{pmatrix} = Rot(\theta)^T * \begin{pmatrix} 0 \\ F_{dis}(\theta) \end{pmatrix} \tag{4.50}$$

4.4 Implementation

The final equation representing the haptic behavior is the sum of all phases of the previous section, on Cartesian coordinates:

$$\begin{pmatrix} F_{hb_z}(\theta) \\ F_{hb_y}(\theta) \end{pmatrix} = \begin{pmatrix} F_{dis_z}(\theta) \\ F_{dis_y}(\theta) \end{pmatrix} + \begin{pmatrix} F_{ret_z}(\theta) \\ F_{ret_y} \end{pmatrix} + \begin{pmatrix} F_c(\theta) \\ 0 \end{pmatrix} + \begin{pmatrix} F_o(\theta) \\ 0 \end{pmatrix} + \begin{pmatrix} F_t(\theta) \\ 0 \end{pmatrix} \tag{4.51}$$

The force feedback is the sum of F_{bh} with the trajectory constraint forces:

$$\begin{pmatrix} F_{h_x}(\theta) \\ F_{h_y}(\theta) \\ F_{h_z}(\theta) \end{pmatrix} = \begin{pmatrix} 0 \\ F_{hb_y}(\theta) \\ F_{hb_z}(\theta) \end{pmatrix} + \begin{pmatrix} F_{x_k}(\theta) \\ F_{y_k}(\theta) \\ F_{z_k}(\theta) \end{pmatrix} + \begin{pmatrix} 0 \\ F_{y_e}(\theta) \\ F_{z_e}(\theta) \end{pmatrix} \tag{4.52}$$

Some simplifications for practical implementation of the final code and real-time adjustments:

- the forces F_{y_e} and F_{z_e} were not considered, as there wasn't noticeable loss of kinetic energy;
- only F_{ret_z} was acting as a return force;
- the viscous damping caused by F_v was also not considered
- the force F_t was also not considered and it was made $\theta_2 = \theta_3$;
- dry friction does not change with the angular position.

Even after these simplifications, Eq. (4.52) still has a total of 16 parameters controlling the characteristics of the force feedback. Also, some issues were found when using those equations to calculate the output forces at the end-effector: high stiffness constants and high damping can generate undesirable vibrations, even making the device unstable. The same problem was found when using very high slopes on the activation functions when the forces were large. To deal with that problem, the combination of maximum stiffness, damping and activation functions slopes that guarantee lack of significant vibrations was found by trial-and-error. In the case of the force required to open and close, the maximum stiffness did not guarantee large forces on a very small region, therefore, the distance which the spring acts was used to control the force, while the stiffness remained constant, with the maximum value that guaranteed stability. Sometimes there could be ambiguity between the dry friction when the speed is positive or negative at very low speeds, if they had different values. The problem was dealt using a threshold $\dot{\theta}_0$ at the activation function S; instead of $S(\dot{\theta}, 0)$, it was used $S(\dot{\theta}, \dot{\theta}_0)$. The way the code was programmed, each parameter could increase or decrease by pressing a key on the keyboard; therefore, each parameter is assigned to 2 keys. For a matter of convenience, not all possible combinations of variation between parameters were represented simultaneously; if that was the case, not only it would be required to map 32 keys, but would also make tests very time consuming, going beyond the scope of this work.

Code Description

While the codes used can be found on appendix A and B entirely, some descriptions from its structure is described here. There are two main files, one python script, responsible for doing the relevant mathematical operations, and an X3D file, responsible to give the instruction to the haptic interface, visual and auditory rendering. These codes are related only to haptic rendering, not for the visual rendering.

X3D file

The forces are defined as xfun, yfun and zfun (force on x direction, y direction and z direction, respectively):

```
<PositionFunctionEffect> <!--force applied on the device/-->
<GeneralFunction DEF="xfun" containerField="xFunction" function="0"
 params="x,y,z"/>
<GeneralFunction DEF="yfun" containerField="yFunction" function="0"
 params="x,y,z"/>
<GeneralFunction DEF="zfun" containerField="zFunction" function="0"
 params="x,y,z"/>
</PositionFunctionEffect>
```

As the name implies (PositionFunctionEffect), they are, in theory, a function of the position from the device (x, y, z). However, the python script allows to circumvent that limitation by including on the function other variable parameters that can include the speed of the tracker (end-effector).

The tracker position is sent to the python script where it will be subject to a mathematical transformation that will ultimately result on the value of the force applied by the haptic interface:

```
<ROUTE fromNode='HDEV' fromField='trackerPosition'
toNode='Force' toField='Force_x'/>
<ROUTE fromNode='HDEV' fromField='trackerPosition'
toNode='Force' toField='Force_y'/>
<ROUTE fromNode='HDEV' fromField='trackerPosition'
toNode='Force' toField='Force_z'/>
```

The tracker velocity is also sent to the python script, where its value is registered on global variables that can be used in other classes:

```
<ROUTE fromNode='HDEV' fromField='trackerVelocity'
toNode='Force' toField='Vel_y'/>
```

To update the value of the global variables in real time, the class Vel_y from the python file is called, where it returns an integer to a variable that has no effect on the simulation:

```
<ROUTE fromNode="Force" fromField="Vel_y"
toNode="dummyspeed" toField="radius"/>
```

The forces calculated on the python script are sent to be used by the haptic interface:

```
<ROUTE fromNode="Force" fromField="Force_x"
toNode="xfun" toField="function"/>
<ROUTE fromNode="Force" fromField="Force_y"
toNode="yfun" toField="function"/>
<ROUTE fromNode="Force" fromField="Force_z"
toNode="zfun" toField="function"/>
```

Key presses are also used to change the values of certain global variables on the python script:

```
<ROUTE fromNode="N" fromField="keyPress"
toNode="Force" toField="force_a_key" />
<ROUTE fromNode="N" fromField="controlKey"
toNode="Force" toField="force_ctrl_key" />
```

Python file

In python script, the functions that calculate the forces are inside classes that return the value of the calculated force as a variable to X3D:

```
class Force_x(TypedField(SFString, SFVec3f)):
def update(self, trackerPosition):


global Posx
global Posy
global Posz
Posx=trackerPosition.getValue().x
Posy=trackerPosition.getValue().y
Posz=trackerPosition.getValue().z

x=Posx

#Initial force to bring the device to an equilibrium point
#when initializer=1
equilibriumx=(-50*Posx-50*Velx)*initializer

#F_x_k once initializer is set = 0
totalforcex=(1-initializer)*(-300*Posx-100*Velx)+equilibriumx


return str(totalforcex)
Force_x=Force_x()
```

There is also a class responsible for misc tasks: updating global variables with the value of the tracker velocity and creating text files that logs the value of the variables that govern the haptic model equations:

```
class Vel_y(TypedField(SFFloat, SFVec3f)):
def update(self, trackerVelocity):

#Obtain velocities
global Vely
global Velx
global Velz
global dxdt
global dydt
global dzdt

Velx=trackerVelocity.getValue().x
Vely=trackerVelocity.getValue().y
Velz=trackerVelocity.getValue().z

dxdt=Velx
dydt=Vely
dzdt=Velz

#This section stores the time history and the values
#of the variables used to adjust the haptic behavior
file=open('dump.txt','w')
file.write(str('frictionforceup=')+ str(frictionforceup) +
str(' frictionforcedown=') + str(frictionforcedown) +
str(' openforce=') + str(openforce) + str(' closeforce=') +
str(closeforce))
file.close()



return 0
Vel_y=Vel_y()
```

Key presses are used to control the values for the global variables that control the haptic model function, and for details on the coding, see Appendix A.

4.5 Experimental results

On the experiment, we tested how the device acted and how close it was to the expected behavior, and also how was the general performance of the device. Unfortunately, there were significant vibration for high values of stiffness. The device also had difficulty to render the dry friction the way it was modelled when we desired higher values (above $10N$). If we were to improve performance, it would be necessary to have low level access to the control algorithm of the device, where the feedback relationship could change to improve the performance of representing the haptic behavior that we wanted (for example, dry friction would have a specific control loop that needed to be active). The device was not able to render so well certain functions directly from X3D/Python because they demand high gains while requiring stability.

The user moved the device according to his whim, while the operator could increase or decrease the values of the parameters that affected the force that the user should apply to move the device. For the purpose of validating the haptic model, particularly how well friction was represented, we retrieved the values for an experiment with constant friction, where the friction for positive angular speed was $c^+ = 1.5N$ and for negative speed $c^- = 1N$, with an mass of $4kg$.

Since the values of speed and the force applied by the user are measured in Cartesian coordinates, they needed to be transformed to polar coordinates in order to compare obtained measurements with the ones estimated by the haptic model. The transformations are present in Sec. 4.3.1. The equivalent inertia of the system, in this case, is:

$$I = m\ R = 1.2kgm. \tag{4.53}$$

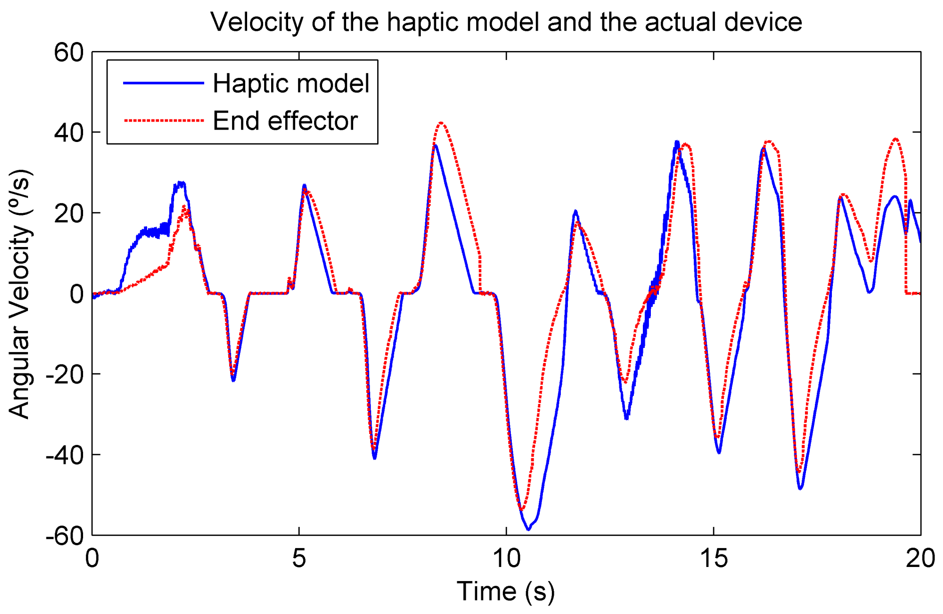

FIGURE 4.13: Comparison between the angular velocity of the haptic model and the actual device when friction is disabled

That inertia represents the resistance to the system to change its angular speed when a force is applied ar the end effector. It is equivalent to the moment of inertia divided by the radius of the trajectory. The response to the input applied by the user (Fig. 4.14) by the haptic model and the actual interface is displayed on Fig. 4.13. It can see that the behavior was very close to the expected, and it demonstrates that the haptic model can be use instead of the actual measurements made by the haptic device.

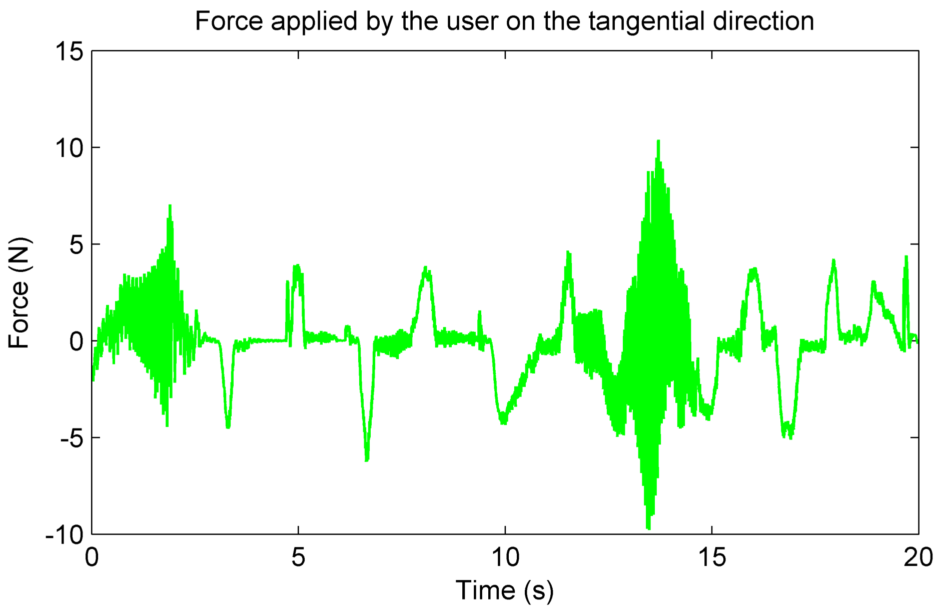

Figure 4.14: Time history of the force applied by the user at the end effector when friction is disabled

Chapter 5

Optimization

A typical mathematical optimization problem consists of minimizing (or maximizing) a (real) function by systematically choosing input values from within a defined set and computing the value of the function with the aim of finding the best solution among all feasible solutions. The standard form for a continuous optimization is

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & F(x) \\ \text{subject to} \quad & G(x) \leq 0 \\ & H(x) = 0 \\ & x_l \leq x \leq x_h \end{aligned} \tag{5.1}$$

where:

- $F(x)$: $\mathbb{R}^n - > \mathbb{R}$ is the objective function to be minimized over the bounded variable x;
- $G(x)$ is a vector that describes the inequality constraints;
- $H(x)$ is a vector that describes the equality constraints.

Solving a well-design optimization problem gives you the best solution to a given problem among a subset of possible solutions, that is, the solutions that satisfies the constraints; because of this it becomes incredibly useful when one wants to design a system where there are multiple solutions satisfying it. The optimization method is adopted because:

1. there are multiple possible combinations of parameters that we must estimate to validate the dynamic model;

2. even if the haptic model and the dynamic model are different, we want to find the parameters of the dynamic model that best fits it to the behavior of the haptic model.

5.1 Designing the problem: Dynamic system parameter estimation

The objective function of our problem computes the maximum difference squared (it also uses the integral of the square difference; sometimes one works better than the other) between state of the dynamic system (θ_s) and the state of the haptic model or physical measurements (θ_h) for a given input force, on a bounded time interval. Defining:

$$f_o(\frac{d^n\theta_s(t)}{dt^n}, \frac{d^n\theta_h(t)}{dt^n}) = (\frac{d^n\theta_s(t)}{dt^n} - \frac{d^n\theta_s(t)}{dt^n})^2 \tag{5.2}$$

The index n indicates which variable of the state is being used:

- $n = 0$ – angular position
- $n = 1$ – angular speed
- $n = 2$ – angular acceleration

The objective function is the maximum value of f on the interval $0 \leq t \leq T$:

$$F(\frac{d^n\theta_s(t)}{dt^n}, \frac{d^n\theta_h(t)}{dt^n}) = \max_{0\leq t\leq T} f_o(\frac{d^n\theta_s(t)}{dt^n}, \frac{d^n\theta_h(t)}{dt^n}); \tag{5.3}$$

or, if the result is not satisfactory, one can try the integral over the interval $0 \leq t \leq T$ of the difference squared:

$$F(\frac{d^n\theta_s(t)}{dt^n}, \frac{d^n\theta_h(t)}{dt^n}) = \int_0^T f_o(\frac{d^n\theta_s(t)}{dt^n}, \frac{d^n\theta_h(t)}{dt^n})dt \tag{5.4}$$

Normally, the difference between using Eq. (5.3) and Eq. (5.4) is negligible, though when one model response differs too much from the other, Eq. (5.3) will minimize the maximum difference at the expense of reducing the smaller differences, while Eq. (5.4) will take into consideration the smaller differences even if that means that a point at the trajectory might have very high inaccuracy. On the models that we used, Eq. (5.3) sufficed, though on some initial models Eq. (5.4) produced better results.

For estimating the parameters of the dynamic system based on measurements, the constraint equation is the differential equation of the dynamic system, solved on the interval $0 \leq t \leq T$, where the parameters are bounded:

$$\begin{aligned} &\dot{x}(t) - f(x(t), p) + g(x(t), p) F_u(t) = 0 \\ &x(0) = x_0 \\ &p_{min} \leq p \leq p_{max}; \end{aligned} \tag{5.5}$$

where

$$\begin{aligned} x(t) &= (\theta_s(t), \frac{d\theta_s(t)}{dt}) \\ \frac{dx(t)}{dt} &= (\frac{d\theta_s(t)}{dt}, \frac{d^2\theta_s(t)}{dt^2}). \end{aligned} \tag{5.6}$$

We use the angular speed to find vector p (corresponding to the parameters of the system), so $\frac{d\theta_h(t)}{dt}$ corresponds to the measurements obtained from the gyroscope, with the force measured from the load cell as $F_u(t)$. In the case of estimating the parameters of the dynamic system based on the behavior of the haptic model, there are two possible solutions:

1. log the data directly from the haptic interface (forces and state history);
2. apply an arbitrary input at the dynamic haptic equations to obtain the time history of the state.

The first solution circumvent the problem of transparency from the device when rendering the forces, if the measures are accurate. The drawback of this solution is that the data comes with noise that could have to be filtered to be used on the optimization, and so information would be lost. The second problem do not address the problem of transparency, but the data comes without noise, and thus can be used directly on the optimization. The simulations are also faster using the second option. In the case of the haptic model, the parameters can be estimated using the same process for their estimation on the dynamic model. The difference is that the constraint equations are reflecting the haptic model (without the latch mechanism):

$$I\ \ddot{\theta}_s(t) = F_{dis}(\theta_s(t), \dot{\theta}_s(t)) + F_{ret_z}\ sin(\theta_s) + F_u(t) \tag{5.7}$$

where I is the inertia of end effector, whose value depends on the its equivalent mass and the chosen value of radius. Note that we do not have to worry about converting a force to a torque nor to take the radius into consideration directly, because the value of I is already taking them into consideration, as it is the moment of inertia in relation to the point of rotation divided by the radius of the trajectory. In this case, if the force is in N, the unit of I is $Kg\ m$.

5.2 Solving the problem: Genetic Algorithm

The simulation software AMESim offers two optimization algorithms: non-linear programming by quadratic Lagrangian (NLPQL) and genetic algorithm (GA). The genetic algorithm offered better results, and thus was chosen to solve the optimization problem. A genetic algorithm (GA) is a method for solving both constrained and unconstrained optimization problems inspired on the natural selection process of biological evolution. A brief description of its terms and the general methodology is given.

Chromosomes or Individuals

For the genetic algorithms, the chromosomes (also called individuals) represent the set of properties (genes), which code the independent variables. The chromosome represents a solution of the given problem and is characterized by the vector of variables. A set of different chromosomes (individuals) forms a generation. By means of evolutionary operators, like selection, recombination and mutation the next generation is created.

Selection

The selection of the best individuals on each generation is based on an evaluation of fitness function, which is also the objective function of the problem. In the system, the fitness function is the sum of the square error between the wanted system response and the real one: individuals with small value of the fitness function will have bigger chances for recombination and for generating offspring.

Recombination

The first step in the reproduction process is the recombination (crossover). In it the genes of the parents are recombined to form an entirely new chromosome.

Mutation

In the terms of genetic algorithm, mutation means random change of the value of a gene in the population. The chromosome, which gene will be changed and the gene itself are chosen by random.

Methodology

The evolution usually starts from a population of randomly generated candidate solutions (individuals or chromosomes) and is an iterative process, where the population in each iteration is called a generation. In each generation, the fitness of every individual in the population is evaluated by computing the value of a fitness function. The more fit individuals are stochastically chosen from the current population, and each individual's set of genes is modified to form a new generation, either through recombination (cross-over) or random variation (mutation). The new generation of candidate solutions is then used in the next iteration of the algorithm; the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

5.2.1 Parameter estimation of the system from measurements

The optimization allows us to estimate the parameters that could not be directly measured, namely the ones that do not correspond to the spatial dimensions of the mechanism: the mass, moment of inertia, spring stiffness, spring pre-load and friction values. The measured force is used as an input on the system, while the output is the state of the system. The output of the system is then compared with the measured state through the objective function. The software allows us to set the population size, the reproduction ratio, the mutation probability, the mutation amplitude and the maximum number of generations, along with the seed value. According to the manual of AMESim, a population larger than 5 times the number of variables often give good results, and the reproduction ratio could be between 50% and 85%. Table 5.1 displays the chosen values to configure the algorithm:

Table 5.2 displays the values that can be obtained through direct measurement; these are the dimensions of the mechanism. For the estimation of the remaining parameters, ten samples were used, and the ranges of the values from the estimated variables can be found at Table 5.3. At figure 5.2, we compare the real behavior of the system with the behavior of model obtained from the optimization of this specific signal, the minimum range and the maximum range, with an maximum error of $0.22m/s$. We assume that the set found through the optimization are close to the true values, though they can actually refer to an equivalent system.

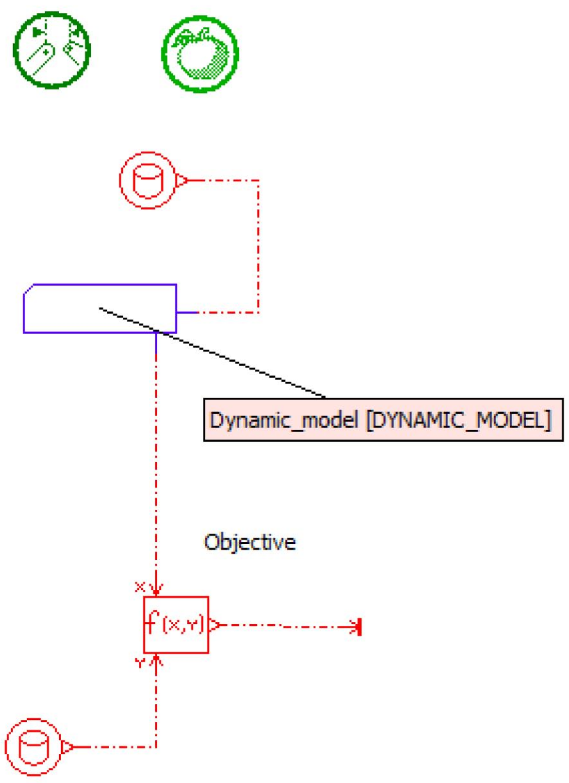

FIGURE 5.1: The output of the system is compared with the desired output after an input

5.2.2 Parameter estimation of the haptic model from measurements

It is possible to make an initial estimative of the parameters from the haptic equation using the same optimization method. The estimation can be made either directly from the measurements, or from the dynamic system model with the estimated parameters. The advantage of using the dynamic system model comes from the fact that the optimization is much faster and it is allowed a greater range of inputs to be tested with different initial conditions. The AMESim model responsible for comparing the haptic model and the dynamic model is displayed on Fig. 5.3.

The block system of haptic model is composed by a block where the angular acceleration is a function of the torque generated by the haptic

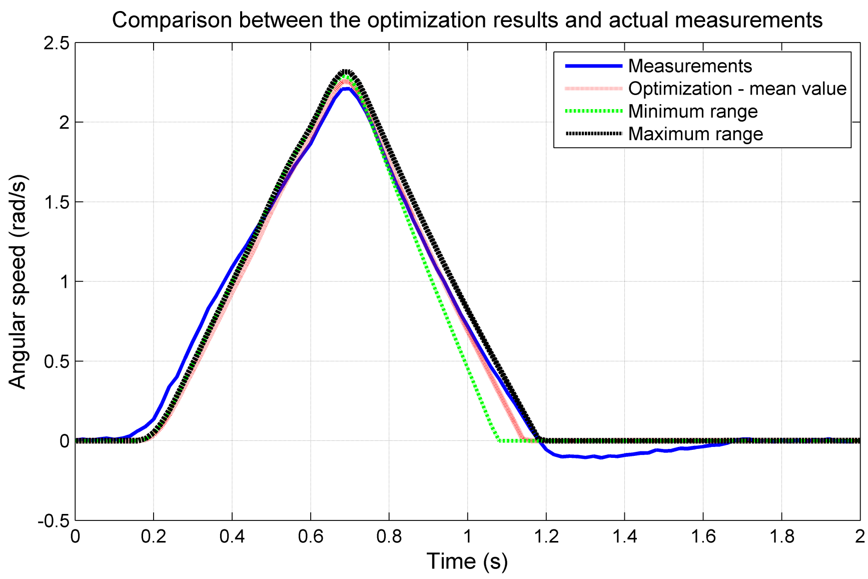

FIGURE 5.2: Comparison between the behavior of the system according to the estimated parameters

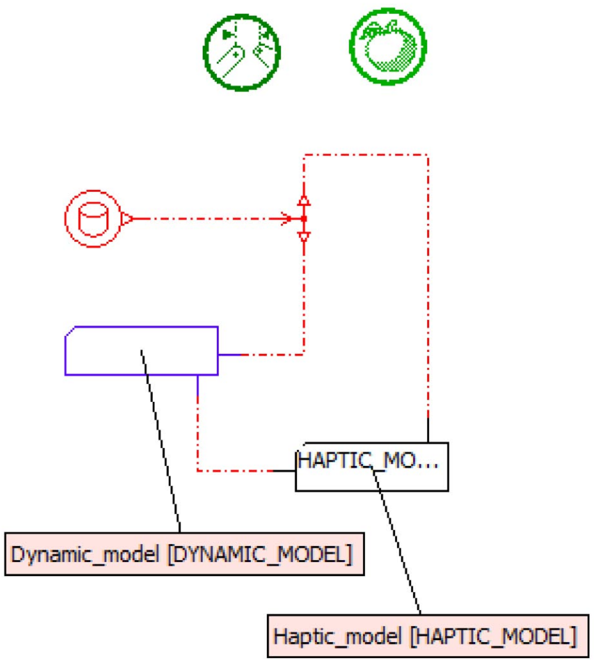

FIGURE 5.3: The haptic model behavior is compared with the system behavior for the same input

Table 5.1: Genetic algorithm parameters

Parameter	Value
Population size	100
Max. number of generations	20
Reproduction ratio	60
Mutation probability	10
Mutation amplitude	0.8

Table 5.2: Measured parameters

Parameter	Value
L	$0.6m$
L_{OA}	$0.03m$
L_{B_x}	-0.01
L_{B_y}	0.09
L_{C_x}	-0.06
L_{C_y}	0.08
L_{D_x}	-0.05
L_{D_y}	0.06

forces and the inertia of the system (Fig. 5.4). It is also where the objective function is calculated.

5.2.3 From haptic model to technical specification

While the equivalent stiffness from the latch mechanism can be directly obtained from the haptic model, the other parameters cannot. The remaining

TABLE 5.3: Estimated parameters of the system

Parameter	Minimum Value (SI)	Maximum Value (SI)
Spring stiffness (K)	393.947	407.33
Spring pre-load (F_{load})	96.820	106.081
Mass (M)	0.387	0.455
Moment of Inertia (I_z)	0.121	0.134
Torque at point O (T_{at_O})	0.00534	0.00695
Torque at point A (T_{at_A})	0.00846	0.00962
Torque at point B (T_{at_B})	0.00476	0.00511
Friction coefficient in B (μ)	0.617	0.996
Force at point C (F_{at_C})	25.971	31.868
Force at point D (F_{at_D})	15.063	22.536

parameters, which are the technical specifications, are analogously obtained using the same method for the initial estimation of the haptic model. This time, however, the parameters of the haptic model are known, while the parameters of the dynamic model are to be obtained. The estimation can be done through the use of the measurements from the state of the haptic device or by applying the same input on the haptic model and dynamic model and minimizing the difference between the outputs. We opted to use the haptic model output instead of the measurements from the haptic device, as the simulation goes much faster.

Due to the amount of variables for the optimization, we decided to maintain the lengths of the original mechanism, changing only the spring stiffness, inertia and frictions, and the obtained result was satisfactory. The comparison between the response of the system and of the haptic model obtained from the experiment (Chapter 4) after the optimization is displayed

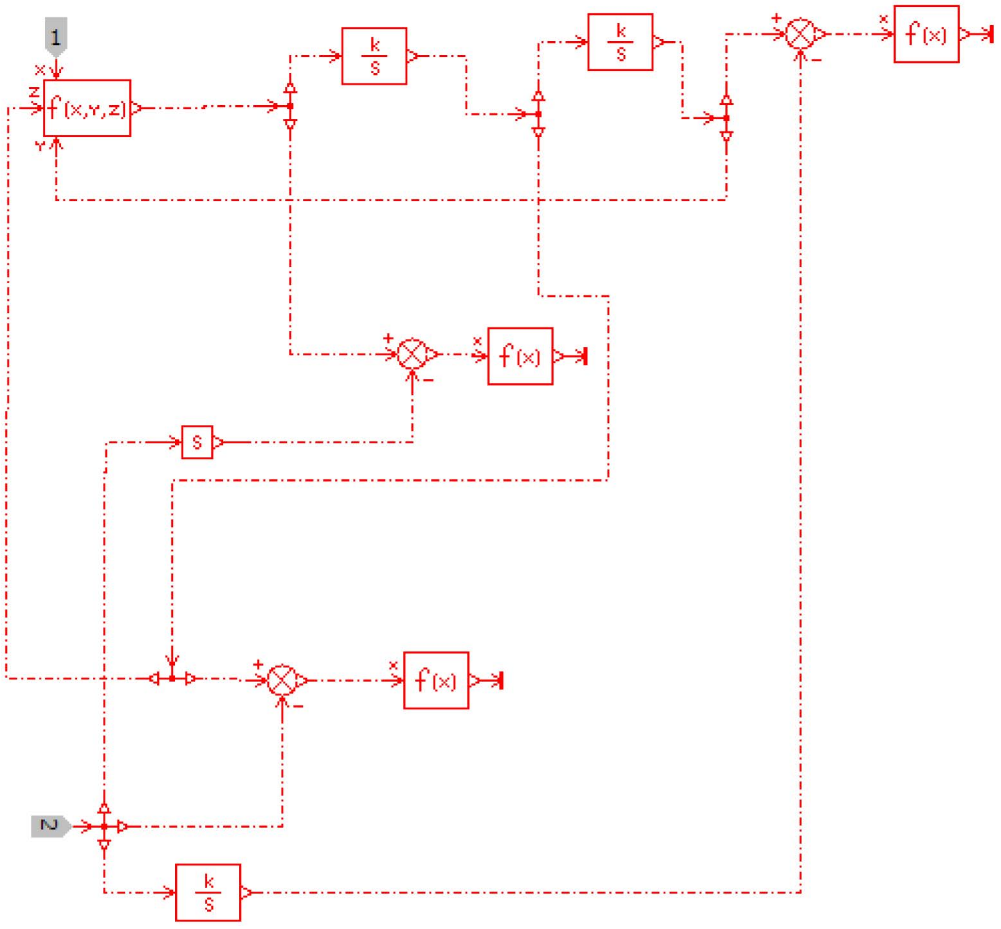

Figure 5.4: The haptic model in AMESim

on Fig. 5.5. The parameters related to the technical specifications are presented on table 5.4.

5.2.4 Redesigning the dishwasher mechanism

The haptic model used to obtain the technical specifications was rather simple, which allowed the existing mechanism to accommodate it. However, the great advantage of using virtual prototypes is that it is not necessary to be restricted to existing mechanisms, and should they represent more complex haptic behaviors, the mechanism would have to be redesigned. This new design will be highly dependent on how the haptic model behaves.

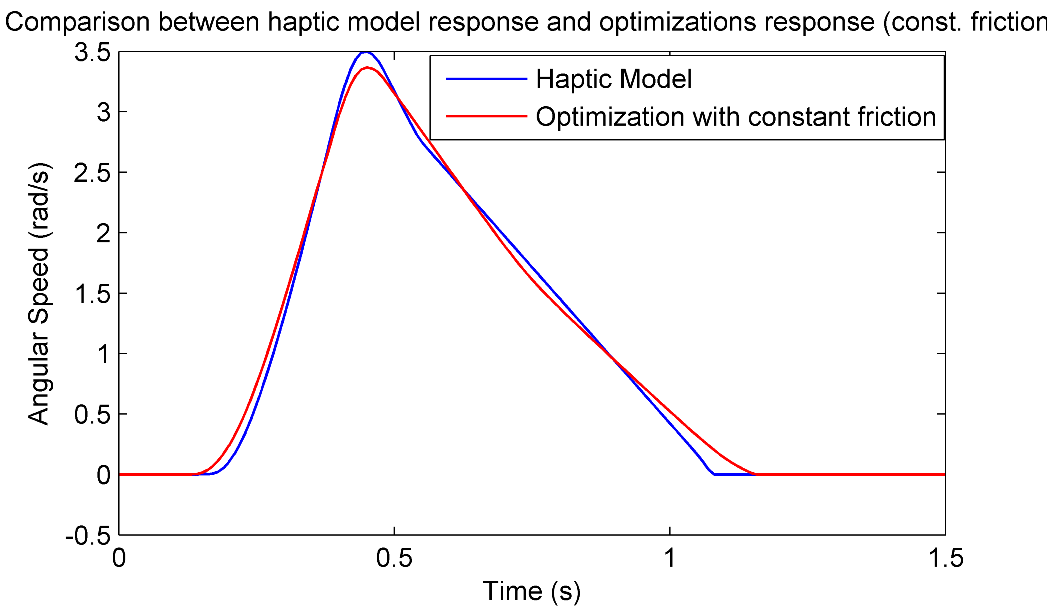

FIGURE 5.5: Comparison between the response of the system and the response of the haptic model for the same input

As an example, if we did not consider the result from Fig. (5.5) satisfactory, we could try to make the magnitude of the friction on point D to be variable according to the position of the door. One way to accomplish this is to divide the trajectory into n sections, and for each interval $[\theta_{k-1}\theta_k]$, the magnitude of the friction can be a fraction A_k, between 0 and 1, of a maximum value F_{at}. We would have:

$$A(\theta) = \sum_{k=1}^{n} A_k(S(\theta, \theta_{k-1}) - S(\theta, \theta_k)) \tag{5.8}$$

The term $(S(\theta, \theta_{k-1}) - S(\theta, \theta_k))$ is a function whose value tends equals 1 if $\theta_{k-1} \leq \theta \leq \theta_k$ and 0 otherwise. That means that the value of $A(\theta)$ will be equal to A_k whenever $\theta_{k-1} \leq \theta \leq \theta_k$. If we divided the interval $0 \leq \theta \leq \pi/2$ into n intervals, we can set the value of A for each interval. The value of friction on point D, in function of the position of the door would be:

$$F_{at_D} = F_at\ A(\theta) \tag{5.9}$$

Table 5.4: Technical specifications obtained from the chosen haptic behavior

Parameter	Value (SI)
Spring stiffness (K)	373.363
Spring stiffness (F_{load})	85.472
Mass (M)	1.232
Moment of Inertia (I_z)	0.0343
Torque at point O (T_{at_O})	0.00431
Torque at point A (T_{at_A})	0.00821
Torque at point B (T_{at_B})	0.00945
Friction coefficient in B (μ)	0.779
Force at point C (F_{at_C})	55.992
Force at point D (F_{at_D})	67.811

We run an example maintaining all the former values from the previous optimization, but we divided the trajectory into 8 sections, and we used another optimization to find the value of $A(\theta)$, whose graphic representation is in Fig 5.6. We compared the results between considering variable friction and constant friction, and it is clear on Fig. 5.7 that using a variable friction offer much better results.

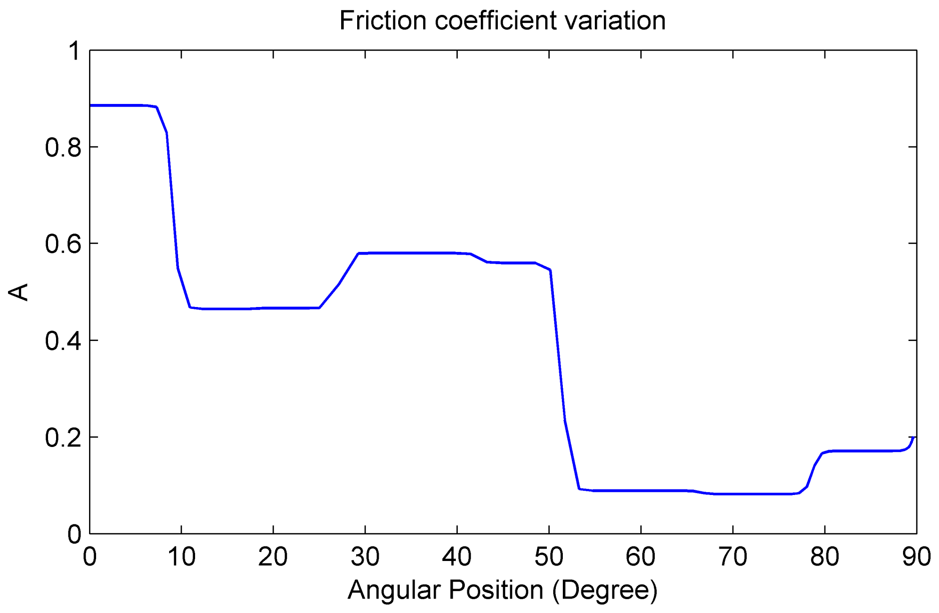

FIGURE 5.6: Value of the coefficent $A(\theta)$, responsible for adjusting the friction at different positions of the door

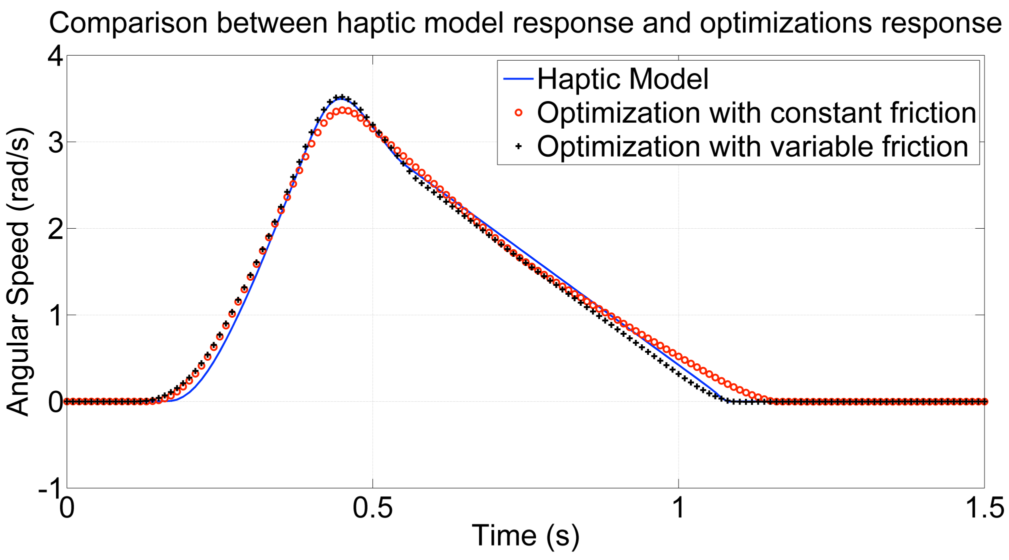

FIGURE 5.7: Comparison between the response of the system and the response of the haptic model for the same input when we use friction that changes with the position of the door [20].

Chapter 6

Conclusion

The work developed has described a methodological approach that enables companies to re-engineer the haptic feedback of the interaction with a product according to users' preferences, and parts of it have already been published in two conferences ([3],[20]) and one journal ([13]). This feedback is part of a more complete multisensory user experience that is becoming important for the success on the market of new products. The methodology proposes the use of haptic interactive Virtual Prototypes as a means to capture the product experience and describes a way to transform perceptual qualitative feedbacks into quantitative design specification. Multisensory interactive Virtual Prototypes are based on visual, auditory and haptic interfaces. While haptic feedback can be adapted on user preferences, visual and sound cues are used to complete the experience. A dishwasher door has been used as case study in order to validate the methodology and to identify the limits. These consisted mainly on the noticeable lack of transparency of the device when using explicit functions that require large gains to be emulated, the size of the workspace for the device used, and the possibility that the results of the optimization might convey a difference that is above the threshold detectable to human senses. In the latter issue, these parameters of the optimization would represent an initial estimation for the technical specifications, and a physical prototype would be necessary to find the adequate values and to tune better the physical sensation. In fact, even with the lack of transparency for the device, the virtual model could still be used as an initial estimation for what is the desired behavior: the order of magnitude for the friction, how fast should the door automatically close, the force to move it, and so on. A physical prototype would be made taking these previously tested factors into consideration, offering a wide range of tests before its commitment

In the case study, the door mechanism has been analysed and transformed in two different models: one simplified and parametric that is used to control the haptic device and allow an interactive design review, and the other detailed used when the users' preferences are captured to extract the design specifications. Both the models, the simplified and the detailed are tuned by means of optimization algorithms. Despite the fact that this methodology has been used to re-engineer an existing door user experience, it has demonstrated that it can be adapted to the design of new products.

Appendix A

Appendix A: X3D file for haptic rendering

```
<X3D>
<Scene>
<IMPORT inlineDEF='H3D_EXPORTS' exportedDEF='HDEV' AS='HDEV' />
<NavigationInfo type="NONE"/>

<!-- Dummy variables/-->
<ViscosityEffect DEF='dummyspeed'
enabled='true'
viscosity='0'
radius='25'
dampingFactor='0'
deviceIndex='0'
/>
<ViscosityEffect DEF='dummyscroll'
enabled='true'
viscosity='0'
radius='25'
dampingFactor='0'
deviceIndex='0'
/>
<ViscosityEffect DEF='dummyclick'
enabled='true'
viscosity='0.0'
radius='25'
dampingFactor='0.0'
deviceIndex='0'
/>
<ViscosityEffect DEF='dummykey'
enabled='true'
viscosity='0.0'
radius='25'
dampingFactor='0.0'
deviceIndex='0'
```

```
/>
<!-- End Dummy variables/-->


<MouseSensor DEF='M'/> <!-- Mouse: scrollUp,scrollDown,leftButton,rightButton/-->
<KeySensor DEF='N'/>
      <Viewpoint DEF='VP1'
    position='0 0.6 2.5'
    orientation='1 0 0 0'
    fieldOfView='0.785398'
    description="Front"
  /> <!-- viewpoint for the CAD model/-->


<!--
<Viewpoint position='0 -0.10 0.8'/> /--><!-- viewpoint for sphere/-->


<!--red sphere/-->
<!--
<Transform translation='0 -0.158 -0.138'>
<Shape>
<Appearance>
<Material diffuseColor='1 0 0'/>
</Appearance>
<Sphere DEF='A' radius='0.3'
/>
</Shape>
</Transform>
/-->

<!--red sphere magnetic/-->
<!--
<Transform translation='0 -0.158 -0.138'>
<MagneticGeometryEffect
            enabled='true'
            startDistance='0.05'
            escapeDistance='0.025'
            springConstant='0'>
            <Sphere USE='A' />
          </MagneticGeometryEffect>
  </Transform>
```

```
/-->



<PositionFunctionEffect> <!--force applied on the device/-->
  <GeneralFunction DEF="xfun" containerField="xFunction" function="0" params="x,y,z"/>
  <GeneralFunction DEF="yfun" containerField="yFunction" function="0" params="x,y,z"/>
  <GeneralFunction DEF="zfun" containerField="zFunction" function="0" params="x,y,z"/>
</PositionFunctionEffect>



   <!--Text for trials/-->
   <Transform translation='0 0 0'>
          <Shape>
            <Text DEF="TESTO"
                    string='"PROVA" '
                    length='' maxExtent='0' solid='true'>
              <FontStyle DEF='F' size='0.1' spacing='1.2' justify='MIDDLE'/>
            </Text>
          </Shape>
   </Transform>
<!--end text for trials/-->

<PythonScript DEF="Force" url="motherdemo.py" />

<!-- Read device coordinate and speed and put in variables/-->
<ROUTE fromNode='HDEV' fromField='trackerPosition'
toNode='Force' toField='Force_x'/>
<ROUTE fromNode='HDEV' fromField='trackerPosition'
toNode='Force' toField='Force_y'/>
<ROUTE fromNode='HDEV' fromField='trackerPosition'
toNode='Force' toField='Force_z'/>
<ROUTE fromNode='HDEV' fromField='trackerVelocity'
toNode='Force' toField='Vel_y'/>
<!-- End Read device coordinate and speed /-->

<!-- Read and compute changes in mouse (scroll and click) /-->
<ROUTE fromNode="M" fromField="leftButton"
toNode="Force" toField="force_left_clk" />
<ROUTE fromNode="M" fromField="rightButton"
```

```
toNode="Force" toField="force_right_clk" />
<ROUTE fromNode="M" fromField="scrollUp"
toNode="Force" toField="force_scr_up" />
<ROUTE fromNode="M" fromField="scrollDown"
toNode="Force" toField="force_scr_down" />
<ROUTE fromNode="N" fromField="keyPress"
toNode="Force" toField="force_a_key" />
<ROUTE fromNode="N" fromField="controlKey"
toNode="Force" toField="force_ctrl_key" />
<!-- End Read and compute changes in mouse (scroll and click) /-->

<!--
<ROUTE fromNode="Force" fromField="PROD_UCT"
toNode="zfun" toField="function"/>/-->


<!-- Use of class to change global variable level, then return values to dummy variables /-->
<ROUTE fromNode="Force" fromField="Vel_y"
toNode="dummyspeed" toField="radius"/>
<ROUTE fromNode="Force" fromField="force_left_clk"
toNode="dummyclick" toField="radius"/>
<ROUTE fromNode="Force" fromField="force_right_clk"
toNode="dummyclick" toField="radius"/>
<ROUTE fromNode="Force" fromField="force_scr_up"
toNode="dummyscroll" toField="radius"/>
<ROUTE fromNode="Force" fromField="force_scr_down"
toNode="dummyscroll" toField="radius"/>
<ROUTE fromNode="Force" fromField="force_a_key"
toNode="dummykey" toField="radius"/>
<ROUTE fromNode="Force" fromField="force_ctrl_key"
toNode="dummykey" toField="radius"/>

<!-- End Use of class to change global variable level,
then return values to dummy variables /-->



<!-- Read variables with coordinates, and calculate the force.
Some of the variables got globally /-->
<ROUTE fromNode="Force" fromField="Force_x"
toNode="xfun" toField="function"/>
<ROUTE fromNode="Force" fromField="Force_y"
```

```
toNode="yfun" toField="function"/>
<ROUTE fromNode="Force" fromField="Force_z"
toNode="zfun" toField="function"/>
<!-- END Read variables with coordinates, and calculate the force.
Some of the variables got globally /-->



<!-- Display text for debugs /-->
<!--<ROUTE fromNode="zfun" fromField="function"
toNode="Force" toField="TESTO_1" />
<ROUTE fromNode="Force" fromField="TESTO_1"
toNode="TESTO" toField="string"/>/-->
<!-- End Display text for debugs /-->


<!-- cylinder magnetic surface /-->
   <!--
   <Transform translation="0.0 -0.15 -0.1" rotation="0 0 1 1.57" >
            <Transform>
              <Shape>
                <Appearance>
                  <Material diffuseColor='1 0 0' />
                </Appearance>
                <Cylinder bottom='false' height='0.0005' radius='0.3'
                          side='true' solid='true' top='false' DEF='E'/>
              </Shape>

              <MagneticGeometryEffect
            enabled='true'
            startDistance='0.03'
            escapeDistance='0.020'
            springConstant='1000'>
                <Cylinder USE='E' />
              </MagneticGeometryEffect>
            </Transform>
          </Transform>
   /-->
   <!-- end cylinder magnetic surface /-->


<Sound  intensity="1" DEF="SOUND" >
```

```
      <AudioClip DEF="APERT" url="apertura.wav"/>
    </Sound>
<Sound  intensity="1" DEF="SOUND2" >
      <AudioClip DEF="CHIUS" url="chiusura.wav"/>
    </Sound>


<PythonScript DEF="PSound" url="sound.py" />

<ROUTE fromNode='HDEV' fromField="trackerPosition"
     toNode="PSound" toField="possound_one"/>
<ROUTE fromNode='HDEV' fromField="trackerVelocity"
     toNode="PSound" toField="velocsound_one"/>
<ROUTE fromNode="PSound" fromField="velocsound_one"
     toNode="SOUND" toField="intensity"/>
<ROUTE fromNode="PSound" fromField="possound_one"
     toNode="APERT" toField="startTime"/>


<ROUTE fromNode='HDEV' fromField="trackerPosition"
     toNode="PSound" toField="possound_two"/>
<ROUTE fromNode='HDEV' fromField="trackerVelocity"
     toNode="PSound" toField="velocsound_two"/>
<ROUTE fromNode="PSound" fromField="velocsound_two"
     toNode="SOUND2" toField="intensity"/>
<ROUTE fromNode="PSound" fromField="possound_two"
     toNode="CHIUS" toField="startTime"/>


<DeviceLog frequency="1000"> <!--force applied on the device/-->

</DeviceLog>


<!-- Dishwasher Model /-->
<Transform translation="0 0 0">
            <Inline url='GEOaperto.x3d' />
          </Transform>

<Transform DEF="C"
translation='0 -0.09 0.28' rotation='0 0 0 0.0'>
<Transform translation='0 0.09 -0.28'>
```

```
<Inline url='GEOsportello.x3d' />
  </Transform>
 </Transform>


 <Transform translation='0 -0.23 0' rotation='1 0 0 1.57'>
<Shape>
<Appearance>
<ImageTexture
repeatS='true'
repeatT='true'
url='textures/parquet.jpg' />
<TextureTransform
center='0 0'
rotation='1.57'
scale='10 10'
translation='-0.1 -0.1' />
</Appearance>
<Rectangle2D size='5.0 2.0' />
</Shape>
 </Transform>

        <PythonScript DEF="PS" url="box_follow.py" />
        <ROUTE fromNode="PS" fromField="changeRotation"
             toNode="C" toField="rotation" />
</Scene>
</X3D>
```

Appendix B

Appendix B: Python file for haptic rendering

```
# w-r-w-y
# w increases the spring force, q decrease
# r reduces oscilations to a certain point, e increases
#(r increases radial damping, e decreases)
# y actives some forces (x force to lock the device), t deactivates
# z decreases the force to open, x increases
# c decreases the friction. v increases
# b decreases the force to close, n increases
from H3DInterface import *
from H3D import *
from H3DUtils import *
import math
import pickle



# Mommy stuff

frictionforceup=0
frictionforcedown=0

# Friction parameters

a1=0.7252/0.75 #mass
a2=(7.0-1.5) #global dry friction
a3=10.41 #global dry frction slope
a4=2.00 #viscous friction
a5=(-6.5+1.5) #second friction
a6=2*5 #second friction slope
a7=0.59 #angular position of friction change
a8=6.132/1.5 #slope of angular change

# Eq Parameter change
```

```
deltaa1=0
deltaa2=0.5
deltaa3=5
deltaa4=0.5
deltaa5=0.5
deltaa6=5
deltaa7=0.1
deltaa8=0.5

# Other changing parameters

impulseforce=0
dthetadt0=0
openforce=-1.2
closeforce=-2.2
returnintensity=0

# declaration of globals

Posx=0
Posy=0
Posz=0
Velx=0
Vely=0
Velz=0
dthetadt=0
theta=0

dydt=0
dxdt=0
dzdt=0




k=2000
deltak=2000
c=200
deltac=200
R=0.3

pi=3.1415926
```

```
a=0

# activators

hapticforce=0
frictionforce=0
lockforce=1
returnactive=0
initializer=1
#unused

click=0
clickdelta=0.3
scroll=0
scrolldelta=0.1
key=0
keydelta=0.5



hdev = None
di = getActiveDeviceInfo()

if( di and len( di.device.getValue() ) > 0 ):
  hdev=di.device.getValue()[0]




class Force_x(TypedField(SFString, SFVec3f)):
def update(self, trackerPosition):

global Posx
global Posy
global Posz
Posx=trackerPosition.getValue().x
Posy=trackerPosition.getValue().y
Posz=trackerPosition.getValue().z

x=Posx

equilibriumx=(-50*Posx-50*Velx)*initializer
```

```
totalforcex=(1-initializer)*(-0*math.tanh(Velx*10)-300*Posx-100*Velx)+equilibriumx


file=open('dump.txt','w')
file.write(str('frictionforceup=')+ str(frictionforceup) +
str(' frictionforcedown=') + str(frictionforcedown) + str(' openforce=') +
str(openforce) + str(' closeforce=') + str(closeforce))
file.close()


return str(totalforcex)
Force_x=Force_x()


class Force_y(TypedField(SFString, SFVec3f)):
def update(self, trackerPosition):
global Posx
global Posy
global Posz


Posx=trackerPosition.getValue().x
Posy=trackerPosition.getValue().y
Posz=trackerPosition.getValue().z

y=0.158+Posy #absolute position y>=0
z=0.138+Posz #absolute position z>=0

theta=math.atan2(y,z) #angular position [0 90°], 0=bottom (fully open), 90° up (closed)

dthetadt=(dydt*z-dzdt*y)/(y*y+z*z) # angular speed



deltaz=z-R*math.cos(theta) #horizontal distance circ and pos
deltay=y-(R*math.sin(theta)) #vertical distance circ and pos

ddeltazdt=dzdt+dthetadt*R*math.sin(theta)
ddeltaydt=dydt-dthetadt*R*math.cos(theta)

Fy1=-k*deltay #circular trajectory spring
```

```
Fy2=-c*ddeltaydt#circular trajectory friction

Fz1=-k*deltaz #circular trajectory spring
Fz2=-c*ddeltazdt#circular trajectory friction

# if dthetadt>dthetadt0+0.1, eq=1, else eq=0
stepangularspeedpositivethreshold=(math.tanh(20*(dthetadt-(0.1+dthetadt0)))+1)/2


# Friction Force
F3=-frictionforcedown*math.tanh(10*dthetadt)-stepangularspeedpositivethreshold*
(frictionforceup - frictionforcedown)
Fz3=-F3*math.sin(theta)
Fy3=F3*math.cos(theta)



# Asymmetric forces equations



# # Force that brings the door back with inpulse

#asymmetricforcey=stepangularspeedpositivethreshold*(6.5+impulseforce)*
math.cos(theta)*impulseactive


equilibriumy=0.8*initializer


totalforcey=(Fy1+Fy2)*(1-initializer)+Fy3+equilibriumy


return str(totalforcey)
Force_y=Force_y()

class Force_z(TypedField(SFString, SFVec3f)):
def update(self, trackerPosition):
global Posx
global Posy
global Posz
global dthetadt
global theta



Posx=trackerPosition.getValue().x
Posy=trackerPosition.getValue().y
Posz=trackerPosition.getValue().z

y=0.158+Posy #absolute position y>=0
z=0.138+Posz #absolute position z>=0

theta=math.atan2(y,z) #angular position [0 90°], 0=bottom (fully open), 90° up (closed)

dthetadt=(dydt*z-dzdt*y)/(y*y+z*z) # angular speed


deltaz=z-R*math.cos(theta) #horizontal distance circ and pos
deltay=y-(R*math.sin(theta)) #vertical distance circ and pos

ddeltazdt=dzdt+dthetadt*R*math.sin(theta)
ddeltaydt=dydt-dthetadt*R*math.cos(theta)

Fy1=-k*deltay #circular trajectory spring
Fy2=-c*ddeltaydt#circular trajectory friction

Fz1=-k*deltaz #circular trajectory spring
Fz2=-c*ddeltazdt#circular trajectory friction




# Friction Force
# if dthetadt>dthetadt0+0.1, eq=1, else eq=0
stepangularspeedpositivethreshold=(math.tanh(20*(dthetadt-(0.1+dthetadt0)))+1)/2


F3=-frictionforcedown*math.tanh(10*dthetadt)-stepangularspeedpositivethreshold*
(frictionforceup - frictionforcedown)
Fz3=-F3*math.sin(theta)
Fy3=F3*math.cos(theta)
```

```
# Spring position unlocking state [between 90 and theta1, no spring,
# between theta 1 and theta 2, stiff spring]

theta1=pi/2-0.5*3.1415/180
theta2=pi/2-(1.7+openforce)*3.1415/180

# Spring positionlocking state [between theta3 and theta4,
# note that their position depends on the unlocked state]


theta3=pi/2-(1.7+openforce)*3.1415/180
theta4=pi/2-(5+openforce+closeforce)*3.1415/180


# angular steps for unlocking state


step01=(math.tanh(1000*(theta-theta1))+1)/2;
step12=(math.tanh(1000*(theta-theta2))+1)/2-(math.tanh(1000*(theta-theta1))+1)/2;

# angular step for locking state

step34=(math.tanh(1000*(theta-theta3))+1)/2-(math.tanh(1000*(theta-theta4))+1)/2;




# Asymmetric forces equations

stepspeedznegative=(-math.tanh(20*dzdt)+1)/2 #if dzdt<0, eq=1, else eq=0
# if dthetadt<0, eq=1, else eq=0
stepangularspeednegativethreshold=(-math.tanh(20*(dthetadt))+1)/2
# if dthetadt>dthetadt0+0.1, eq=1, else eq=0
stepangularspeedpositivethreshold=(math.tanh(20*(dthetadt-(0.1+dthetadt0)))+1)/2

# Return force

thetareturn=(45)*3.1415/180
# 1 if theta>thetareturn, else 0
stepreturnforce=(math.tanh(1000*(theta-thetareturn))+1)/2;
```

```
returnforce=-1.5*stepreturnforce*returnactive*(1+returnintensity)

# Unlocking and locking forces
ksoft=1000

springunlocking=-(pi/2-theta)*ksoft*step12+(dthetadt)*ksoft/10*step12
springlocking=((theta4-theta)*ksoft*step34-(dthetadt)*ksoft/100*step34)*
stepspeedznegative*lockforce


# Force that brings the door back with inpulse


# asymmetricforcez=-stepangularspeedpositivethreshold*(0.5+impulseforce)*
math.sin(theta)*impulseactive
equilibriumz=-0.8*initializer

# Final force

totalforcez=(Fz1+Fz2)*(1-initializer)+Fz3+(springunlocking+springlocking)+equilibriumz



return str(totalforcez)
Force_z=Force_z()

class Vel_y(TypedField(SFFloat, SFVec3f)):
def update(self, trackerVelocity):
global Vely
global Velx
global Velz
global dxdt
global dydt
global dzdt

Velx=trackerVelocity.getValue().x
Vely=trackerVelocity.getValue().y
Velz=trackerVelocity.getValue().z

dxdt=Velx
dydt=Vely
```

```
dzdt=Velz





return 0
Vel_y=Vel_y()




class testo(TypedField(MFString, SFString)):
def update(self, event):
global int_ervallo
routes_in1 = self.getRoutesIn()
TesTo = routes_in1[0].getValue()
return ["Intervallo:      "+str(theta),"Legge forza:",TesTo]

TESTO_1=testo()





class left_clk(TypedField(SFFloat, SFBool)):
def update(self, leftButton):
global click
if leftButton :
click=click+clickdelta
return click

force_left_clk=left_clk()



class right_clk(TypedField(SFFloat, SFBool)):
def update(self, rightButton):
global click
if rightButton :
click=click-clickdelta
return click
```

```
force_right_clk=right_clk()

class Scr_up(TypedField(SFFloat, SFBool)):
def update(self, scrollUp):
global scroll
if scrollUp :
scroll=scroll+scrolldelta
return scroll
force_scr_up=Scr_up()


class Scr_down(TypedField(SFFloat, SFBool)):
def update(self, scrollDown):
global scroll
if scrollDown :
scroll=scroll-scrolldelta
return scroll

force_scr_down=Scr_down()


class a_key(TypedField(SFFloat, SFString)):
def update(self, keyPress):
global k
global a
global c
global hapticforce
global a1
global a2
global a3
global a4
global a5
global a6
global a7
global a8
global frictionforce
global lockforce
global openforce
global closeforce
global returnactive
global initializer
global returnintensity
global impulseforce
global frictionforceup
global frictionforcedown

a=keyPress.getValue()
if a == 'q' :
initializer=1
return initializer
if a == 'w' :
initializer=0
return initializer

#x force activator

if a == 'r' :
frictionforceup=frictionforceup+0.5
return frictionforceup
if a == 'e' :
frictionforceup=(frictionforceup-0.5)*(1+math.tanh(50*(frictionforceup-0.5)))/2
return frictionforceup


# friction



if a == 'y' :
frictionforcedown=frictionforcedown+0.5
return frictionforcedown
if a == 't' :
frictionforcedown=(frictionforcedown-0.5)*(1+math.tanh(50*(frictionforcedown-0.5)))/2
return frictionforcedown


if a == 'u' :
openforce=(openforce-0.3)
return openforce
if a == 'i' :
openforce=openforce+0.3
return openforce

# close force value
```

```
if a == 'o' :
closeforce=(closeforce-0.3)
return closeforce
if a == 'p' :
closeforce=closeforce+0.3
return closeforce




# lock

if a == 'c' :
lockforce=0
return lockforce
if a == 'v' :
lockforce=1
return lockforce

# returnforce activator
if a == 'b' :
returnactive=0
return returnactive
if a == 'n' :
returnactive=1
return returnactive

# initializer

if a == 'm' :
initializer=0
return initializer
if a == ',' :
initializer=1
return initializer

# open force value

if a == 'a' :
openforce=openforce-0.3
return openforce
```

```
if a == 's' :
openforce=openforce+0.3
return openforce

# close force value

if a == 'g' :
closeforce=closeforce-0.3
return closeforce
if a == 'h' :
closeforce=closeforce+0.3
return closeforce

# return force intensity

if a == '5' :
returnintensity=returnintensity-0.5
return returnintensity
if a == '6' :
returnintensity=returnintensity+0.5
return returnintensity

# impulse intensity
if a == '3' :
impulseforce=impulseforce-0.5
return impulseforce
if a == '4' :
impulseforce=impulseforce+0.5
return impulseforce




if a == 'd' :
a2=a2-deltaa2
return a2
if a == 'f' :
a2=a2+deltaa2
return a2

if a == 'j' :
a4=a4-deltaa4
```

```
return a4
if a == 'k' :
a4=a4+deltaa4
return a4
if a == '1' :
a5=a5-deltaa5
return a5
if a == '2' :
a5=a5+deltaa5
return a5

return a7
if a == '7' :
a8=a8-deltaa8
return a8
if a == '8' :
a8=a8+deltaa8
return a8



force_a_key=a_key()

#class ctrl_key(TypedField(SFFloat, SFInt32, SFBool)):
# def update(self, actionKeyPress, isActive):
# global key
# a=actionKeyPress
# if a == 97 and isActive :
# key=key+keydelta
# return key
#
#force_ctrl_key=ctrl_key()


class ctrl_key(TypedField(SFFloat, SFBool)):
def update(self, controlKey):
global key
if controlKey :
key=key-keydelta
return key

force_ctrl_key=ctrl_key()
```

Bibliography

[1] C. Spence and A. Gallace. Multisensory design: Reaching out to touch the consumer. *Psychology & Marketing*, 28(3):267–308, 2011.

[2] K. S. Bhutta and F. Huq. Benchmarking–best practices: an integrated approach. *Benchmarking: An International Journal*, 6(1):254–268, 1999.

[3] Guilherme Phillips Furtado, Francesco Ferrise, Serena Graziosi, and Monica Bordegoni. Optimization of the force feedback of a dishwasher door putting the human in the design loop. In *ICoRD'13*, pages 939–950. Springer, 2013.

[4] Encyclopaedia Britannica. Virtual reality (vr), Last accessed on December 2012. URL http://global.britannica.com/EBchecked/topic/630181/virtual-reality-VR.

[5] M. A. Leenders and B. Wierenga. The effectiveness of different mechanisms for integrating marketing and r&d. *Journal of Product Innovation Management*, 19(4):305–317, 2002.

[6] Abbie Griffin and John R. Hauser. Integrating r&d and marketing: a review and analysis of the literature. *Journal of product innovation management*, 13(3):191–215, 1996.

[7] Jennifer A. Harding, K. Popplewell, Richard Y. K. Fung, and A. R. Omar. An intelligent information framework relating customer requirements and product characteristics. *Computers in Industry*, 44(1):51–65, 2001.

[8] Sunghwan Shin, In Lee, Hojin Lee, Gabjong Han, Kyungpyo Hong, Sunghoon Yim, Jongwon Lee, YoungJin Park, Byeong Ki Kang, Dae Ho Ryoo, et al. Haptic simulation of refrigerator door. In *Haptics Symposium (HAPTICS), 2012 IEEE*, pages 147–154. IEEE, 2012.

[9] Michael Strolz, Raphaela Groten, Angelika Peer, and Martin Buss. Development and evaluation of a device for the haptic rendering of rotatory car doors. *Industrial Electronics, IEEE Transactions on*, 58(8): 3133–3140, 2011.

[10] Monica Bordegoni and Francesco Ferrise. Designing interaction with consumer products in a multisensory virtual reality environment: This paper shows how virtual reality technology can be used instead of physical artifacts or mock-ups for the new product and evaluation of its usage. *Virtual and Physical Prototyping*, 8(1):51–64, 2013.

[11] Monica Bordegoni, Francesco Ferrise, and Joseba Lizaranzu. Use of interactive virtual prototypes to define product design specifications: a pilot study on consumer products. In *VR Innovation (ISVRI), 2011 IEEE International Symposium on*, pages 11–18. IEEE, 2011.

[12] Francesco Ferrise, Monica Bordegoni, and Umberto Cugini. Interactive virtual prototypes for testing the interaction with new products. *Computer Aided Design Applications*, 10(3):515–525, 2012.

[13] Francesco Ferrise, Serena Graziosi, Guilherme Phillips Furtado, Monica Bordegoni, and Dino Bongini. Re-engineering of the haptic feedback of a dishwasher door. *Computer-Aided Design and Application*, 10(6): 995–1006, 2013.

[14] LMS. Lms imagine.lab amesim - lms international, Last accessed on November 2012. URL http://www.lmsintl.com/LMS-Imagine-Lab-AMESim.

[15] Vincent Hayward, Oliver R Astley, Manuel Cruz-Hernandez, Danny Grant, and Gabriel Robles-De-La-Torre. Haptic interfaces and devices. *Sensor Review*, 24(1):16–29, 2004.

[16] Richard Q Van der Linde, Piet Lammertse, Erwin Frederiksen, and B Ruiter. The hapticmaster, a new high-performance haptic interface. In *Proc. Eurohaptics*, pages 1–5, 2002.

[17] H3D API. H3d.org: Open source haptics, Last accessed on November 2012. URL http://www.h3dapi.org/.

[18] Python. Python programming language – official website, Last accessed on November 2012. URL http://www.python.org/.

[19] Web3D Consortium. What is x3d, Last accessed on November 2012. URL http://www.web3d.org/realtime-3d/x3d/what-x3d/.

[20] Guilherme Phillips Furtado, Francesco Ferrise, Serena Graziosi, and Monica Bordegoni. Digitalizing and capturing haptic feedback in virtual prototypes for user experience design. In *IEEE DSP 2013*. 2013.