Surface Geometry and the Haptic Rendering of Rigid Point Contacts

by

Kevin Casey Walker

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

© Kevin Casey Walker 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis examines the haptic rendering of rigid point contacts in virtual simulations. The haptic renderers generate force feedback so that the operator can interact with the virtual scenes in a realistic way. They must be able to recreate the physical phenomena experienced in the real world without displaying any haptic artifacts. The existing renderers are decomposed into a projection function and a regulation scheme. It is shown that the pop-through artifact, whereby the virtual tool instantaneously jumps between two distant surface points, is caused whenever the operator encounters a singularity within the renderer's projection function. This was well known for the minimum distance based renderers, but it is shown here that such singularities arise with the constraint based renderers as well.

A new projection function is designed to minimize the existence of singularities within the model. When paired with an appropriate regulation scheme, this forms the proposed mapping renderer. The new projection is calculated by mapping the model onto a canonical shape where the haptic problem is trivial, e.g. a circle in the case of a 2D model of genus zero, which avoids pop-through on smooth models. The haptic problem is then recast as a virtual constraint problem, where the traditional regulation schemes, designed originally for planar surfaces, are shown to introduce a velocity dependent error on curved surfaces that can distort the model's rendering and to couple the regulation towards and dynamics along the constraint. Set stabilization control, based on feedback linearizing the haptic device with respect to a virtual output consisting of coordinates transversal and tangential to the model surface, is proposed as an alternative. It is shown to be able to decouple the system into transversal and tangential subsystems that can then be made asymptotically stable and assigned arbitrary dynamics, respectively.

Acknowledgements

I would like to start by thanking my supervisor, Dr. David Wang, for his guidance during my years as his student. He was always available when help was needed and has provided much needed advice on many occasions. Dr. Christopher Nielsen also had a significant impact on my work and I learned much from co-authoring a couple of papers with him. Dr. Spiro Karigiannis was instrumental in introducing me to the area of differential geometry, guiding me in how it relates to the haptics problem. I would like to thank the members of my PhD Examining Committee Drs. Shahin Sirouspour, Glenn Heppler, Daniel Davison and Dana Kulić for their feedback and guidance. Their interest in my work provided much needed motivation.

I would also like to thank John Daly, Michael Tribou, Andre Hladio, Regina Leung, Roop & Willem Petersen, John Tang and Nizar Messaoudi for their support. The companionship of my fellow grad students has been invaluable and I cherish the friendships that have developed.

The support from my family has been overwhelming. My mother and father have always shown the utmost respect for higher education and consistently encouraged me to pursue my abilities to their furthest reaches. My brother Martin, his wife Carrie, and their children, Josh, Chelsea and Katelyn, have patiently listened as I explained what it was that I was working on,

offering their encouragement and enthusiasm. My in-laws decided to show their support in a very tangible way, providing a shower of gifts as chapters were completed in the writing phase, always providing something to look forward to. This includes baskets of beer, sweets and games from Andrew, Erin, Aidan, Claire, and Michael Jardine (top, left in photos on the previous page), Yvonne, Jacob, Cecilia, Patrick and Elizabeth Simpson (top, middle), Peter, John, Catherine, Thomas and Robert Jardine (top,right), and Claire, Michael and Maria Connell (bottom, middle). The heads of the clan, Lawrence and Janet Jardine, topped it off by presenting me with an engraved box containing a vintage wine (bottom, right) and a celebratory dinner to mark the thesis completion. Their enthusiasm and prayers for this undertaking strengthened my perseverance to make it through to the end,

None of this would have been possible without my wife, Mary-Elizabeth. From our initial decision to move from Halifax to Waterloo so that I could start my PhD, to encouraging me through the final days of writing, her love and support were invaluable throughout this whole process. Her help, including gifts of encouragement (whiskey stones at bottom, left), perspective when I was too wrapped up in my work, and prayers for the graces that I needed, was instrumental in finishing this long running work.

Dedication

To my darling wife and, above all, to God, who invited me to discover His Truth.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computer simulations have long held in an important role in engineering, allowing designers to assess their plans and researchers to test new theories. Interactive simulations are used in the nuclear and aviation industries for teaching procedural skills, where operators can work through training scenarios in a safe learning environment. More recently, researchers began exploring how these simulations could be adapted to teach manual tasks. Dental and surgical procedures require dexterity and precision that can be taught using simulators that incorporate the operator's kinesthetic input through robotic interfaces capable of delivering force feedback. These haptic simulators are essentially teleoperated robotic systems where the remote environment has been replaced with a virtual world [6].

The word haptic is defined as "relating to or based on the sense of touch" [7]. Haptic simulators allow operators to interact with virtual worlds through their sense of touch, either in a purely exploratory mode or in a truly interactive way where the operator can modify the virtual world through their actions. If the tactile feedback from the simulators closely matches that of the physical world, then the operator will be able to learn new skills in training that can later be applied in the physical world. The research work on haptic simulators is organized around creating the most realistic experience possible by designing high performance haptic devices, by accurately simulating the physical phenomena of interest, and by developing controllers that drive the haptic device in order to display those phenomena.

This thesis concentrates on how the haptic device should be controlled to best recreate real world physical interactions. Commonly referred to as the haptic renderer, these algorithms bridge the virtual world and the haptic device, calculating an appropriate response to the operator's actions. Several contact scenarios are considered in this work consisting of the interaction between a rigid tool and a rigid, frictionless surface. The geometric characteristics of the model's surface play a significant role, with the smooth or angular nature of the model having a significant impact on the performance of a haptic renderer. The contact scenarios are chosen to highlight the shortcomings of various algorithms in different situations. The existing haptic renderers are evaluated, enumerating their advantages and disadvantages in different scenarios.

A new haptic renderer based on mapping the virtual objects onto a canonical shape is then proposed to overcome the disadvantages of current methods on curved models. While the new method achieves the desired level of continuity for the response force on curved models, it shares the same shortcoming as the other existing haptic renderers in not explicitly providing the centripetal acceleration required to follow a curved surface. This leads to an error term dependent on the operator's velocity along the surface, distorting the model's shape in practical scenarios.

Nonlinear control techniques are applied to the haptics problem to more fully incorporate information about curved constraints into the controller. The problem is posed as the stabilization of a set consisting of the points on the model's surface that have a velocity tangent to the surface. Feedback linearization is applied to decouple the problem according to the task dimensions parallel and perpendicular to the surface, incorporating the nonlinear aspects of both the haptic device and the surface geometry. The system's dynamics can be separated into transversal and tangential subsystems governing the motion towards and along the model's surface, facilitating the control design and ensuring that the haptic simulator is capable of meeting its design objectives.

1.1 Original Contributions

This thesis makes various contributions in different areas that have been disseminated in various papers [8, 9, 10]. Other contributions were made to the field that are not included in the scope of this thesis [11, 12]. The contributions from these works and this thesis are now summarized.

- I demonstrate in Subsection 4.1.3 that pop-through, which was purportedly solved by the constraint methods, can still occur in certain scenarios. The severity of the pop-through increases with penetration depth.

- A new projection function is constructed for use in a haptic renderer to try and resolve the pop-through problem. In Chapter 5, the model is mapped onto a canonical shape so that the haptics problem can be solved in a straightforward manner in an alternate domain. The resulting solution is then mapped back onto the original model to form a position error vector. Where the previous renderer's projection fields map line segments inside the model onto a point on its surface, the new projection function maps curves onto points on the model's surface.

- To define the mapping function, I propose a new flow velocity function (5.14) to adapt the Level Set Method for use in robotic control applications. This generates the coordinate system that serves as the basis for constructing the mapping between the physical and simplified domains of the mapping projection function.

- I proposed a new haptic regulation scheme in Section 5.4 for use with projection fields whose lines of constant projection are curved.

- I proposed the use of a list of physical phenomena from the interaction between a rigid tool and surface as a set of necessary behaviours for a haptic simulator to recreate in Section 3.2. The models and test trajectories from Section 6.1 then constitute a test suite that provides a fast, efficient means to evaluate a renderer's performance.

- I perform a comparison between the typical haptic regulation schemes listed in Section 4.2 and the proposed regulation scheme Section 5.4. In Section 6.3, it is demonstrated that none of those methods are capable of following curved surfaces. They are all based on local planar approximations of the model's surface that neglect the higher order surface characteristics, which must be taken into consideration or the resulting controller will have an error proportional to the square of the operator's tangential velocity.

- Set stabilization is applied to the haptics problem in Chapter 7 to simultaneously incorporate the nonlinear aspects of the haptic device and the model's surface. The approach is first applied to a model in two dimensions where it is shown to outperform the traditional haptic regulation schemes when re-framed as bilateral virtual constraints.
- The previous results from set stabilization are extended to closed two dimensional surfaces in three dimensions in Section 7.4. To tackle higher dimensional constraints like surface in 3D, the controller must switch between different local coordinate system as the haptic device travels throughout its workspace. A smooth switching controller is designed to address the haptic control problem, with experiments demonstrating its effectiveness and practicality.

1.2 Thesis Overview

The remainder of this thesis is organized as follows:

Chapter 2 provides background context on the haptics problem. Haptics is a multidisciplinary field so this chapter gives a summary of the robotic control and computer science aspects of the problem. Basic assumptions such as the type of haptic device used and the well-posedness of the control problem provide context for the discussion that ensue in later chapters.

Chapter 3 investigates the techniques used to evaluate haptic simulators. Quantifying the results is not as straightforward as in the automatic control context because the system's performance is ultimately determined by how the operator perceives the virtual scene. The behaviour of the system can be analysed qualitatively, however, by listing the physical phenomena that the simulator should display in different scenarios. The point contact between two rigid objects is studied to construct a list of necessary behaviours that haptic simulators should exhibit.

Chapter 4 considers the existing haptic renderers, analysing them according to a projection / regulation framework. Their projection functions are shown to contain discontinuities at various points on the inside of the model. The commonly used haptic regulation schemes are presented. Common ways to extend the basic haptic renderer are included to show how effects like static and viscous friction, textures, force shading and contact impulses can be added to a basic renderer.

Chapter 5 constructs a new haptic renderer based on a novel projection function with an associated regulation scheme. The gridding of the model's interior is performed via curve shortening flows to define a mapping between the model and a circle in an another domain. The projection is performed on the circle, with the results mapped back to the physical space. It is shown that this minimizes the occurrence of pop-through on smooth models.

Chapter 6 compares the renderers from Chapters 4 and 5 across various scenarios. Simulation and experimental results are used to show that none of the renders is capable of correctly rendering all of the presented scenarios. While the constraint renderer is superior at rendering angular objects, the mapping renderer performs better on curved models. The performance of all renderers is shown to converge as the rendered stiffness increases. The haptics problem is then recast as a virtual constraints problem, where the closed loop dynamics are formulated for the regulation schemes in Section 4.2 and Section 5.4. It is shown that none of them can asymptotically stabilize a curved constraint. The controllers all rely on a position error to supply the requisite centripetal acceleration instead of accounting for the surface's curvature directly in the regulation scheme.

Chapter 7 proposes set stabilization as a means of incorporating the constraint curvature information into the controller, so that the transversal and tangential subsystems can be decoupled by a state feedback controller. Feedback linearization is applied to a virtual output consisting of a distance to the constraint and a coordinate along the constraint. This simplifies the control design process by allowing the specification of the desired behaviour along the task space dimensions in a manner that guarantees that the control in one dimension will not impact the other dimensions. Analyses and experiments demonstrate that the set stabilization approach delivers better tacking on a circle than the standard haptic regulation schemes.

The construction of a switching controller is explored for closed two dimensional constraints in three dimensions. The model's surface is covered by an atlas, which is used to generate a set of feedback linearizing controllers defined over different portions of the model's surface. The controller is shown to switch smoothly when transitioning between different coordinate charts in the atlas.

Chapter 8 provides a summary of the results from this thesis and makes suggestions for future research directions to advance the results presented here.

Chapter 2

Haptics Background

The field of haptics is a broad area of study because of the versatility of our sense of touch. Various books [13, 14, 15, 16] and literature reviews [17, 18, 19, 20] show the great diversity of uses for force feedback. Three popular modes of haptic interaction are rehabilitation, visualization and simulation.

In the rehabilitation sciences, patients grasp robotic devices that provide guidance or resistance through a set of exercises to help them recover mobility and motor function lost through strokes or accidents [21]. The haptic device can help patients accomplish motions that they are otherwise unable to accomplish, helping to strengthen weak muscles and stretch soft tissues. Resistive devices can further strengthen muscles and improve motor learning by making regular exercises more challenging. The rehabilitation devices can provide support when re-learning to walk so that patients cannot fall, avoiding injuries.

In the scientific visualization community, complex datasets often arise that cannot be adequately displayed in two or three dimensions, like the velocity vector field of a complicated fluid flow. Haptic visualizations present additional information using tactile feedback to augment the graphical display [22, 23]. For example, vector fields can be displayed by applying a reaction force proportional to the element in the vector field at that point. By expanding the modalities through which information can be communicated between the computer and the operator, more dimensions of a problem can be explored in a natural way that helps build intuition around the behaviour of interest.

In the simulation community, the haptic device is used to interface with a virtual environment, reproducing interactions with the real world for training and educational purposes [24]. Dental and medical simulators let novices hone their skills without putting patients at risk and allow experienced professionals the chance to rehearse and refine complex procedures. The haptic device maps the operator's motions into the virtual world, where physical simulations determine how the operator's actions have affected the patient and calculate the reaction forces that the operator should experience. The performance of the operator can be measured and evaluated based on the operation's objectives, with repeated practice possible until a certain level of skill is demonstrated.

This thesis focuses on the use of haptics in simulators, which bridges the virtual and physical worlds through the use of robotic devices. Haptic simulators are designed to mimic our interactions with the physical world, reproducing the sensations associated with exploring and manipulating objects. The overarching goal of the work presented here is to make the interactions with virtual environments as realistic as possible to the point that the operator ceases to notice that they are manipulating an electromechanical device instead of an actual object. This property is referred to as transparency [25], which was originally defined in the context of teleoperative systems [26]. Transparency provides a measure of how much resistance to motion the robotic device exhibits in free space or how much give there is in a hard surface, quantifying the difference between the ideal and actual system behaviour [27]. While transparency provides a useful, quantifiable performance measure, the ultimate performance of such a simulator can be measured by a haptic Turing Test. The original Turing Test involved a person (interrogator) communicating via a chat program with another person and with a computer program designed to mimic a person [28]. If the interrogator cannot distinguish between the real and artificial intelligence, then the machine is said to have passed the Turing test. A similar test could be constructed for haptic simulators where a blind-folded operator is presented with both a real physical interaction scenario and its simulated counterpart. If the operator cannot consistently distinguish between the two, then the haptic simulator would be said to have passed the test.

This chapter contains a brief overview of the types of haptic devices reported in the literature with their salient properties. A high-level functional block diagram for haptic simulators based on impedance type devices is presented, with introductory details for each of the subsystems involved. The computational structure of the simulator and the major factors affecting system performance are enumerated.

2.1 Haptic Simulators

The great variety of haptic simulators can be broadly categorized according to the nature of the force feedback that they deliver. Tactile devices, like the one pictured in Figure 2.1, focus on stimulating the cutaneous receptors in the skin. The cutaneous receptors respond to local stimuli and provide the sensations of touch, vibration, temperature, pressure and sliding [15]. This type of device is dedicated to reproducing the sensations of direct contact between the operator and an object, as opposed to the indirect contact that takes place when using a tool. A literature review of this type of device can be found in [29].

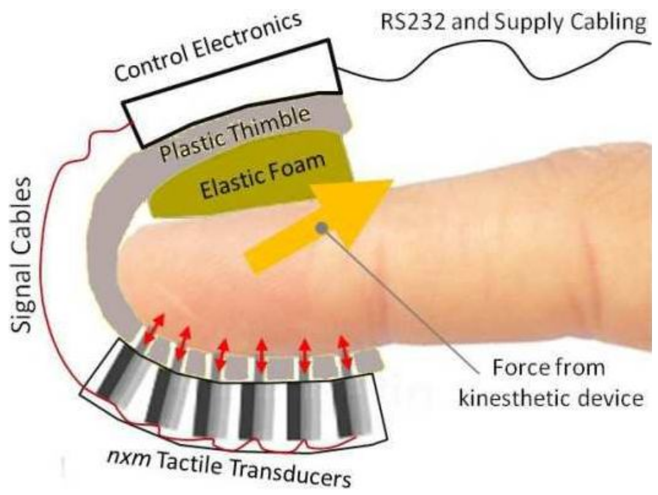


Figure 2.1: Tactile haptic device stimulating the pad of the finger [1]

Kinesthetic devices, on the other hand, focus on stimulating the receptors in the joints and muscles that underpin our sense of proprioception, which tells us the relative position of our limbs and the effort being exerted by our muscles [15]. The kinesthetic sense involves bulk forces experienced through multiple muscles and joints, as opposed to the fine-grained tactile sensations that occur in the immediate vicinity of contact with the skin. The kinesthetic sense plays a particularly important role when we interact with objects through a grasped tool. When using a knife to carve wood, for example, the hand holding the knife does not come directly into contact with the workpiece so the tactile sense plays a less important role. It is the forces arising from the contact and the visual feedback that guide the interaction.

Some kinesthetic devices can be quite complex, involving many degrees of freedom and displaying different forces on different fingers like the glove in Figure 2.2. Most kinesthetic devices, however, have fewer degrees of freedom, with the operator grasping some form of handle attached to a robot manipulator's end-effector. A detailed account the architecture of different haptic devices is given in [30].

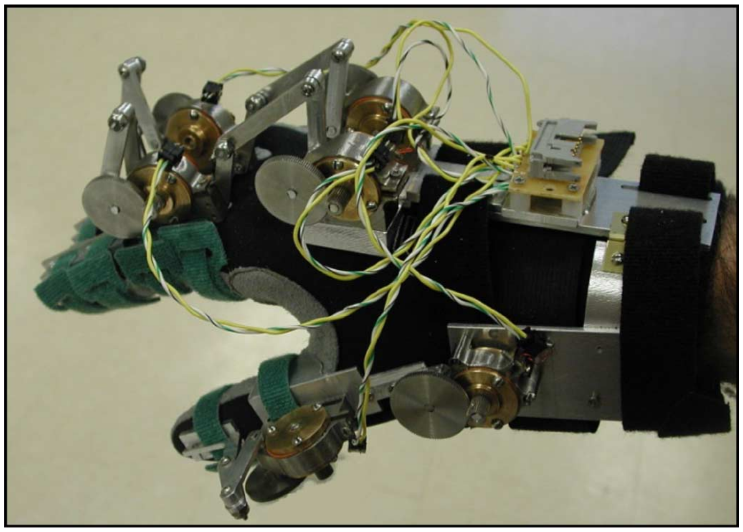


Figure 2.2: A kinesthetic haptic device delivering different force feedback to different fingers [2]

Kinesthetic devices can be classified according to whether they are admittance or impedance type devices. Admittance type devices measure the forces exerted by the operator and command positions / velocities / accelerations. They are typically constructed to have a high mechanical stiffness and employ high gear ratios to mitigate the influence of external disturbances on the device's motion. This construction amplifies the internal friction of the device, making it difficult for the operator to move the device through its workspace when the power is off, i.e. it is non-back-drivable. As such, admittance devices typically rely on force sensors to communicate the actions of the operator to the system, with the ensuing motion of the device being due to primarily to the action of the robot controller. Active admittance type devices include research devices like the 3 DOF parallelogram robot developed in [31] or the 6 DOF admittance device developed by the same group shown in Figure 2.3 [3]. Many general purposes industrial robots can also be used as an admittance type devices when equipped with a force sensor, while Moog's HapticMaster is a commercially available admittance device built specifically for haptic applications [32].

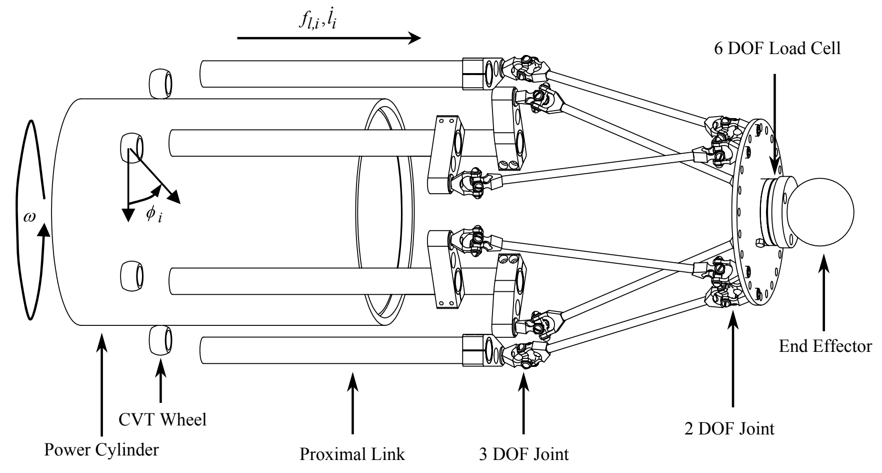


Figure 2.3: The Cobotic Hand Controller admittance type haptic device [3]

While admittance devices are capable of displaying very stiff virtual environments, their construction requires force sensors and precisely machined components, leading to relatively costly implementations [3]. Impedance devices offer a lower cost alternative to admittance devices, but are more limited in the magnitude of the virtual stiffnesses that they can render before exhibiting limit cycles or instabilities [33]. They operate by measuring the positions / velocities / accelerations of the device and delivering forces via their actuators. These devices ideally have little inertia, little friction, accurate position measurements, and high force output capabilities. The devices are back-drivable so that the operator can move the end-effector through the workspace relatively easily when the device is powered off.

To minimize the effect of inertia, wire-based systems suspend the end-effector, like the example in Figure 2.4 [4]. The most common kind of impedance devices, however, are robotic manipulators. These include commercial offerings targeted towards haptics applications like Sensable's PHANToM series [34], Force Dimension's sigma, omega and delta series [35], Haption's Virtuose and Inca series [36], Quanser's Haptic Wand [37] and Barrett Technology's WAM Arm [38].

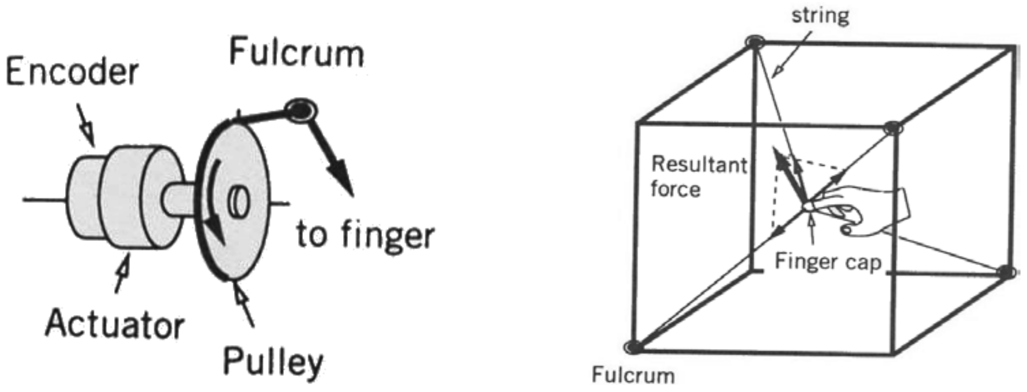


Figure 2.4: Wire-based impedance type haptic device designed to minimize system inertia [4]

These devices typically have a force output limited to tens of Newtons and a position resolution on the order of tens of micrometers. Their limited force capabilities make significant penetrations of the HIP into the model possible, which highlights the importance of considering significant penetration depths on the quality of the rendering as done in Chapters Chapter 4, Chapter 5, and Chapter 6. Given the popularity, availability and lower cost, impedance type robotic manipulators are the platform considered in this thesis. The next section therefore considers the characteristics of this type of device and their use in haptic systems in detail.

2.2 Impedance Device Based Haptic Simulators

The operator uses haptic devices like the one shown in Figure 2.5 by grasping the robot's end-effector to use it as a 3D input device. The device's encoders measure its joint configurations, which determines the position of the end-effector in the robot's workspace through the manipulator's forward kinematics [39]. When the device's position is mapped into the virtual world, it defines the Haptic Interface Point (HIP). The virtual tool is located at the Ideal Haptic Interface Point (IHIP), representing the position of the tool when the constraints of the virtual world are taken into consideration, as shown in Figure 2.6. A visual rendering of the scene is shown on a monitor or 3D goggles, giving the user a sense of their position relative to the virtual environment by drawing the virtual tool at the IHIP. The physics engine simulates various physical phenomena of interest in the virtual world (e.g. collisions, rigid body dynamics, contact friction, elastic model deformations, etc.) and the haptic renderer calculates the force feedback that the actuators of the haptic device deliver to the operator.

The block diagram in Figure 2.7 delineates one possible computational structure found in haptic simulators, but this organization is not universal. If the virtual world is simple enough, for example, the physics engine can be included in the haptics thread. The operator grasps the haptic device, which then share a common position y, and exerts a force F_o on the device. The haptic device measures its joint angles q, which the robot controller translates into the position of the HIP, hip, in the virtual world. The robot controller also generates forces / torques τ on the haptic device's joints based on the commanded response force F_r from the virtual world. The IHIP, $ihip$, calculated by the haptic renderer is forwarded on to the graphics thread for drawing.

Realistic simulators may have to re-create many physical phenomena, leading to a complicated and computationally intensive physics engine. In the example of a surgical simulator, this includes collision detection, multi-body dynamics, friction effects, deformations of complex tissue exhibiting nonlinear elastic behavior and topological changes in the environment due to e.g. cutting and suturing [40]. Other non-haptic effects may also need to be included for greater realism, like fluid dynamic simulations associated with blood and smoke occluding the site of an operation [41]. Each of these components requires computationally intensive numerical simulation, making the physical simulation a challenging problem to be solved online for use with

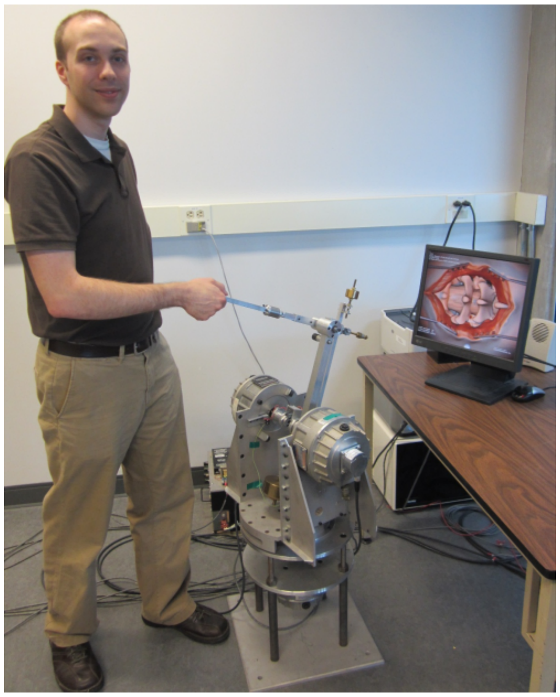
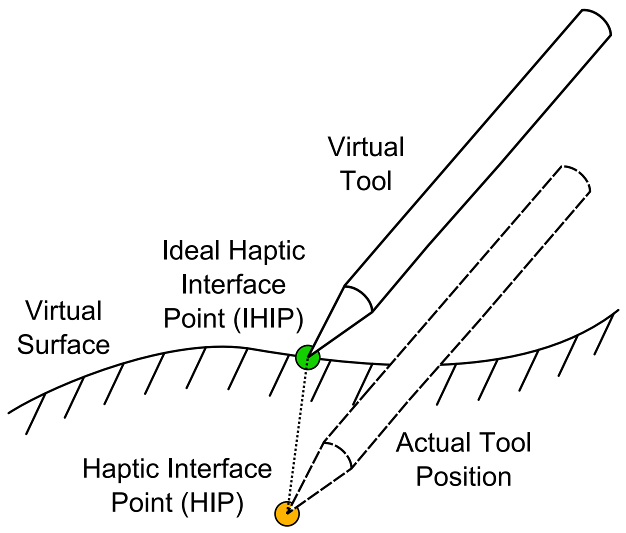


Figure 2.5: Experimental platform for this thesis, consisting of an impedance type haptic device

human-in-the-loop haptic simulators. The haptic renderer operates at a sampling rate on the order of 1 kHz to reproduce fine texture effects and contact dynamics that contain high frequency information, while the physics engine can operate more slowly at around the 30-100 Hz range [14, 42]. The graphical rendering runs at rates on the order of 30-60 Hz to generate smooth visualizations. Where the physics engine deals with the full complexity of the virtual world, the haptic renderer operates on a simplified local contact model to calculate the ideal position of the virtual tool as well as the force feedback to be delivered to the operator [13]. This has led to a multi-threaded, multi-rate architecture for haptic simulators to bridge the high-update rate of the robotic controller necessary to produce high quality force feedback and the greater computational demands of the physics engine [43].



Virtual Tool
Ideal Haptic Interface Point (IHIP)
Virtual Surface
Actual Tool Position
Haptic Interface Point (HIP)

Figure 2.6: Conceptual representation of the virtual world showing penetration of virtual tool into model

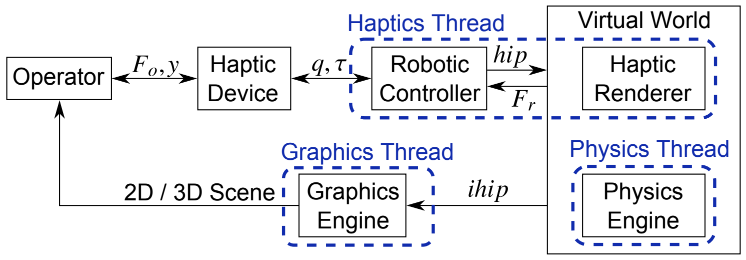


Figure 2.7: Block diagram for impedance type haptic simulators

2.3 Haptic Thread

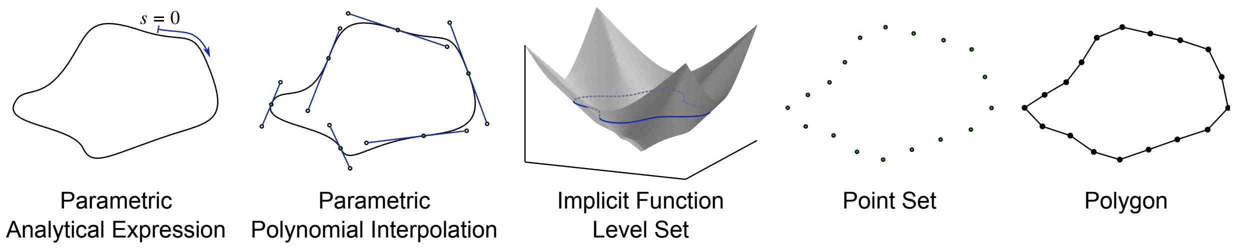
The haptics thread includes the haptic renderer in addition to the robotic controller, as depicted in the block diagram of Figure 2.7, but does not contain the physics simulation. The haptic renderer calculates the position of the virtual tool given the position of the robot's tool handle and the model's geometry, as was shown in Figure 2.6. Given the discrepancy between the two tool positions, the haptic renderer calculates the response force to minimize the position error and generate realistic force feedback.

Early haptic renderers were unable to treat the full model geometry at robotic update rates and so relied on intermediate contact models that would be updated periodically in the physics thread's simulation [44]. Later haptic works benefited from faster computers and more computationally efficient data structures tailored to the haptic rendering problem so that the contact location could be updated at robotic rates even on relatively complex rigid models [45]. The simplified local contacts models are typically formed by the intersection of one or more constraint planes that confine the virtual tool to a plane, line or point on the model's surface [46].

2.4 Physics Thread

The virtual world, also called the virtual environment, of the haptic simulator is a computer based simulation of a defined set of physical phenomena. It contains a set of models representing objects with which the operator can interact. The virtual worlds considered in this thesis are comprised of a collection of rigid unmovable frictionless models. In this simplified case, the sole responsibility of the physics engine is to detect collisions between the virtual tool and the models. Collision detection would ideally be included in the robotic controller loop, but the high update rate required can make this a challenging task. The runtime for collision detection algorithms is dependent on multiple factors including the capabilities of the computing hardware, the complexity of the virtual world and the representation of the model. The form of the model also has a significant impact on the form of the haptic renderer so a brief overview of model representations is conducted here.

There are a variety of model representations that have been used to construct the virtual worlds of haptic simulators, shown for two dimensions in Figure 2.8. This diversity arises from the way that the models are created as well as the different application areas that have influenced the field of haptics. Some simple shapes can be defined analytically by parametric curves, which will be explored in depth in Chapter 5. Parametric models are common in the field of Computer Assisted Design (CAD), where surfaces are formulated as one or more parametric patches. They are constructed from a set of control points coupled with an interpolation scheme, e.g. Nonuniform Rational B-Splines (NURBS) [47], that can form both smooth and angular surfaces. Implicit models define the model's surface as a level set $\{y \in \mathbb{R}^N | f(y) = 0\}$, often combining simple shapes using constructive solid geometry to produce more complex models [48]. Point set models represent objects through an unstructured collection of surface points obtained through e.g. laser scanners or Light Detection And Ranging (LIDAR) systems [49].



$s = 0$
Parametric Analytical Expression
Parametric Polynomial Interpolation
Implicit Function Level Set
Point Set
Polygon

Figure 2.8: Most common model representations

Polygon models, the most common representation in haptics, consist of a series of vertices connected by edges to form faces that combine to cover the surface of the models in the virtual environment. These models are particularly popular among computer graphics and finite element modelling communities, where the models are often constructed by designers in software rather than generated from acquired data. The structured relationships between the vertices, edges and faces are used to define data structures for storing the model and the algorithms that act on them [45]. Polygon meshes can exactly represent objects that are comprised of flat faces, but can only approximate curved surfaces. The memory requirements of the model is related to the number of polygons in the mesh, but is independent of the physical dimensions of the object. To accurately capture regions of high curvature, many small triangles may be required, which increases the storage requirements for the model as well as the processing time of algorithms that act on the model. The collision detection schemes for polygon models often employ a hierarchical spatial partitioning scheme to avoid performing collision detection on each individual triangle in a mesh, typically leading to a computational complexity of $O(\log n)$ for a model with n faces [13]. The run-time for the algorithms is dependent on whether there is a collision or not and where in the partitioning tree the contacted primitive is located, giving the algorithms a range of possible run-times from a few microseconds to a few milliseconds [13]. Multiple triangular meshes of different resolutions representing the same model can be used together in a multi-resolution scheme to provide different levels of detail. This allows the algorithms to undergo a loss of accuracy in order to maintain a desired time of execution, sacrificing absolute accuracy to adhere to timing constraints [50].

Another class of representation are volumetric models that divide the virtual world into voxels, the elementary unit of volume in a discretized space. Data is stored either for each voxel indicating whether or not the voxel is occupied or for each vertex where the value of a function is recorded. Datasets of this form arises from Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) acquisitions. The size of the dataset depends on the size of the voxel and the total volume being imaged, but is independent of the geometry of the object being scanned. Volumetric models are dense models in the sense that voxel values are stored for all points in the model space, whether they are occupied or not. Collision detection for a point type object and a volumetric dataset consists of indexing the tool's position into the object's dataset and checking whether a given voxel is occupied. A comparison between polygon and voxel representations of the same model were conducted in [51], showing that the voxel methods required 486 KFLOPS (floating point operations) while the polygon methods required 10.38 MFLOPS to render the same model to a similar degree of accuracy. The simplicity of collision detection on volumetric models affords a significant computational advantage over polygon models. The storage requirements for the volumetric and polygon models were 8MB and less than 0.1MB, respectively, demonstrating the storage savings of polygon models.

While the approaches of existing haptic rendering techniques can be applied to any model form, the actual implementation is tightly coupled to the model representation. Conversion between model forms is possible, e.g. a volumetric representation of a polygon model can be obtained by performing collision detection at every voxel and recording whether or not a collision resulted. The conversion process typically leads to loss of accuracy, whose significance depends on the particulars or the model forms and the conversion process itself.

2.5 Modelling and Control of an Impedance Haptic Device

The haptic device can be controlled in a variety of ways. The most basic method relies solely on the kinematics of the device, while more complicated dynamic models can be used to improve the system's performance. The basics of modelling and control of a robotic manipulator are presented here. For more details, the reader can consult e.g. [39] or [52].

2.5.1 Robot Modelling in the Joint Space

The robot's joint angles are measured via encoder and are denoted $q \in \mathbb{R}^N$, where N is the number of DOF (degrees of freedom) of the robot. The physically realizable values of q form a set called the joint configuration space $\mathcal{G} \subseteq \mathbb{R}^N$.

The robot dynamics express the relationship between the motion and forces / torques at the manipulator's joints. Using Lagrangian mechanics, the dynamics of systems like the manipulator in Figure 2.5 can expressed using differential equations in the form of

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \,, \tag{2.1}$$

where M is the manipulator's inertia matrix, C encompasses the centrifugal and Coriolis terms, g are the gravitational effects, τ are the joint torques [39]. The vector of torques comprises m elements, corresponding to the control inputs to the system. The systems considered are fully actuated.

Assumption 2.1. *The robot is fully actuated if $N = m$, i.e. the number of control inputs to the system is equal to the number of DOFs. The motion of each joint is affected by an actuator.*

For robotic manipulators, the joint angles q are also the generalized coordinates of the system while the joint torques τ are the generalized forces [53]. The joint torques are composed of multiple elements as

$$\tau = \tau_a + \tau_f + \tau_o \,, \tag{2.2}$$

where τ_a is applied by the actuators, τ_f is due to friction and τ_o arise from the contact between the operator and the manipulator's end-effect. The actuator torques τ_a are the control inputs to the robotic sub-system through which the haptic simulator contributes to the motion of the manipulator.

2.5.2 Robot Modelling in the Output Space

The position and motion of the robot's end-effector can be translated into the output space. The forward kinematics of the robot uses the geometry of the manipulator to calculate the position of the end-effector relative to a fixed reference point (typically at the base of the robot or with respect to its rest / home configuration) as

$$y = w(q)\,. \tag{2.3}$$

The function $w : \mathbb{R}^N \to \mathbb{R}^p$ maps the robot's joint space to its output space in Cartesian coordinates where p is the number of dimensions in the output. In general, both the position and orientation of the end-effector can be contained within y, but the orientation problem is not considered in this thesis. The forward kinematics w map the configuration space $\mathcal{G}$ onto the reachable workspace $\mathcal{R} \subseteq \mathbb{R}^p$ in the output space.

The manipulator's Jacobian J relates the robot's joint velocities to its end-effector's velocity in the output space. It is given by

$$J(q) = \frac{\partial p(q)}{\partial q} \tag{2.4}$$

and relates joint space velocities $\dot{q}$ to output space velocities $\dot{y}$ as

$$\dot{y} = J(q)\dot{q}\,, \tag{2.5}$$

The Jacobian plays another important role in relating the torques at the robot's joints and forces at its end-effector. The principle of virtual work can be used to derive the transformation from the commanded force to the commanded joint torques [39, p. 243] as

$$\tau = J^{\intercal} F\,, \tag{2.6}$$

where F is the force at the end-effector expressed in Cartesian coordinates.

There are some configurations of the robot where its capabilities are reduced. At these points in the configuration space $\mathcal{G}$, called kinematic singularities, the robot cannot move in certain directions [39]. They can arise when the robot is at the boundary of its workspace since it cannot reach beyond its workspace or when two or more joints come into alignment. These points are singular in the sense that the Jacobian losses rank and the manipulator losses a DOF at the kinematic singularities so that given a velocity $\dot{y}$ in (2.5), there may not exist a corresponding joint velocity $\dot{q}$ or vice versa. The set of non-singular points in a robot's configuration space $\mathcal{G}$ is defined as the dexterous workspace $\mathcal{Q}$ [54].

Definition 2.1. *The dexterous workspace $\mathcal{Q} \subseteq \mathbb{R}^p$ comprises the set of points*

$$\mathcal{Q} = \left\{ q \in \mathcal{G} \subseteq \mathbb{R}^N : \mathrm{rank}(J(q)) = p \right\} \tag{2.7}$$

For the Jacobian to have full rank at a configuration q, the N columns of $J(q)$ must span $\mathbb{R}^p$. Since p is determined by the application, the robot must have at least p DOFs to meet this condition, i.e. $N \geq p$. The manipulator is kinematically redundant if its number of joints is larger than the dimension of its workspace [55]. It is assumed that the manipulators in this work are not redundant, that is $N \leq p$, although the results could be extended by incorporating a redundancy resolution scheme [56]. Therefore $N = p$ and the Jacobian (2.4) is a square matrix.

Assumption 2.2. *The robot's mechanism has the same number of DOFs as outputs, i.e. $N = p$. This means that the robot is not redundant and is not under actuated.*

The relationship between joint torques and end-effector forces (2.6) will also be well defined in the dexterous workspace $\mathcal{Q}$. Since the Jacobian is a square matrix according to Assumption 2.2 and full rank on $\mathcal{Q}$ according to Definition 2.1, there exists a unique τ for each F and vice versa. The robot is therefore capable of delivering force feedback in any direction on the dexterous workspace.

2.5.3 Robot Control in Cartesian Coordinates

The robot dynamics in (2.1) are expressed according to the joint configurations, which are intimately related to the construction of the haptic device. The device's details can be abstracted away to a degree through an inner controller so that the haptic renderer can operate in Cartesian space without reference to the manipulator's construction or configuration. Such an abstraction decouples the haptic device's controller from the task related details, simplifying the design of the haptic simulator through a separation of concerns from a systems point of view. This can be accomplished in various ways, including open-loop and closed-loop control schemes. Surveys of control architectures that have been applied to haptics systems can be found in [57] and [58]. The simplest of these schemes uses an open-loop inner controller based on the forward kinematics of the device so that the robotic sub-system can be encapsulated as the "Cartesian device" in Figure 2.9, having a commanded Cartesian force F_r as its input and the Cartesian position y as its output. The IHIP corresponds to the desired output y_d.

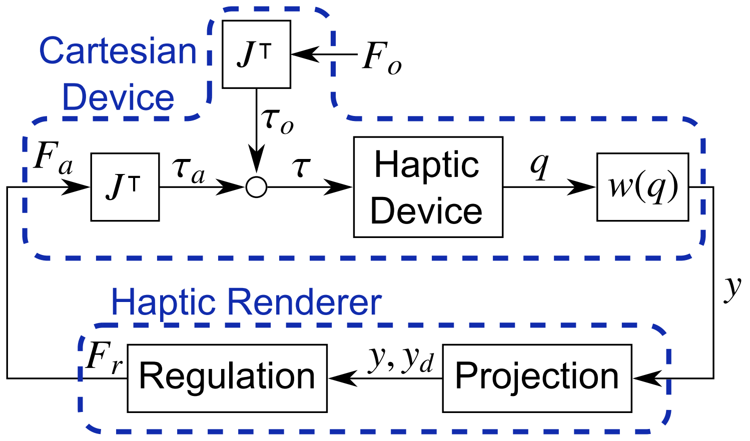


Figure 2.9: Cartesian operation of haptic devices with open-loop static model based control

The joint space dynamics in (2.1) can be expressed in Cartesian space by applying the change of coordinates provided by the forward kinematics (2.3) along with its derivative to translate the joint velocities. When the controller is chosen as

$$\tau_a = J^\top F_a, \tag{2.8}$$

where F_a is an external control input, this yields the Cartesian representation of the device

$$\hat{M}\ddot{y} + \hat{B}\dot{y} + \hat{g} = F \tag{2.9}$$

where $\hat{M} = J^{-\top} M J^{-1}$, $\hat{B} = J^{-\top} C J^{-1} - J^{-\top} M J^{-1} \dot{J} J^{-1}$ and $\hat{g} = J^{-\top} g$ are the inertia, Coriolis/centrifugal and gravity effects expressed in Cartesian space. The force at the end-effector F can be broken down into Cartesian forces analogous to the joint torques from (2.2) as

$$F = F_a + F_o + F_f \tag{2.10}$$

where F_a, F_o, and F_f are the forces due to the actuators, operator and joint friction.

The matrices in the Cartesian dynamics of (2.9) are the dynamics of the device as expressed in the output space. The static controller (2.8) is unable to compensate for the dynamics of the device, so the operator will experience the device's inertia and its Coriolis, centrifugal, friction and gravity related forces since they appear in the expression between the operator's force F_o and the device's position y. While these dynamics effects can be neglected if they are sufficiently small[1], better performance can be obtained by incorporating a dynamic model of the device into a closed-loop dynamic compensation scheme.

2.5.4 Dynamic Model Based Compensation

Aspects of the robot's dynamics that can be modelled can theoretically be canceled by the controller. Dynamic model-based compensation includes terms to cancel quantities from the left hand side of the Cartesian dynamic model (2.9), creating the inner-loop controller

$$F_a = \hat{B}\dot{y} + \hat{g} - F_f + F_r, \tag{2.11}$$

where F_r is the outer-loop's control input. This controller applies torques to counter-act the elements of the robot's dynamics, assuming that its dynamics are well known. When gravity exerts a downwards force of 3N on the end-effector, for example, the controller (2.11) will provide an upwards force of 3N so that it were as if the robot were operating in a world without gravity. The same can be said for the other dynamic effects as well. The block diagram for this control scheme is in Figure 2.10.

When the dynamic compensator (2.11) is applied to the robot governed by (2.1) and the forward position and velocity kinematics are used to transform the joint coordinates to the output space, the resulting closed-loop dynamics are

$$\hat{M}\ddot{y} = F_r + F_o \tag{2.12}$$

These dynamics show that the inertia of the robot remains unaffected under the dynamic controller (2.11) as $\hat{M}$ remains unchanged. Introducing additional terms through F_a cannot influence

[1]which is the approach taken in Sensable's OpenHaptics Toolkit for its line of PHANToM devices

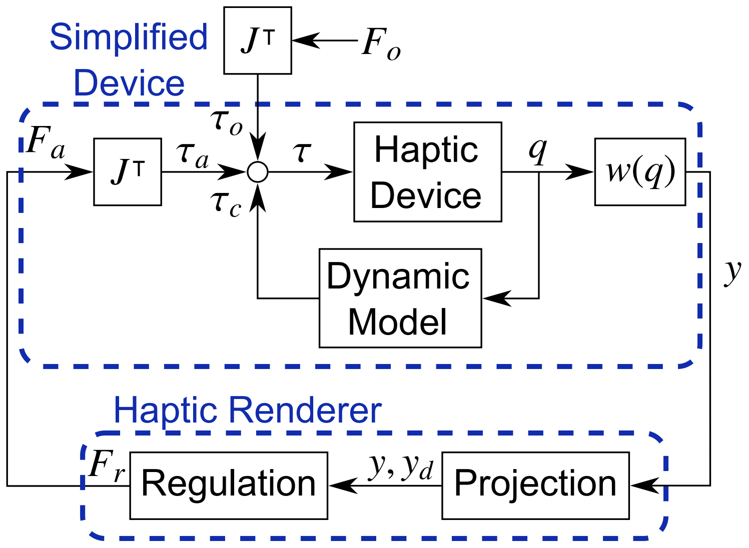


Figure 2.10: Cartesian haptic device operation with closed-loop dynamic compensation

the relationship between $\ddot{y}$ and F_o unless either one or the other of those quantities can be measurements or estimated. The compensation of the device's inertia is visited again in Section 8.2.

Ideally, the closed-loop inertia matrix would be the identity matrix multiplied by a scalar to simulate the dynamics of a point mass (the model adopted for the virtual tool in Section 3.2). If the inertia matrix $\hat{M}$ is significantly different from the virtual tool's inertia matrix, then the realism of the simulator will suffer. The inertia matrix of actual robotic devices tends to possess diagonal elements whose magnitudes vary significantly and also have off-diagonal elements. If the diagonal elements of $\hat{M}$ differ significantly, it acts as a non-uniform scaling matrix in (2.12) so that the forces and accelerations will point in different directions. This creates an anisotropic inertia, making the virtual tool feel heavier in some directions than in others [59].

The anisotropic inertia of the device can only be corrected when force or acceleration measurement signals are available for inclusion in F_a. The operator's force can then be subtracted out of the system through the controller to be reintroduced according to the desired inertia. Force signals are not always available, however, because of the expense of multi-axis force / torque sensors. Robots are often only equipped with position encoders, from which the robot's velocity is estimated by numerical differentiation and filtering. Differentiating the velocity signal to obtain accelerations can introduce too much noise to for use in feedback control [60].

2.5.5 State Space Haptic Device Models

The robot's dynamics can alternatively be expressed with a nonlinear state space model instead of a joint space model (2.1) or a Cartesian space model (2.9). The dynamics of robotic manipulators have greater structure than the general nonlinear dynamic form usually considered [61]. This model will be used extensively in Chapter 7. The dynamics of the systems in this thesis have state equations of the form

$$\begin{aligned} \dot{x} &= f(x) + g(x)u & (2.13) \\ &= \begin{bmatrix} x_v \\ f_v(x) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{N\times m} \\ g_v(x_c) \end{bmatrix} u\,, & (2.14) \end{aligned}$$

where m is the number of control inputs, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are smooth functions, and $u \in \mathbb{R}^p$ is the control input. For an N DOF 2nd order mechanical system, the state vector has $n = 2N$ elements and can be partitioned into configurations and velocities as $x = [x_c, x_v]^\intercal$, where $x_v = \frac{d}{dt}x_c$. It is often the case that $x_c = q$, so that x_c resides in the configuration space $\mathcal{G}$. In this case, the joint space model (2.1) is related to (2.14) as

$$g_v(x_c) = M^{-1}J^\intercal\,,\ f_v(x) = \quad -M^{-1}\left(C(x_c, x_v) + g(x_c)\right)\,. \tag{2.15}$$

The output of the system is

$$y = h(x)\,, \tag{2.16}$$

where $h : \mathbb{R}^n \to \mathbb{R}^p$ is a smooth function and $\frac{\partial h}{\partial x_v} = \mathbf{0}$. The output function (2.16) is the forward kinematics (2.3) expressed with respect to the state space instead of the joint configurations, so that $w(q) = h(q, \cdot)$ where $\cdot$ indicates an arbitrary velocity.

Assumptions 2.1 and 2.2 apply to the state space model as well as the joint space model, so that the number of DOFs N equals the number of outputs p equals the number of inputs m. The Jacobian of the output (2.16) with respect to the state is now an $N\times n$ matrix and has the structure

$$\begin{aligned} J(x) &= \frac{\partial h(x)}{\partial x} & (2.17) \\ &= \begin{bmatrix} \frac{\partial h(x)}{\partial x_c} & \frac{\partial h(x)}{\partial x_v} \end{bmatrix} & (2.18) \\ &= \begin{bmatrix} \frac{\partial h(x)}{\partial x_c} & \mathbf{0} \end{bmatrix} & (2.19) \end{aligned}$$

Because of the block structure of the Jacobian, the Jacobian with respect to the joint configurations $\frac{\partial h(x)}{\partial x_c}$ will often appear, which is the same as the Jacobian (2.4) from the joint space model.

The concepts of kinematic singularities and manipulator redundancies are incorporated into the vector relative degree in the state space formulation of the haptic device [62]. The relative degree of a single-input / single-output system expresses how many times the output must be differentiated for the control input to appear. The vector relative degree extends this concept to multiple-input / multiple-output systems, assembling the relative degrees of each element of the output into a vector $\{r_1, \ldots, r_N\}$.

Definition 2.2. *The system with dynamics (2.13) and output (2.16) has a vector relative degree of* $\{r_1, \ldots, r_N\}$ *at a point* $x^\star$ *if:*

1. $L_{g_j} L_f^k h_i(x) = \mathbf{0}$ *for all* $1 \le j \le N$, *for all* $1 \le i \le N$, *for all* $k < r_i - 1$ *for all* x *in a neighborhood of* $x^\star$, *where* $L_f h$ *is the Lie derivative* $\frac{\partial h}{\partial x} f(x)$. *And*

2. The $N \times N$ *matrix*

$$D(x) = \begin{bmatrix} L_{g_1} L_f^{r_1 - 1} h_1(x) & \cdots & L_{g_N} L_f^{r_1 - 1} h_1(x) \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^{r_N - 1} h_N(x) & \cdots & L_{g_N} L_f^{r_N - 1} h_N(x) \end{bmatrix} \tag{2.20}$$

is nonsingular at $x = x^\star$.

This definition of vector relative degree is adapted from [62] for the square systems considered here where $N = m$. The matrix D is sometimes referred to as the system's decoupling matrix according to role it will play in the feedback linearizations of Subsection 6.3.1.

The relative degree combines both how the state vector appears in the output y through the output function h (kinematic singularities), as well as how the control input u enters the nonlinear state space system through the matrix g to impact the state vector (degree of actuation). It is desired that the haptic device should have a vector relative degree of $\{2, .., 2\}$ so that its motion can be controlled in all of the dimensions of the output space [63]. This means that through an appropriate feedback controller, the system can be made to look like a 2nd order mass spring damper in each of its output dimensions. The vector relative degree will decrease at points in the robot's workspace where kinematic singularities arise since the robot loses a degree of freedom at those points and will not be able to move along certain directions.

Definition 2.3. *A* 2nd *order mechanical system is said to have a* ***full vector relative degree*** *if* $\{r_1, \ldots, r_N\} = \{2, .., 2\}$

The expression for the vector relative degree of mechanical systems has greater structure than that of the general form in Definition 2.2. It admits certain simplifications that will facilitate finding the singular points later on.

Proposition 2.1. *The vector relative degree of a mechanical system with dynamics (2.14) and output (2.16) will have full vector relative degree at* $x^\star$ *if*

$$\operatorname{rank}\left(\frac{\partial h}{\partial x_c} g_v\right) = N \tag{2.21}$$

at $x = x^\star$.

Proof. The proof follows closely from the results of [63]. The proposition follows from the simplifications of Definition 2.2 based on the block structure of the dynamics (2.14) and the independence of the output (2.16) on the velocity states x_v. The first element of Definition 2.2 holds true for all mechanical systems. $L_{g_j} h_i(x)$ will always be zero since

$$\begin{aligned} L_g h(x) =& \frac{\partial h(x)}{\partial x} g(x) \\ =& \begin{bmatrix} \frac{\partial h(x)}{\partial x_c} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ g_v(x_c) \end{bmatrix} \\ =& \mathbf{0}\,. \end{aligned}$$

This condition is therefore automatically satisfied for the class of systems considered here.

The second element of Definition 2.2 can be simplified under the block structure of the dynamics in (2.14). The decoupling matrix (2.20) will be nonsingular if $\operatorname{rank}(D(x)) = m$. The rank expression can be simplified as

$$\begin{aligned}
\operatorname{rank}(D(x)) =& \operatorname{rank}\left(L_g L_f h(x)\right) \\
=& \operatorname{rank}\left(L_g\left(\frac{\partial h(x)}{\partial x_c} x_v\right)\right) \\
=& \operatorname{rank}\left(\left[\begin{array}{cc} \frac{\partial}{\partial x_c}\left(\frac{\partial h(x)}{\partial x_c} x_v\right) & \frac{\partial h(x)}{\partial x_c} \end{array}\right]\left[\begin{array}{c} \mathbf{0} \\ g_v(x_c) \end{array}\right]\right) \\
=& \operatorname{rank}\left(\frac{\partial h}{\partial x_c} g_v\right), \qquad (2.22)
\end{aligned}$$

which implies that the proposition holds at those points $x = x^\star$ whenever (2.21) is satisfied. □

The condition (2.21) provides insights into whether or not the dynamics of the haptic device can be compensated for at a particular point in the manipulator's workspace and whether forces at the end-effector can be generated in arbitrary directions. While the dexterous workspace in Definition 2.1 defines a nonsingular set of points, it may not be possible to find a path from one point in the output space to another without encountering a singularity in transit. To this end, the functional configuration space and the functional workspace are defined.

Definition 2.4. *Let $x^\star = (x_c^\star, x_v^\star)$ be a point in the state space of (2.14) where $\left.\frac{\partial h}{\partial x}\right|_{x=x^\star}$ is full rank. A functional configuration space of (2.14) at $x^\star$, denoted $\mathcal{D}_{x^\star} \subseteq \mathbb{R}^N$, is the largest open, connected set containing $x_c^\star$ with the property that $\frac{\partial h}{\partial x}$ is full rank at each point in $\mathcal{D}_{x^\star}$.*

The functional configuration space $\mathcal{D}_{x^\star}$ is the set in the state space of contiguous points where the manipulator's Jacobian is well defined. The system may have multiple disconnected functional configuration spaces based on different points $x^\star$ since the robot's singularities are capable of dividing the dexterous workspace into multiple disjoint sets. The manipulator in Section 2.5.5 is an example of a system with multiple disjoint functional configuration spaces. The functional configuration space has a corresponding set in the output space called the functional workspace of the manipulator.

Definition 2.5. *Given a functional configuration space $\mathcal{D}_{x^\star}$ of (2.14) at $x^\star = (x_c^\star, x_v^\star)$, the associated set of points in the output space is called a functional workspace of (2.14) at $y^\star = h(x^\star)$*

and is given by

$$\mathcal{F}_{y^\star} = h(\mathcal{D}_{x^\star}, \boldsymbol{0}) \subseteq \mathbb{R}^p.$$

Two robotic systems are considered in this thesis: an ideal point mass robot is used for analysing and simulating the haptic renderers and the 5-bar parallelogram manipulator pictured in Figure 2.5 is used for conducting the experiments in this thesis.

Ideal Point Mass Robot Model

The point mass robot has a simple structure that will yield tractable expressions for closed-loop system analyses later in Section 6.3 and Section 7.3. Because the point robot includes only the inertial element of a dynamic mechanical system, the insights obtained using this robot highlight shortcomings in the controllers related to the rendering algorithms and the constraint geometry. It is also shown in Subsection 6.3.1 that complicated robots can be made to resemble this simplistic robot in theory through feedback linearization.

The point mass robot defines the component functions in the dynamics of (2.14) as

$$f_v^{point}(x) = \mathbf{0}_{N\times N}, \quad g_v^{point}(x) = \frac{1}{m_{point}}\mathbf{I}_{N\times N}\,, \tag{2.23}$$

corresponding to the dynamics of a point $m_{point}\ddot{y} = F$ with mass m_{point} in $\mathbb{R}^N$ under the action of the force F. The output function for the point mass robot is

$$y = h^{point}(x) = x_c\,. \tag{2.24}$$

The Jacobian with respect to the state variable for this manipulator is thus

$$J^{point}(x_c) = I\,. \tag{2.25}$$

The functional configuration space $\mathcal{D}^{point}$ consists of the entirety of $\mathbb{R}^n$ for this system, which can be readily verified since the test in Proposition 2.1 yields

$$\begin{aligned}\operatorname{rank}\left(\frac{\partial h^{point}}{\partial x_c} g_v^{point}\right) &= \operatorname{rank}\left(\frac{1}{m_{point}}\mathbf{I}_{N\times N}\right)\\ &= N\,,\end{aligned}$$

which holds for all $x \in \mathbb{R}^n$. Therefore the point robot has a well defined relative degree of $\{2,..,2\}$ for all $x \in \mathbb{R}^n$. The functional workspace $\mathcal{F}^{point}$ is then all of $\mathbb{R}^N$ due to the structure of (2.24).

Parallelogram Robot - 3 DOF Case

The manipulator from Figure 2.5 is used for the experimental work of this thesis. The robot consists of a gravity balanced 5-bar parallelogram linkage with counter-weights attached to the end of its links, as seen its schematic representation in Figure 2.11. The gravity balancing simplifies the dynamic model by nullifying the term related to gravity. The parallelogram linkage rests on a rotating base, providing three actuated degrees of freedom. The robot is direct-drive, avoiding the use of gear boxes between the motors and the attached links. This makes the robot back-drivable, that is to say easily moved by hand. The dynamics and output of the device are covered here, with more details on its implementation including its dynamic parameters in Appendix A.

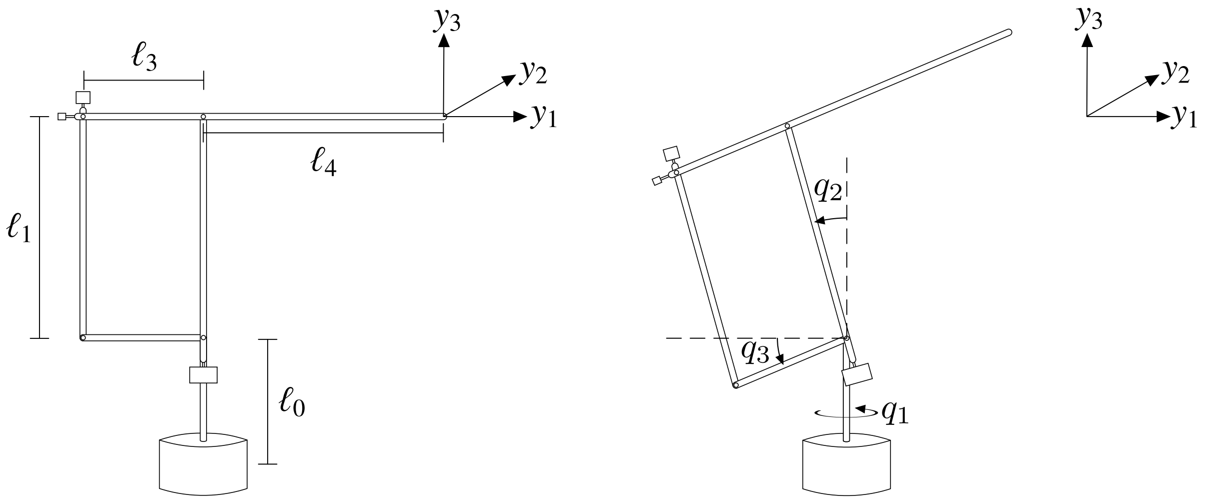


Figure 2.11: Schematic of the parallelogram manipulator in its home configuration $x_c = 0$ (left) and indicating its joint conventions (right)

The construction, dynamics and system identification for the robot were previously covered in [64, 65]. The model's dynamic form is converted into the form of (2.14), whose component functions are given by

$$f_v^{para}(x) = \begin{bmatrix} -\frac{b_1 x_4}{d_0(x)} \\ -\frac{b_2 x_5}{d_2} \\ -\frac{b_3 x_6}{d_3} \end{bmatrix}, \; g_v^{para}(x) = \begin{bmatrix} \frac{1}{d_0(x)} & 0 & 0 \\ 0 & \frac{1}{d_2} & 0 \\ 0 & 0 & \frac{1}{d_3} \end{bmatrix}, \tag{2.26}$$

where

$$d_0(x) = d_1 + d_2 \sin(x_2)^2 + d_3 \sin(x_3)^2 . \tag{2.27}$$

and b_1, b_2 and b_3 are rotational dampings, and d_1, d_2 and d_3 are positive rotational inertias.

With the conventions depicted in Figure 2.11, the forward kinematics for parallelogram robot are given by

$$y = h^{para}(x) = \begin{bmatrix} \cos(x_1)(\ell_4 \cos(x_3) - \ell_1 \sin(x_2)) - \ell_4 \\ \sin(x_1)(\ell_4 \cos(x_3) - \ell_1 \sin(x_2)) \\ \ell_4 \sin(x_3) + \ell_1 \cos(x_2) - \ell_1 \end{bmatrix}, \tag{2.28}$$

where ℓ_1 and ℓ_4 are manipulator's link lengths [64, 65]. Calculating the partial derivatives of (2.28), its Jacobian is given by

$$J^{para}(x_c) = \begin{bmatrix} \sin(x_1)(\ell_1 \sin(x_2) - \ell_4 \cos(x_3)) & -\ell_1 \cos(x_1)\cos(x_2) & -\ell_4 \cos(x_1)\sin(x_3) \\ -\cos(x_1)(\ell_1 \sin(x_2) - \ell_4 \cos(x_3)) & -\ell_1 \sin(x_1)\cos(x_2) & -\ell_4 \sin(x_1)\sin(x_3) \\ 0 & -\ell_1 \sin(x_2) & \ell_4 \cos(x_3) \end{bmatrix} \tag{2.29}$$

which is used for the static model-based compensation of (2.8) applied in Chapter 6.

The manipulator has several mechanical stops and limit switches to prevent damage. The configuration space is therefore restricted to the set

$$\mathcal{G}^{para} = \left\{ x_c \in \mathbb{R}^3 : |x_1| < 2.4, |x_2| < 2.4, -1 < x_3 < 2.3, |x_2 - x_3| \leq 1 \right\}, \tag{2.30}$$

where all of the joint angles are expressed in radians. The inequality constraint $|x_2 - x_3| \leq 1$ represents a mechanical stop that prevents the parallelogram from collapsing, which would make ℓ_1 and ℓ_3/ℓ_4 collinear.

Proposition 2.1 will hold and the manipulator will have a full relative degree at those points in $\mathcal{G}^{para}$ where

$$\operatorname{rank}\left(\frac{\partial h^{point}}{\partial x_c}\right) = N \tag{2.31}$$

since g_v^{para} is full rank for all x (notice that $d_0(x) > 0, \forall x$). The singularities therefore arise wherever the Jacobian loses rank. The determinant of the Jacobian in (2.29) is zero whenever

$$\ell_1(\sin(2x_2 - x_3) + \sin(x_3)) = \ell_4(\cos(x_2) + \cos(x_2 - 2x_3)). \tag{2.32}$$

Therefore, the determinant vanishes if and only if one or more of the following holds

1. $x_2 - x_3 = (2k+1)\pi/2, k \in \mathbb{Z}$: The parallelogram linkage is fully extended or fully collapsed and the robot loses a degree of freedom. This singularity arises when the manipulator reaches the edge of its workspace.

2. $\ell_1 \sin(x_2) = \ell_4 \cos(x_3)$: The end-effector is located on the axis of revolution of the base joint, x_1, and the robot loses a degree of freedom. This is a kinematic singularity inherent to the robot's construction.

The first condition is only encountered outside of $\mathcal{G}^{para}$ since the mechanical stops prevent the parallelogram linkage from collapsing. The second condition, however, restricts the size of the functional workspace since the singularity divides the workspace in two disconnected subsets. The functional configuration space $\mathcal{D}_0^{para}$ is the largest open subset containing the robot's home configuration $x_0 = 0$, given by

$$\mathcal{D}_0^{para} = \{x \in \mathcal{G}^{para} : \ell_1 \sin(x_2) < \ell_4 \cos(x_3)\} \ . \tag{2.33}$$

The associated functional workspace $\mathcal{F}_0^{para}$ is found by mapping $\mathcal{D}_0^{para}$ through h^{para} to get the colored region in Figure 2.12. Within $\mathcal{F}_0^{para}$, the manipulator can be adequately controlled for haptics purposes.

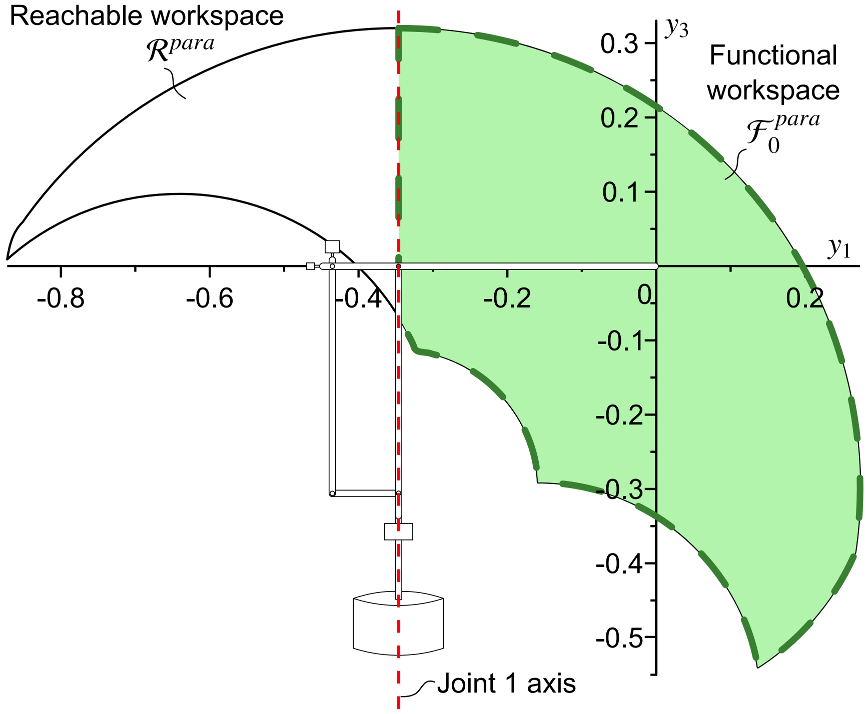


Figure 2.12: Workspaces of the parallelogram manipulator

Parallelogram Robot - 2 DOF Case

Many of the experiments investigate the controller's behaviour in a planar two dimensional scenario. In this case, the base joint of the parallelogram robot is locked so that

$$q_1 = \dot{q}_1 = 0\,. \tag{2.34}$$

A reduced model of the device is obtained by making this substitution into the state space model for the three DOF model above and removing the rows and columns that become constant. The new state vector has $n = 4$ elements, with $x_c = [q_2, q_3]^\intercal$.

The 2 DOF dynamics are obtained by removing elements from the 3 DOF dynamics (2.26) to form

$$f_v^{para2d}(x) = \begin{bmatrix} -\frac{b_2 x_3}{d_2} \\ -\frac{b_3 x_4}{d_3} \end{bmatrix},\; g_v^{para2d}(x) = \begin{bmatrix} \frac{1}{d_2} & 0 \\ 0 & \frac{1}{d_3} \end{bmatrix}. \tag{2.35}$$

With the locked condition (2.34), the second element of the output is removed to form

$$y = h^{para2d}(x) = \begin{bmatrix} \ell_4 \cos(x_2) - \ell_1 \sin(x_1) - \ell_4 \\ \ell_4 \sin(x_2) + \ell_1 \cos(x_1) - \ell_1 \end{bmatrix}. \tag{2.36}$$

The Jacobian for the output (2.36) is then

$$J^{para2d}(x_c) = \begin{bmatrix} -\ell_1 \cos(x_1) & -\ell_4 \sin(x_2) \\ -\ell_1 \sin(x_1) & \ell_4 \cos(x_2) \end{bmatrix}, \tag{2.37}$$

which is invertible for all points in the reduced order configuration space

$$\mathcal{G}^{para2d} = \left\{ x_c \in \mathbb{R}^2 : |x_1| < 2.4, -1 < x_2 < 2.3, |x_1 - x_2| \leq 1 \right\} \tag{2.38}$$

since the second singularity encountered with the three DOF model does not arise in this case. The system therefore has full vector relative degree on the entire configuration space and

$$\mathcal{D}^{para2d} = \mathcal{G}^{para2d}. \tag{2.39}$$

2.6 System Limitations

The haptic simulator will ideally maintain the position of the HIP on the surface of the virtual object through the action of its controller. There are, however, multiple factors that arise in the system's implementation that prevent this most basic of physical constraints from being maintained. Firstly, the position of the end-effector is measured through the encoders on the robotic interface. These measurements discretize its position in space so that the HIP can only occupy certain points, as depicted in Figure 2.13. Note that while the points in the diagram are evenly spaced for illustration purposes, the nonlinear forward kinematics of the robotic device and the resolution of the encoders would determine the exact placement of these points. Given that the virtual models are not necessarily aligned to this grid of points, the HIP may never resolve to a point on the surface of the model, jumping between discrete points just inside or outside of the model.

Secondly, the haptic simulator cannot respond instantaneously to changes in the physical world. Its implementation on digital hardware introduces a temporal discretization due to the

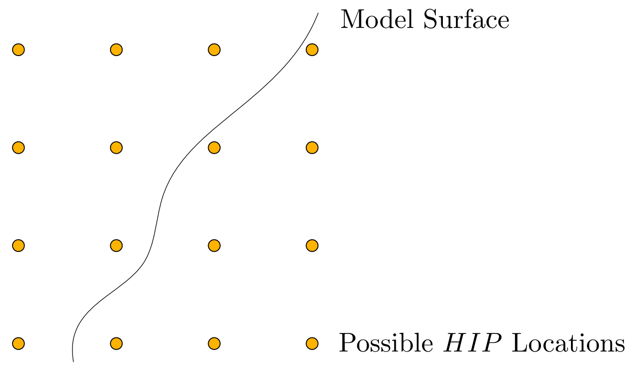


Figure 2.13: Spatially discretized position measurements and the model's surface

sample and hold operations as well as a processing delay for calculating the desired response force. During a given sampling interval the robot's position is sampled from the encoders, its forward kinematics (both position and velocity kinematics) are calculated, the haptic renderer is evaluated given this new state information, and the response force is commanded of the motors, as shown in Figure 2.14.

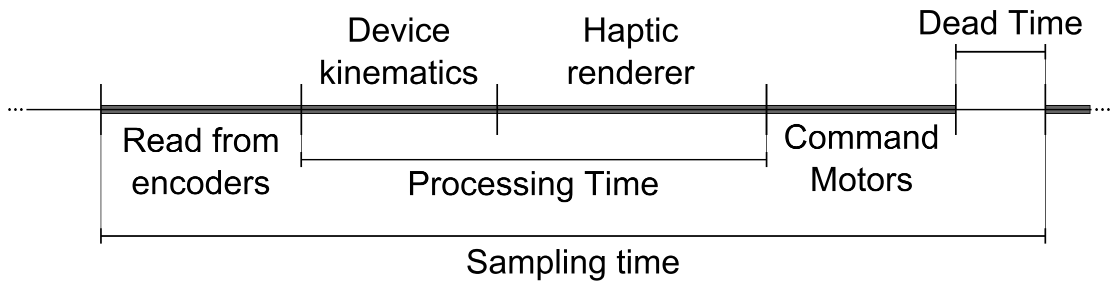


Figure 2.14: Timing diagram for the robot control loop

While reading from the encoders and outputting the response force tend to be fairly fast operations on modern data acquisition hardware, the processing time associated with the kinematic and renderer calculations introduce a non-negligible delay into the system. Delay in the control loop can have a significant impact on the system's performance, as discussed in numerous teleoperation works [66]. In between sampling instants, the HIP will travel a certain distance that can cross the model boundary, as shown in Figure 2.15, before the system has a chance to react.

These factors impact the stability of the haptic system. Instabilities or limit cycles in the

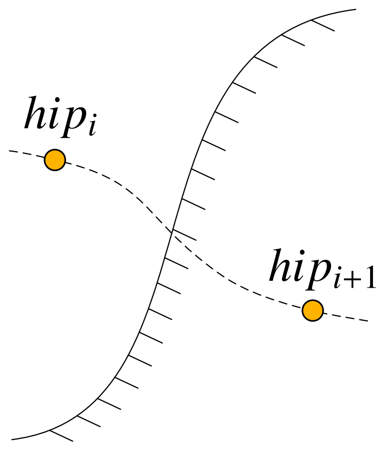


Figure 2.15: Crossing of the model surface due to the processing delay in the control loop

haptic system cause unwanted vibrations in the haptic system in the best case scenario and large, uncontrolled velocities of the manipulator creating unsafe conditions in the worst case scenario. They arise when the stiffness of the surface being rendered passes a certain threshold, which is dependent upon both the haptic renderer and the haptic device. Rigid objects are simulated using a stiff virtual spring and damper system, as opposed to the theoretical infinite stiffness of a rigid object, to maintain stable operation.

The stability of the haptic system can be difficult to ascertain because the operator is included in the system's feedback loop. Analytical models for the operator are difficult to obtain, so researchers typically rely on the existence of reasonable limits for parameters that describe the operator's actions. If the human operator is a passive element, then the haptic system will be stable if the haptic device and haptic controller are also stable [67, 68]. The maximum stiffness that can be safely rendered has been studied for sampled-data systems in a passivity context where the robotic device's dynamics are linear [67]. The stiffness of the virtual spring K is limited by

$$K < \frac{2\left(b - |B|\right)}{T} \tag{2.40}$$

where b is the natural damping of the haptic device, B is the damping of the virtual environment, and T is the sampling time [67]. The stability of the system can also be investigated using non-linear control techniques through Lyapunov function based designs [69]. The controller can be made to adapt to the nonlinear model of the robotic device and the operator's exogenous input to improve the system's performance while guaranteeing stable operation [70]. Regardless of the sophistication of the control methods used, however, there will always exist a maximum renderable stiffness imposed either by the maximum control gains of the system, the maximum output force of the actuators or the maximum force that can be tolerated by the robot's mechanism. The system's renderable stiffness can be improved by the inclusion of force sensor data to cancel the portion of the operator's force acting into the surface, as discussed further in the future research section in Subsection 8.2.3.

Because of the finite stiffness being rendered, an equilibrium point exists inside the model when the operator pushes with a constant force, as shown in Figure 2.16. If the gain K is too low, then the operator will perceive an elasticity to the surface where the model should be rigid. It is therefore important to minimize the sampling time T so that the stiffness K can be maximized.

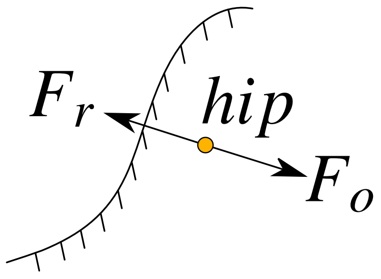


Figure 2.16: Equilibrium position of HIP inside of model

Given all of these circumstances, the simulator is unable to keep the HIP on the surface of the model. The penetration of the operator into the virtual object is an eventuality that must be addressed. It represents a state that cannot occur in the physical world, so in a sense it gives the haptic designer an element of freedom in determining how such an error should be resolved. The next chapter looks at how haptic systems are evaluated in the face of such errors and considers the physical behaviours associated with contact to define a list of qualitative behaviours that the system should exhibit.

Chapter 3

Performance Criteria and Expectations of a Haptic System

Haptic simulators are designed to emulate physical phenomena from the real world to create immersive virtual worlds. Whether or not a simulator faithfully recreates the physical world is a complicated question. The first section of this chapter considers different approaches to the evaluation of haptic simulators. While the definitive assessment of a system lies in the perceptions of the operator as determined through human factors studies, the ability of the renderer to qualitatively recreate certain physical phenomena is proposed as an efficient preliminary performance metric. The second section defines the phenomena expected from the simulator so that the designs presented in Chapters 4 and 5 can be compared along these lines in Chapter 6.

3.1 Evaluation of Haptic Simulators

The evaluation of a haptic simulator is closely related to the purpose of the system. The type of haptic systems considered in this thesis seek to faithfully mimic the behaviour of the physical world to create an immersive virtual environment using as little computational resources as possible. Translating these performance objectives into evaluation criteria for haptic simulators, though, is a challenging task. System performance is dependent on how the haptic device and the haptic renderer are perceived by the operator. The two elements are connected through a feedback loop, making it difficult to draw direct comparisons of renderers across studies that use different experimental hardware.

The haptic device itself can be evaluated as a robotic device with respect to its force output capabilities, inertia, etc. The parameters of this nature can often be quantified theoretically or experimentally for direct, objective comparison and also used as requirements in the design of new haptic devices. Several researchers have proposed different lists of performance criteria with a degree of overlap between them, which have been combined to form Table 3.1 [71, 72, 73, 74]. It should be noted that some parameters vary throughout the workspace of the robot since they are dependent on the configuration of the device, like the peak force at a given position in the workspace.

The performance of the haptic device can be measured according to its physical characteristics, but the analysis of the haptic system's performance as a whole is not as straightforward. In other robotics applications like trajectory tracking that do not involve an operator in the loop, the performance of a system can be quantified, for example, as the mean squared distance between the actual and desired trajectories. Haptic simulators, however, have no *a priori* desired trajectory so that an error signal is more difficult to define. The system is based on a feedback loop between the operator and the haptic renderer, with the desired result being a particular perception of the haptic stimuli by the operator. Haptic perception can be defined as the process of interpreting touch information [15], which looks beyond the sensory stimuli to also consider how the brain processes this information. For this reason, many evaluation techniques have focused on experiments involving human factors testing instead of analysing measured signals, e.g. [75]. These tests evaluate the system as a whole including the device, rendering algorithm and operator's sensory capabilities.

Human factors based evaluation of a haptic simulator measures the qualitative perception of a particular characteristic being simulated or the effectiveness with which the operator can perform tasks. Different rendering methods can be compared or the parameters of a specific renderer can be optimized based on the reported experience of the operator [76]. Comparison to real world interactions can be made by presenting the operator with one real and several virtual interaction scenarios, asking them to select the simulation that most closely emulates the real interaction

Table 3.1: Quantitative Criteria for Evaluating Haptic Devices

Device Aspect	**Description**
Degrees of freedom	The number of joints in the haptic device [73]
Workspace Size	The size of volume through which the end-effector can move without encountering manipulator singularities [73]
Price	The total cost of materials and labour for building the haptic device
Position resolution	The smallest distance between distinct position measurements [73]
Precision	The difference between the target and average actual positions [73]
Device-Body Interface	Shape of handle, whether the device is held or the operator is strapped into a brace, etc. [73, 71]
Structural Response	The degree to which the elasticity of the device will distort the simulation, a.k.a. the Structural Deformation Ratio [72, 71]
System Bandwidth	Frequency response of system [72, 71]
Device Z-width	Range of impedances that the device can stably render [74]
Energy flux	The power density of the device, change of energy by unit of time [72, 71]
Peak force	The largest force, measured at the end-effector in Cartesian coordinates, that the manipulator can exert [73]
Peak acceleration	The maximum acceleration of the device given the inertia of the structure and capabilities of the actuators [73]
Inertia	The perceived mass of device [73]
Damping	The ability of a device to generate forces relative to velocity [73]
Friction	The degree to which the device resists motion [72]
Sampling rate	Maximum frequency with which signals can be measured and / or commanded [72]

[75]. In the haptic training scenario, measurements of the training regimen's effectiveness can be obtained by quantifying the reduction in errors committed and in the time required to perform tasks, comparing across haptically and non-haptically trained populations [77].

There are several challenges with human-trial based evaluations. The applicability of results from human factors studies do not necessarily generalize from the particular haptic device used to haptic devices in general. The results of an experiment conducted to isolate and compare the differences between two haptic renderers could change if the experiment were conducted on another device, such as if the initial experiment used a device that filtered out high-frequency force components through its mechanical structure and then the experiments were repeated using a different device that would transmit those components to the operator.

These studies are also time consuming to conduct relative to signal analysis measures like the mean-squared error. The study must be designed, ethics approval obtained, subjects recruited, experiments conducted and results analysed. The results of these studies answer very specific research questions, and thus a series of different experiments are usually required to fully ascertain the fidelity of a given simulation. This is especially important given the interactions between different perceptions that arise in haptic illusions [78]. The various modalities must be studied independently as well as together to gather an overall model of the user in the haptic environment [79]. The taxonomy of [80], for example, lists 6 different generic tasks operator's use to perceive and interact with the virtual environments without getting into the particular motions characteristic of each mode. A large number of subjects must be recruited to obtain statistically significant results, which can be challenging for a single study but becomes even more so when a whole series of studies must be performed. This type of evaluation technique is considered out of scope for the current study.

It is therefore advantageous to develop expedient preliminary evaluation methods for use in the research and development cycle of a haptic simulator. There are certain cases where analysing the input and output signals of a haptic renderer can provide insights into the system's behaviour. When working on improving the computational complexity of an algorithm, the results from different implementations of the renderer can be compared to ensure their functional equivalence [51]. The various system's signals can also be compared for a specific renderer to quantify the effect of changing a parameter such as the model's resolution [51]. The renderer's performance can also be validated in an absolute sense by comparing the signals of a renderer to those from a real physical interaction [51]. A recorded tool trajectory in the real world can be "inverted" through the haptic renderer to extract an equivalent HIP trajectory in the virtual world that would generate the forces recorded in the real world. The interior trajectory then serves as the input to the haptic renderer, and its rendered surface trajectory and response forces can be compared against the real world signals. To do so requires the inverse of the haptic renderer, which may not exist or if it exists, may not have a unique inverse. The inverse will not exist, for example, for the minimum distance renderer when the real interaction applies large forces near a corner since pop-through, described later in Section 3.2, would occur and reduce the renderer's response force.

Another testing methodology based on signal analysis verifies that the renderer displays certain desired phenomena without exhibiting specific unwanted artifacts. Haptic artifacts are analogous to visual artifacts in computer graphics that constitute a misrepresentation of the scene being displayed. Such a qualitative analysis provides a simple means to ensure a basic level of realism, providing a set of necessary conditions for a successful simulator. This was the approach taken with the initial development of the constraint-based algorithms, where success was measured by the absence of pop-through in certain scenarios [44].

This thesis identifies a list of phenomena that a renderer should exhibit in different situations based on our expectations from the physical world. Qualitative metrics of this sort allow for comparisons across renderers in a fast and efficient way, serving as a first round of testing to see if a renderer merits further consideration in a subsequent round of time consuming user studies. While such an approach may appear like an oversimplification, it will be shown in Chapter 6 that none of the current renderers are capable of meeting all of the desired characteristics while avoiding haptic artifacts. The discussion in the next section examines the physics being simulated to identify the list of phenomena to be considered. While haptic simulators in general include simulations of deformable objects [11, 12], the current scope here is limited to rigid bodies.

3.2 Contact Between Rigid Bodies

This section outlines the physical case being considered in this thesis, including the assumptions made to define the scope of the problem and the physical behaviour that arises from the contact. It is assumed that the operator interacts with an environment comprised of rigid, immovable objects using an inflexible tool. Since the analysis of Chapter 6 deals primarily with the influence of model geometry on the performance of the renderer, the assumptions introduced in this section are intended to remove details that would obfuscate later analyses.

The tool experiences forces from contact with the surface and from the operator's grasp, as depicted in Figure 3.1. The operator's hand forms several contact patches with the tool, which are shaded in gray in the figure. Each contact patch applies a force to the tool, denoted $F_{knuckle}$, F_{middle}, F_{index}, and F_{thumb} to relate the individual forces to the portion of the hand from which they originate. The contact force exerted by the object on the tool's tip is denoted F_c.

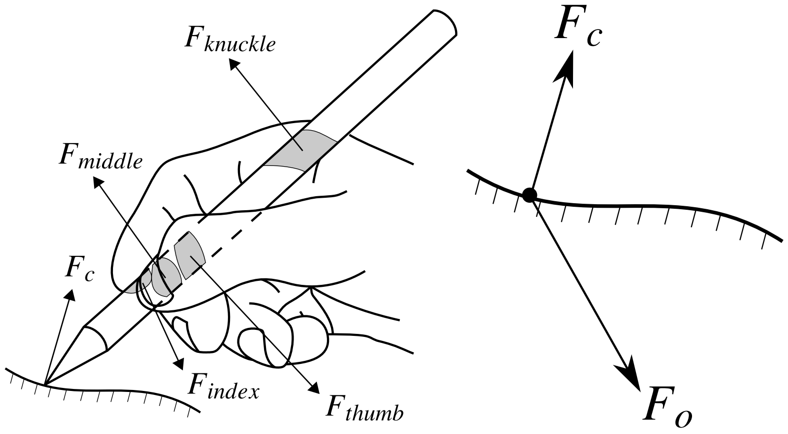


Figure 3.1: Realistic and simplified tool-based interactions with a surface

The complicated contact model on the left side of Figure 3.1 can be simplified to yield the point model on the right. The many distinct forces exerted by the operator can be summed into a single operator's force F_o. We typically maintain the orientation of our tools relative to a global coordinate frame while we work, e.g. our pens approximately maintain their orientation relative to the page as we write. Assuming that the torques induced by the contact between the object and the tool are approximately counteracted by the operator so that the tool's orientation remains fixed, its angular dynamics can be omitted from the model. It is also assumed that the operator compensates for the gravitational force on the tool. If the mass of the tool is small, then gravity will have a small impact relative to the contact forces. The tool's dynamics are then governed by

Newton's second law of motion for a point mass

$$m_{tool}\ddot{y} = F_c + F_o\,, \tag{3.1}$$

where m_{tool} is the mass of the tool, which is driven by the operator's force and the forces arising from the surface. The dynamics in (3.1) constitute the desired dynamics for the haptic simulator, which seeks to emulate the tool and the forces acting upon it that arise due to contact with the surface. The rest of this section discusses the various effects that will arise through the contact force F_c.

The tool is free to move in space or to slide along the surface of an object in the environment. The tool is assumed to have a fine tip like an awl or a probe with a small spherical tip, and makes contact at a single point with the objects in the environment. The motion of the tool along the surface is assumed to be frictionless. This does not represent a limiting assumption because friction is usually added to a basic, frictionless haptic renderer, the details of which are included in Subsection 4.3.3.

These assumptions about the contact give rise to specific physical behaviours in different interaction scenarios that are now defined. Starting with the Continuous Contact Point Phenomenon, the tool has inertia and will not therefore instantaneously jump from one point in space to another.

Continuous Contact Point Phenomenon. *The Continuous Contact Point Phenomenon is the tendency of the point of contact on an object's surface to trace a continuous path until the contact is broken.*

When a haptic renderer violates the Continuous Contact Point Phenomenon, the resulting haptic artifact is called pop-through. Pop-through occurs when the IHIP instantaneously jumps from one point on the model's surface to another distant surface point, as shown in Figure 3.2. While haptic systems are implemented as sampled data systems and it could be said that the IHIP never forms a continuous trajectory from the point of view of the controller, the magnitude of the pop-through artifact is not diminished when the system's sampling time is reduced. As the HIP approaches a continuous trajectory, distinct discontinuities in the IHIP trajectory will remain around the occurrence of pop-through.

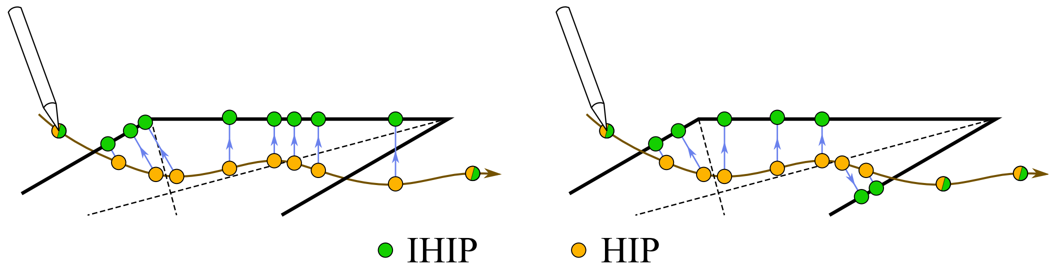


Figure 3.2: Projection of the HIP to the IHIP exhibiting the Continuous Contact Point Phenomenon (left) and undergoing pop-through (right)

The other phenomena of interest studied here derive from the nature of the contact force, which can be decomposed into components as

$$F_c = F_n + F_{imp} + F_{cen}\,, \tag{3.2}$$

where F_n is the normal force, F_{imp} is the impulsive collision force and F_{cen} is the centripetal force. Each of these forces are detailed below, presented in a manner that aligns with the evaluation of haptic renderers in Chapter 6.

3.2.1 Contact Normal Force

The normal force is a reaction to a component of the operator's force acting on the tool, preventing the operator from forcing the tool into the object. It is determined by the object's surface geometry and the operator's force on the tool. The direction of the normal force is determined by the surface's normal vector n_c on smooth regions, but an equivalent contact normal n_e is required on angular regions where the surface's normal vector is not well defined. Angular regions consist of corners or edges where two or more smooth surfaces intersect. If the intersecting surfaces are curved, they can be linearized at the point of contact and locally modeled as planes for the purposes of defining n_e. The normal force is then equal and opposite to the component of the operator's force parallel to n_e

$$F_n = -(n_e \cdot F_o)n_e\,. \tag{3.3}$$

Contact Normal Force Phenomenon. *The Contact Normal Force Phenomenon is the force equal and opposite to the portion of the operator's force in the contact normal direction.*

The Normal Force on Smooth Surface Regions

On smooth regions of the object's surface, its normal vector is well-defined. The direction of the contact force is determined by the surface's normal vector

$$n_e = n_c. \tag{3.4}$$

As seen in Figure 3.3, the equivalent contact normal n_e is simply the surface's normal vector n_c.

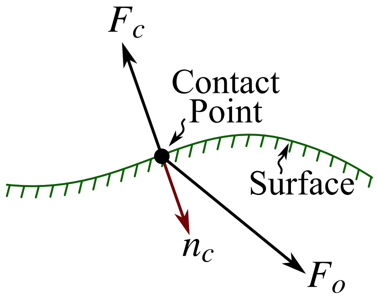


Figure 3.3: Contact on a smooth surface with the same contact and surface normals n_e and n_c

Pressley [81] gives a good account of the properties of surfaces from a differential geometry point of view and Chapter 5 will provide definitions for the tangent and normal vectors of a surface. Here it suffices to say that the normal vector is the cross product of the surface's tangent vectors, with the tangent vectors being directional derivatives that are only defined on smooth portions of the surface. On the model's convex and concave model features, the equivalent contact normal therefore cannot be defined using (3.4).

The Normal Force on Convex Angular Regions

When sliding over an edge in a dynamic interaction scenario, the contact will be broken at least momentarily as the tools momentum carries it off of the surface, as seen in Figure 3.4. Prior to reaching the corner, the tool's velocity will be tangent to the surface. Upon reaching the corner the operator's force will drive the tool towards the next face, but the momentum of the tool will keep it moving tangent to the previous face, thus breaking the contact. The normal force instantaneously disappears as the contact is broken, presenting a discontinuous force to the operator. Contact may subsequently resume, depending on F_o and the shape of the surface. The equivalent contact normal will be n_{pre} before the contact is broken, and then n_{post} if the contact is subsequently re-established on the next face.

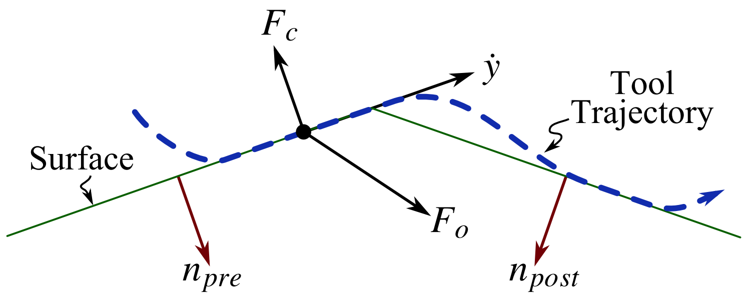


Figure 3.4: Contact normal maintaining its direction n_{pre} until breaking contact

Broken Contact Phenomenon. *The Broken Contact Phenomenon is the instantaneous disappearance of the normal force when sliding over a convex edge or corner.*

The Normal Force on Concave Angular Regions

Where the contact is broken temporarily when traversing over a convex corner, the contact can be maintained across a concave corner. Consider the tool's dynamic sliding over the surface pictured in Figure 3.5. The equivalent normal will be according to the direction of the normal vector of the model's face before the corner $n_e = n_{pre}$. As the tool enters the corner, the equivalent contact normal will transition instantaneously in space from $n_e = n_{pre}$ to $n_e = n_{post}$, although the transition may take some time. The equivalent contact normal can take on a continuous range of values between n_{pre} and n_{post} depending on the direction of F_o when the point of contact is on the corner. The equivalent contact normals for three different operators forces are shown in Figure 3.5. When the operator's force F_o^1 points into the zone under the initial face, labelled the "Pre-Corner Zone", then $n_e^1 = n_{pre}$ and the tool will start to slide to the left. When F_o^3 points under the "Post-Corner Zone", then the equivalent contact normal is $n_e^3 = n_{post}$ and sliding to the right will ensue. But when the contact force points in between the normal vectors, the contact will remain at rest in the corner in static equilibrium and the equivalent contact normal will be in the direction of the operator's force F_0^2. The equivalent contact normal is given by $n_e^2 = \frac{F_o^2}{\|F_o^2\|}$ in the "Corner Zone".

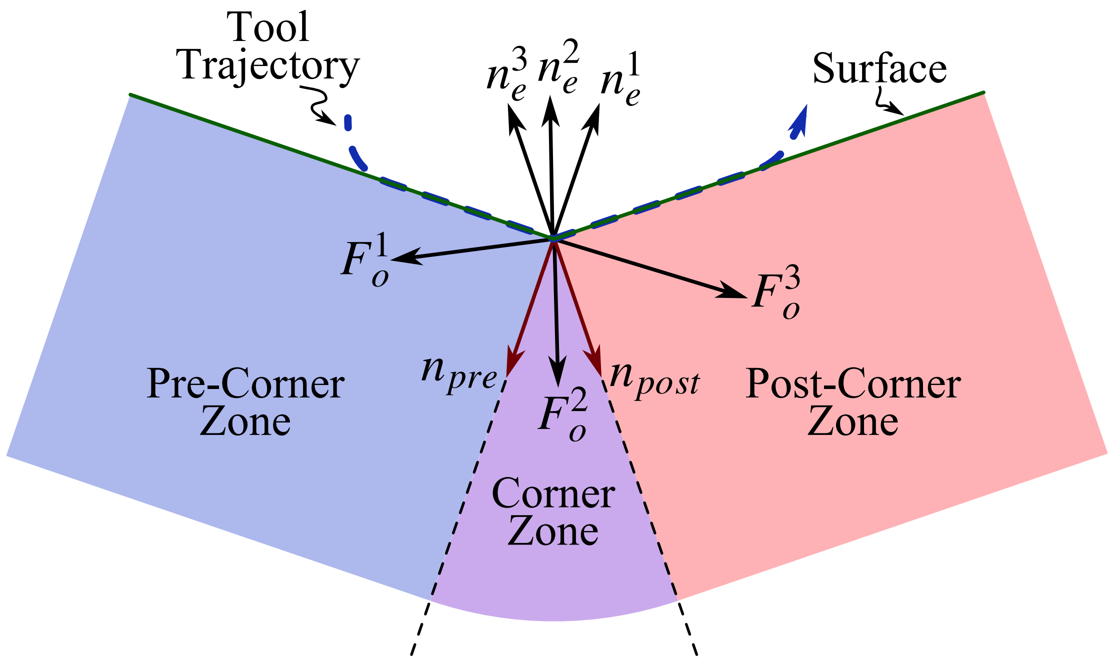


Figure 3.5: Equivalent contact normal directions n_e^i for different operator's forces F_o^i on a concave corner

The important behaviour with regards to haptic systems is that the transition of the equivalent contact normal and hence the contact force direction happens instantaneously in space. This abrupt switch in force direction, in conjunction with the impulsive collision force described shortly in Subsection 3.2.2, helps the concave corners feel crisp and sharply defined.

Abrupt Switch Phenomenon. *The Abrupt Switch Phenomenon is the instantaneous change in the contact force direction for a small motion of the contact point. It is related to the change in the equivalent contact normal direction that occurs between surfaces meeting at a concave angle.*

3.2.2 Impulsive Collision Force

When the tool initially makes contact with the surface there is an abrupt change in its momentum delivered through an impulse like force with a high magnitude but of short duration. Since the object is assumed to be fixed to the earth frame, the transfer of momentum is related to the mass and velocity of the tool as well as the nature of the collision [82]. The momentum of the tool in the direction of the contact normal will either be canceled out in the case of an inelastic collision or reversed in the case of an elastic collision. The impulsive collision force will be in the direction of the contact normal with a large magnitude and short duration.

The impulsive collision force occurs when the tool is moving in free space and encounters an object for the first time or when it is already in contact with the surface and encounters a new face on the object. When sliding into a concave corner, for example, a new face of the object is contacted and an impulsive force will be generated from this new collision. Before the collision, the tools momentum is $p_{pre} = m_{tool}v$ where v is the tool's velocity. After forming a contact with a previously uncontacted model surface, the momentum of the tool will become

$$p_{post} = m_{tool}v - (1 + c_r)m_{tool}(v \cdot n_e)n_e\,, \tag{3.5}$$

where c_r is the coefficient of restitution with $c_r = 1$ being a perfectly elastic collision and $c_r = 0$ a perfectly inelastic collision [82]. The change of momentum due to the collision is therefore

$$\Delta p = p_{post} - p_{pre} = c_r m_{tool}(v \cdot n_e)n_e\,. \tag{3.6}$$

The change in momentum is caused by the impulsive collision force acting over the duration of the collision Δt_{col}

$$F_{imp} = \frac{\Delta p}{\Delta t_{col}}\,. \tag{3.7}$$

The time period Δt_{col} for colliding rigid bodies is theoretically infinitesimally small, with a correspondingly large impulse type impulsive force F_{imp}. In practice, however, the collision occurs over a finite period of time with an associated finite contact force.

Impulsive Collision Force Phenomenon. *The Impulsive Collision Force Phenomenon arises whenever a face of the object is initially encountered, cancelling or reversing the momentum of the tool in the direction of the equivalent contact normal.*

3.2.3 Centripetal Force

While the normal contact normal force acts in a direction perpendicular to the surface, it is not the only force acting in that direction during sliding contacts. When the surface curves, a centripetal force appears so that the tool will follow the object's contours [82]. The centripetal force is

$$F_{cen} = m_{tool}\kappa\dot{y}^2 \tag{3.8}$$

where κ is the curvature of the surface. The curvature is formally defined later in Subsection 5.2.1. The faster the motion of the tool and the greater the surface's curvature, the larger the centripetal force.

Centripetal Force Phenomenon. *The Centripetal Force Phenomenon ensures that the tool follows the curvature of the object's surface and prevents interpenetration between objects. It is normal to the surface and its magnitude is proportional to the square of the velocity and to the curvature of the surface in the direction of motion.*

3.3 Summary

If a haptic simulator is capable of generating the Continuous Contact Point, Contact Normal Force, Broken Contact, Abrupt Switch, Impulsive Collision Force and Centripetal Force Phenomena in the appropriate circumstances while avoiding the pop-through artifact, it has the potential for creating a realistic simulation. These behaviours form a set of minimum requirements for a haptic simulator. They cannot completely characterise the degree of realism of a haptic simulator, but the absence of any of these behaviours in a haptic simulator will compromise its effectiveness. The next chapter looks at the properties of the existing haptic renderers to determine their performance relative to the phenomena outlined here.

Chapter 4

Haptic Rendering

The haptic renderer is the algorithm by which the position of the virtual tool is determined and the response force is calculated. The response force is calculated based on the IHIP, the HIP and the state of the virtual world including the model's geometry at the point of contact. The response force is intended to mitigate the penetration of the virtual tool into the model and to recreate the stimuli arising from real world interactions. The formulation of the haptic renderer is therefore central to the quality of the haptic simulator. The ideal haptic renderer will exhibit all of the physical behaviours outlined in Section 3.2 without suffering from the pop-through artifact.

Haptic renderers can be decomposed into two processes: projecting the HIP to the surface in defining the IHIP and calculating the response force. Finding the IHIP amounts to projecting the HIP from the model's interior to its surface, defining a desired position for the haptic device's end-effector. The response force calculation then acts on the position and velocity error between the HIP and IHIP to find a suitable force to present to the operator via the haptic device. The implementations of most haptic renderers are constructed explicitly along this dichotomous structure, but other methods can be reformulated into an equivalent form according to the projection / regulation framework. For example, [83] scales the gradient of an implicit model's signed distance function to calculate the response force, which is equivalent to projecting to the closest point on the surface and then applying a proportional controller.

The details of the haptic device are assumed to have been abstracted away to a degree through the use of an inner-loop robotic controller, such as those mentioned in Sections 2.5.3 or 2.5.4, so that the renderers deal only with positions and forces in the output space. While this separation of design aspects simplifies the construction of haptic simulators, the characteristics of the target device still affects the performance of the system. The device's performance measures from Table 3.1 list a myriad of ways that the device could negatively impact the performance of the haptic simulator. The scope of this chapter considers the performance of the haptic renderer, temporarily ignoring the influence of the device an operator on the system.

Existing haptic renderers that address the rigid point-contact problem can be grouped into two categories according to their projection method: minimum distance methods and constraint methods. Within each group, individual works provide implementations for different model representations or propose more computationally efficient solutions. The details of these two classes of projections are considered first in Section 4.1. The commonly used regulation strategies in haptics are then considered in Section 4.2. A brief summary of additional haptic effects frequently added to a basic renderer to simulate more complex behaviours follows in Section 4.3.

4.1 Projection Functions

The projection function sends points from the model's interior to its surface. The interior points represent possible HIPs, and the surface point determines a position for the IHIP that does not penetrate into the model. The function can be thought of as a vector field, which facilitates its analysis. At each possible HIP, there is a vector pointing to its associated IHIP on the surface. The collection of projection vectors over the interior of the model can be studied to determine the properties of the projection function. The continuity of the vector field or lack thereof has a major impact on the performance of the renderer in light of the Continuous Contact Point Phenomenon, so the causes of discontinuities are of primary importance. It will be shown how certain features of the model's geometry will lead to singularities in the projection function's vector field under both the minimum distance and constraint renderers. Many of the properties studied are scale invariant and will occur whether the model is microscopic or macroscopic. The model's size only becomes a factor when studying the haptic system as a whole including the operator and device, which will follow in Chapter 6.

4.1.1 Test Models

The models in a haptic simulator represent physical objects in 3D, but the behaviour of the haptic renderer is more clearly visualized in 2D. The analysis carried out here considers the haptic projection function within a planar cross section of a model, which means that the results of Chapters 4, 5 and 6 are only directly applicable to 3D models constructed by extrusion as in Figure 4.1. Extruded models have a constant cross-section so that the motion can be separated into a component without curvature along the model's axis of extrusion and a component in a 2D cross section containing the progenitor curve. This modelling technique is valid when the output remains away from the end caps of the model where the rendering of its edges requires a full 3D treatment. The generalization to the full 3D case involves considering the intersection of multiple faces instead of just two [46]. This type of extension makes the presentation much more cumbersome, but does not change the fundamental nature of the conclusions drawn from this analysis.

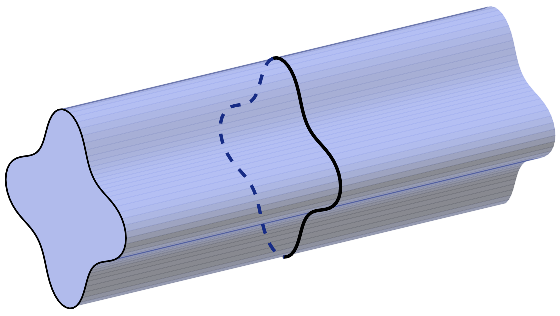


Figure 4.1: Model formed by extrusion with its progenitor curve in black

A set of test trajectories across specific model features that would ideally generate the physical phenomena listed in Section 3.2 are required to evaluate the renderers. To this effect, an angular star model and a smooth Cassinian model are chosen so that the rendering of convex corners, concave corners and curves can be assessed. These models are the progenitor curves in Figure 4.1. While the models have exact parametric representations, the renderers operate more commonly on discretized approximations to the original object. For this reason, different representations of the same models are generated. While the experimental results are only conducted on the level set model for the minimum distance method and only on the polygon model for the constraint renderer, the discussion includes their projection fields on exact representations of the model as well. Considering both exact and approximate model forms can determine which effects are due to discretization versus the character of the model's surface.

The minimum distance renderer implementation used in this thesis operates on a zero level set representation of the model. The parametric model is evenly sampled by a grid of 100×100 points that covers the model space, with the model's signed distance function being calculated at each grid point. The bilinear interpolation is used to determine the value of the signed distance function in between samples.

The constraint renderer as implemented for this thesis operates on polygon models. The model's surface is sampled using a series of vertices that are connected by straight edges. In the 3D case, the edges connect to form faces defining a surface. The polygonal representation can perfectly represent angular models, but approximates curved surfaces. Increasing the number of vertices improves the approximation so that the desired level of accuracy may be obtained.

The star is an angular model with both convex and concave corners at its vertices. The model's exact and approximate representations are shown Figure 4.2. The interior angle at its points is 60°, unless otherwise noted. Because of the angular nature of the model, the polygonal representation is an exact representation of the original parametric model. The level set representation, however, distorts the model around its corners due to the use of numerical approximations. The bilinear interpolation blunts the corners slightly, with the degree of distortion depending on the density of the sampling grid.

The Cassinian is a curve that contains regions of both positive and negative curvature, giving rise to smooth peaks and valleys. It is defined parametrically as

$$y(s) = \left(1 + \sqrt{\frac{a_n}{a_d}}\right)^{-\frac{1}{n_l}} \begin{bmatrix} \cos(s)\left(\cos(n_l s) + \sqrt{\frac{a_n}{a_d} - \sin(n_l s)^2}\right)^{\frac{1}{n_l}} \\ \sin(s)\left(\cos(n_l s) + \sqrt{\frac{a_n}{a_d} - \sin(n_l s)^2}\right)^{\frac{1}{n_l}} \end{bmatrix} \tag{4.1}$$

where $s \in [0, 2\pi)$ is the parameter, n_l is the number of lobes, and a_n and a_d influence the degree

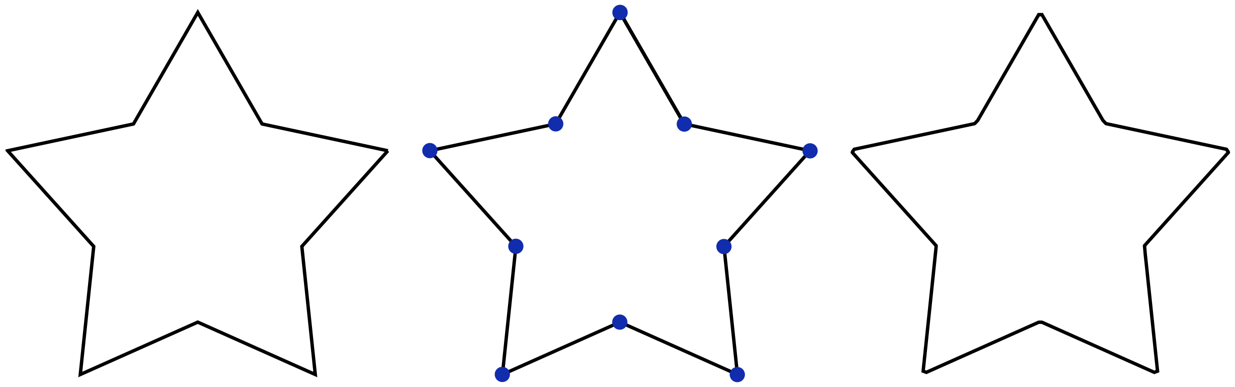


Figure 4.2: Parametric (left), polygonal (middle) and zero level set (right) representations of the Star model

to which the Cassinian is circular. Unless otherwise noted, the parameters used were $n_l = 4$, $a_n = 2$, $a_d = 1$. The various model representations for the Cassinian are shown in (4.1). The polygonal representation in the center of (4.1) uses 40 vertices to sample the surface, chosen because it resulted in sufficient smoothness of the model so that the effects of discretization were not perceived in experiment. The level set representation on the right side of (4.1) looks identical to the original model, but in fact oscillates slightly around the exact model's contours. The bilinear interpolation better represents the smooth Cassinian model than the angular star model as it averages between samples that cannot represent sharp details as readily.

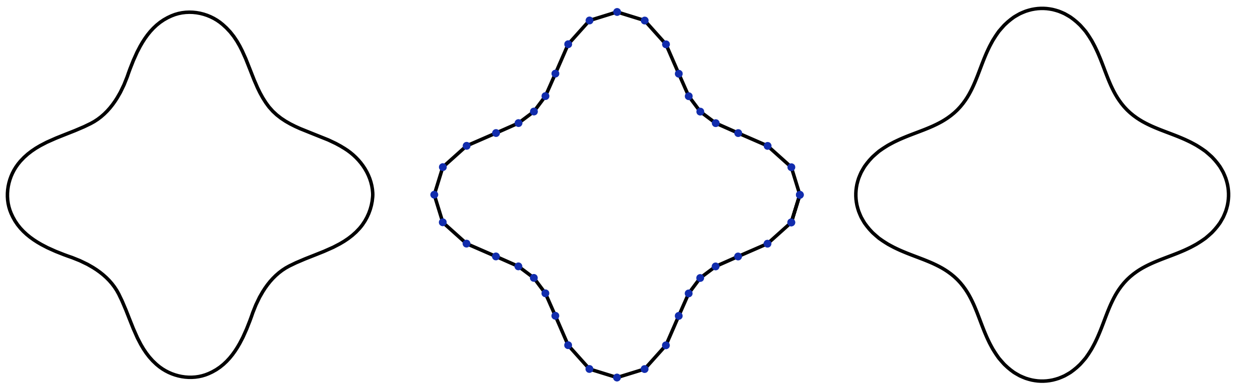


Figure 4.3: Parametric (left), polygonal (middle) and zero level set (right) representations of the Cassinian model from (4.1)

4.1.2 Minimum Distance Projection

With the minimum distance method, the HIP is mapped onto the closest point on the model's surface to determine the IHIP. Members of this family of algorithms have also been called penalty-based method [83] or vector field methods [17]. Several implementations exist for different model representations including parametric surfaces [84], implicit functions [83, 85], point clouds [49, 86], and voxel data [22, 87, 88].

The minimum distance renderers calculate the IHIP by projecting the HIP to the closest point on the surface. The minimum distance projection function is defined as

$$\rho_{md}(p) = \underset{s \in \mathcal{S}}{\arg\min} \, |p - s|_2 \,, \tag{4.2}$$

where $\mathcal{S}$ is the set of points on the model surface. The difficulty with (4.2) is that the function is not well defined for all points inside the model. In fact, there is guaranteed to be at least one point inside the model where (4.2) is undefined, corresponding to the center of a circle inscribed in the model. In most cases, there will be a broader set of points on the model interior for which ρ_{md} is multivalued, and hence not well defined. Models other than the circle will have multiple circles that can be inscribed within them that touch the surface at two or more points. The centers of all such inscribed circles combine to form the model's medial axis. The medial axis extends right to the surface at the corners of the model, as seen in Figure 4.4.

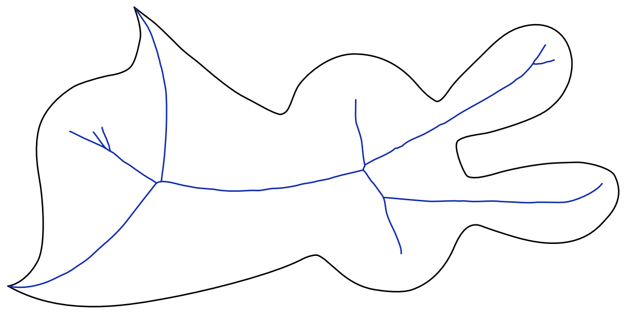


Figure 4.4: Example of a model's medial axis [5]

Blum [89] described the medial axis as the self-intersections of a wave front originating at the model's surface and moving inward normally to itself. The loci of points where the wave self-intersects forms a set of points $\mathcal{X}$ that have more than one closest point on the surface and traces out a skeleton inside the model. The plots in Figure 4.5 illustrate how points in the model are mapped onto its surface under ρ_{md}, where the blue lines indicate the direction of projection and the dashed red lines mark the set of points $\mathcal{X}$ where the projection is not well defined. The function ρ_{md} is thus well defined on the domain $\mathcal{I}-\mathcal{X}$, where $\mathcal{I}$ is the set of points inside the model. The lines of constant projection extend inwards at right angles to the surface, mapping line segments in the interior to points on the surface.

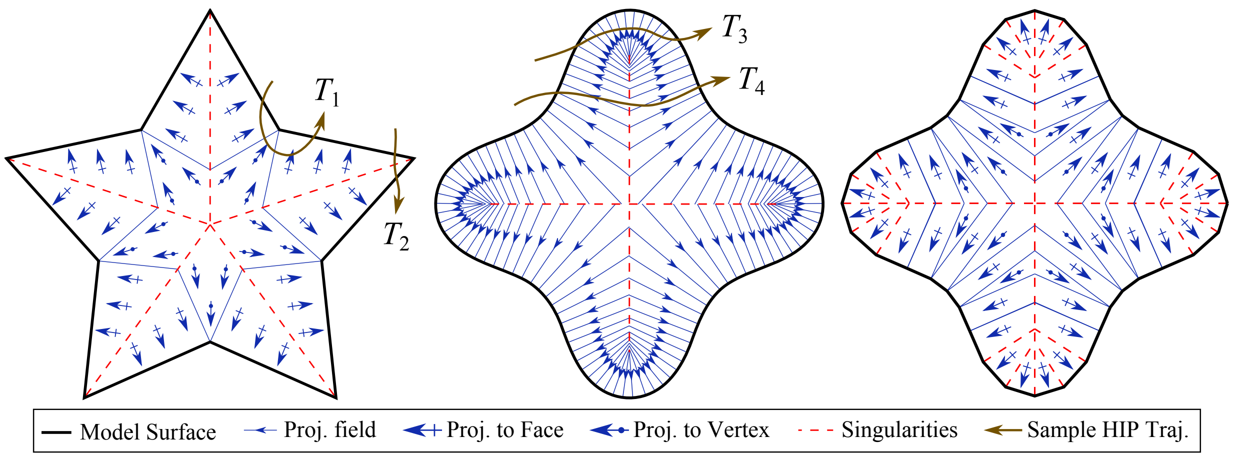


Figure 4.5: Vector field lines and discontinuities for the minimum distance projection ρ_{md} on the star (left), parametric Cassinian (middle) and polygonal Cassinian (right)

The pop-through phenomenon defined in Section 3.2 occurs when is when the HIP's trajectory crosses the model's medial axis for the minimum distance method [84]. While the HIP is far from the medial axis, the IHIP will move smoothly across the model's surface. But when the HIP crosses the medial axis, the IHIP jumps discontinuously from one region of the model's surface to a distant surface point. This transition causes the virtual tool to jump from one side of the model to another between two sample times.

Pop-through can happen on either angular or smooth models under ρ_{md}. The minimum distance projections on the star model in Figure 4.5 are smooth for the HIP trajectory T_1 since the HIP stays away from the medial axis, while T_2 crosses the medial axis and will present a discontinuity in the associated IHIP trajectory. The minimum distance projection experiences pop-through for even the smallest penetrations around convex corners because the medial axis extends all the way to the surface in those areas.

Smooth models are also subject to pop-through under ρ_{md}. For the parametric model representation, the medial axis is located at a depth of at most $1/\kappa$ from the surface and so there exists a region through which the HIP may pass without encountering singularities. Trajectory T_3 on the Cassinian model in the center of Figure 4.5 has continuous projections because the HIP's penetration into the model is sufficiently small. Trajectory T_4, however, crosses far enough into the model to encounter the medial axis and will thus experience a discontinuity in its projections.

Discretization of the Cassinian changes the model's medial axis, as can be seen from the red dashed lines on the right side of Figure 4.5. The model nows has a number of corners meeting at shallow angles so the medial axis extends all the way to the surface of the object in numerous places. While crossing any of the dashed lines in the fan of singularities within the Cassinian's lobe by the HIP will lead to pop-through, the IHIP would only jump over a small portion of the model's surface between two adjacent faces. The pop-through would become more severe when the HIP crosses through the medial axis of the original parametric model, which would result in the IHIP transitioning to a distant model face.

4.1.3 Constraint Projection

The constraint method was specifically designed to avoid the pop-through problem of the minimum distance method [44]. The constraint method is also referred to as the boundary method [17], the constraint based method [44], and the surface contact proxy method [46]. Other implementations include [90, 91, 92, 93, 94, 95]. These rendering algorithms implement the same approach to the haptic problem, focusing on making the algorithm more computationally efficient, with the exception of [47] that considers the rendering of NURBS instead of polygon models.

Where the minimum distance projection solved a global distance minimization problem to find the projection point, the constraint projection performs a local distance minimization. The algorithm is composed of an initialization and a tracking phase. A collision detection algorithm is run in the initialization phase to find the first IHIP. In subsequent update steps, the new IHIP is determined by sliding the previous IHIP along the surface towards the HIP until the position error starts to increase. For polygonal models, the IHIP will slide along the surface unless another face constrains its motion, which prevents the virtual tool from passing through the virtual object and gives the method its name.

The constraint method's projection function takes as arguments the current HIP and the previous IHIP, which introduces an element of memory into the function. The projection function is of the form

$$\begin{aligned} ihip_k &= \rho_{constr}(hip_k, ihip_{k-1}) \\ \rho_{constr} &: \left(\mathcal{I} \cup \mathcal{P}_{ihip}, \mathcal{S}\right) \to \mathcal{S}, \end{aligned} \tag{4.3}$$

where k is current time step and $\mathcal{P}_{ihip}$ is a set extending beyond the interior of the object which is dependent on the current IHIP and the model geometry. Where the minimum distance projection ρ_{md} from (4.2) was only defined on a subset of the model's interior, ρ_{constr} is well defined even beyond the model's interior.

Extending the domain of the projection function is accompanied by an increase in complexity. Where the domain of ρ_{md} was the model's interior, the domain of ρ_{constr} consists of a set extending beyond the model interior and the model's surface. The minimum distance projection was visualized as a vector field, but the constraint projection is a higher dimensional entity that is more difficult to interpret. The field can be visualized for particular past IHIPs, however. The function ρ_{constr} is piecewise in the previous IHIP, with the projection field switching abruptly as the IHIP traverses the surface. The purple bar in Figure 4.6 indicates the range of possible previous IHIPs that share this same projection field. When the IHIP crosses over to another edge on the model, the projection field will switch instantaneously. Due to the symmetry of the star, the projection field for IHIPs on other faces of the model are a rotated version of the field drawn in Figure 4.6.

The constraint projection function ρ_{constr} manages to solve the pop-through problem in some cases, such as for the interaction on the star model in Figure 4.6. The vector field is singularity free, in contrast with the minimum distance projection on the left side of Figure 4.5. This

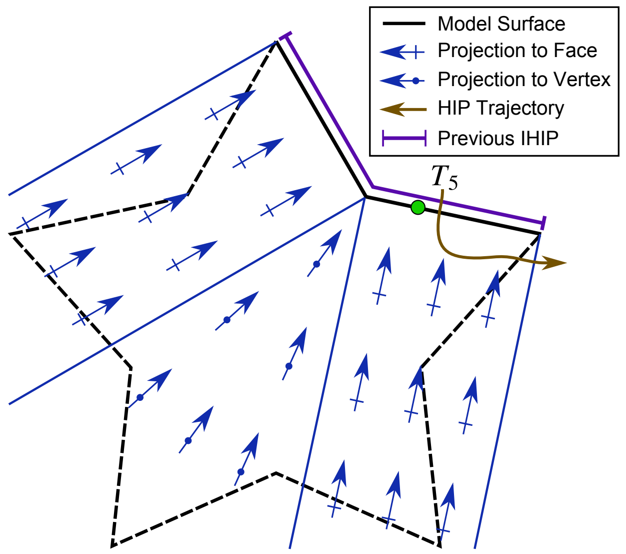


Figure 4.6: Vector field lines for the constraint projection ρ_{constr} on the star for a previous IHIP in the given range

means that the IHIP's trajectory will be continuous for any continuous HIP trajectory on this model. The trajectory T_5 in Figure 4.6, for example, will have a continuous HIP projection with the contact breaking upon sliding off of the edge of the model, which is in line with the Broken Contact Phenomenon. The projection field in Figure 4.6 also demonstrates how the vector field extends beyond the model's interior. The local constraint of the flat face means that the HIP can penetrate indefinitely into the model without sliding off of the model or transferring to another model face. This behaviour enables the constraint method to prevent the IHIP from popping-through to the opposite face on the star model.

While the memory associated with the constraint algorithm can prevent pop-through in some cases, it can still occur, especially on smooth models. Because the Cassinian is a smooth model, it would be expected that mapping a continuous HIP trajectory through ρ_{constr} would produce a continuous IHIP trajectory. The projection fields from Figure 4.7 on the Cassinian's parametric representation, however, contain discontinuities as indicated by the dashed red lines. The left side of Figure 4.7 shows field lines for the range of IHIP's indicated by the purple bar, which extends between two inflection points on the surface. While this field does contain singularities, none of them will be encountered in practice since the IHIP will transition outside of the range indicated by the purple bar before the HIP reaches a singular point. Upon transitioning outside of the range of the purple bar, the projection field will instantly switch to the field on the right side of Figure 4.7.

The singularities on the right side of Figure 4.7 can contribute to pop-through. The HIP trajectory T_6 shows a case where the IHIP would snap from one side of the model to the another between samples as the HIP crosses the dashed red line, causing pop-through. There is no edge or vertex to constraint the motion of the IHIP in the update phase, so once the HIP crosses the medial axis the proxy continues to traverse the surface to minimize the HIP / IHIP distance until reaching the opposite side of the Cassinian's lobe. This case is identical to the trajectory T_4 from the minimum distance case in Figure 4.5. If the trajectory T_6 were made shallow enough, the discontinuity, and thus the pop-through, would again be avoided.

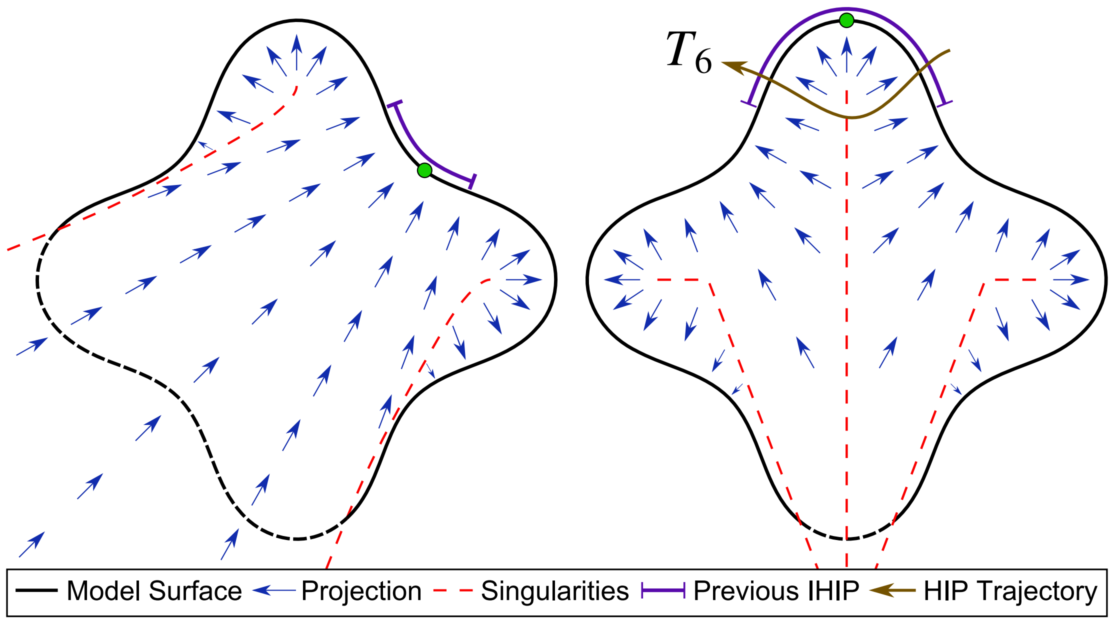


Figure 4.7: Vector field for the constraint projection ρ_{constr} on the Cassinian with singularities that cannot be encountered by the operator (left) and that can result in pop-through (right)

The Cassinian is still subject to pop-through when discretized for the constraint method, although the form of the projection function and hence location of the discontinuities changes. The projection field for the discretized model in Figure 4.8 is formed of piecewise sections that map rectangular regions onto the model's faces and triangular regions on the model's vertices. As with the minimum distance projection, line segments in the interior are mapped onto surface points. The HIP trajectories T_7 and T_8 in Figure 4.8 both experience pop-through, although to different degrees. Trajectory T_7's IHIP will undergo a jump that skips over a small portion of the edge adjacent to the previous IHIP and would likely go unnoticed by the operator, while T_8's IHIP trajectory transitions to the opposite side of the Cassinian's lobe. The pop-through scenario shown in T_8 is significant because it would be rendered in the same manner as a corner on the model and thus distort the perceived nature of the smooth model. The severity of this effect will be explored experimentally in Subsection 6.1.3.

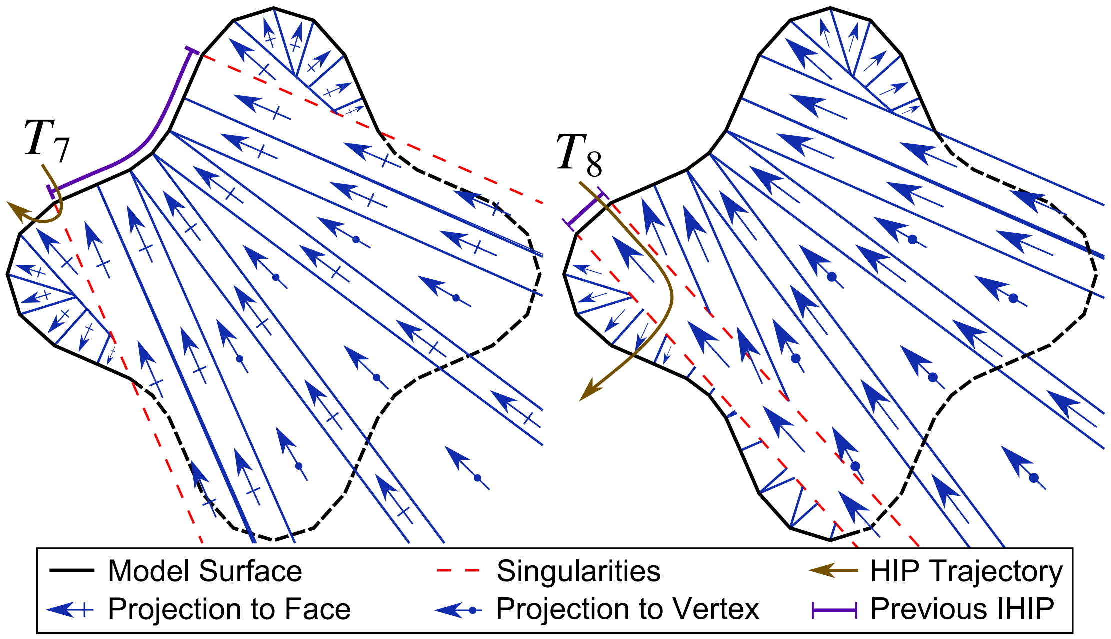


Figure 4.8: Vector field lines for the constraint projection ρ_{constr} on a sampled model for various given previous IHIPs.

4.2 Regulation Schemes

The regulation scheme uses the HIP, IHIP and the model's geometry to determine an appropriate response force to be delivered to the operator. The projection function is used to define a desired position for the haptic device's end-effector

$$y_d = \rho(y)\,. \tag{4.4}$$

The difference between the desired and actual end-effector positions is then defined as the position error

$$\begin{aligned} e_p &= hip - ihip \\ &= y - \rho(y)\,. \end{aligned} \tag{4.5}$$

The response force is typically calculated using a PD controller

$$F_r = -K_p e_p - K_d e_v\,, \tag{4.6}$$

where e_v is the velocity error and K_p, K_d are constant control gains. In other control applications, integral control is often added to eliminate steady-state error. In the haptic scenario, however, accumulating error in one direction is inappropriate since the position error vector changes direction as the operator traverses different portions of the model. This could skew the response force's direction and distort the model's geometry.

The PD controller is for the outer-loop controller that acts in Cartesian space, mapping position errors to force commands in the output space. The inner-loop controllers translating the commanded response force to joint space were discussed in Section 2.5. It is the definition of the velocity error e_v that distinguishes the various regulation schemes commonly encountered. Four definitions from the literature are summarized in Table 4.1: Projected Proportional (PP) [46, 45], Projected Proportional with Viscous Damping (PP+V) [83], Projected Proportional and Derivative (PP+D) [44], and Projected Proportional with Projected Derivative (PP+PD) [96].

The proportional only controller PP represents a spring connecting the HIP to the IHIP. Viscous damping is sometimes added to the manipulator's motion to form the PP+V controller in an attempt to improve the stability of the system by resisting the motion of the HIP. Alternatively, the traditional form of PD controller is defined by the PP+D controller that uses the derivative of the position error instead of the velocity of the manipulator as the velocity error, which captures the relative motion between the HIP and IHIP. Finally, planar approximations to the surface

Table 4.1: Regulation schemes used for haptic rendering

Controller	**Velocity Error Definition**	**Regulation Scheme**
Projected Proportional (PP)	$e_v = 0$ (4.7)	$F_r = -K_p e_p$ (4.8)
Projected Proportional with Viscous Damping (PP+V)	$e_v = \dot{y}$ (4.9)	$F_r = -K_p e_p - K_d \dot{y}$ (4.10)
Projected Proportional with Derivative (PP+D)	$e_v = \dot{e}_p$ (4.11)	$F_r = -K_p e_p - K_d \dot{e}_p$ (4.12)
Projected Proportional with Projected Derivative (PP+PD)	$e_v = (n_e \cdot \dot{y}) n_e$ (4.13)	$F_r = -K_p e_p - K_d (n_e \cdot \dot{y}) n_e$ (4.14)

are incorporated the PP+PD controller, which projects the velocity error onto the surface's normal vector to only consider velocity errors transversal to the surface. A full evaluation of these regulation schemes is conducted in Section 6.3.

In addition to the linear regulation schemes listed in Table 4.1, the use of nonlinear schemes has also been reported. To calculate the response force, [47] used $F_r = k_p e_p^{1/2} + k_d e_p^{1/2} \dot{e}_p^{1/2}$. The nonlinear spring and damping profiles increases the stiffness of the surface for small penetrations, while avoiding actuator saturation by growing the force response more slowly than a linear function for large errors.

4.3 Additional Haptic Effects

While the regulation scheme defined by the PD controller (4.6) is almost ubiquitous as the basis for haptic renderers, several additions are commonly included to introduce haptic effects less easily captured by the model's geometry. The model's shape can be manipulated through force shading or texture mapping, while the dynamics on the surface can be more fully defined by introducing friction models. The perception of the surface's hardness can also be improved by generating contact impulses. These effects are presented here to show how a basic haptic renderer is extended to include advanced functionality, but their effects are not evaluated to maintain the focus on the physical phenomena outlined in Section 3.2.

4.3.1 Force Shading

Force shading techniques modulate the direction of the response force based on an interpolation of normal vectors stored at each model vertex [97]. The response force is therefore aligned not with the model's physical geometry, but according to meta-data stored with the model. Force shading is intended to smooth out the rendering of coarse polygon models by continuously varying the contact normal vector according to the contact location.

The technique requires the calculation and storage of a normal vector at each vertex. To smooth an angular surface, numerical approximations to the normal vector for a vertex can be calculated using the position of the surrounding vertices. If a model has both smooth and angular regions, care must be taken when calculating the vertex normals to ensure that angular regions do not undergo rounding as well. Given an IHIP with an associated set of barycentric coordinates a_i, the force shaded normal is calculated as the weighted average of the adjacent vertex normals

$$n_e = \sum a_i n_i \tag{4.15}$$

where n_i are the normal vectors associated with the i^{th} vertex.

4.3.2 Texture

The basic PD regulation scheme in (4.6) has often been augmented to emulate the effects of surface texture because capturing minute surface deviations would greatly increase the density of voxel based representations or the number of vertices, edges and faces in polygon based representations. Using a height map to vary the shape of the surface and the direction of the surface normal, similar to bump mapping in computer graphics, provides a convenient way to modulate the response force and produce the requisite vibrations [88]. Height maps can also be generated stochastically instead of "painting" the entire surface with particular bumps and scratches [98]. Instead of varying the height of the surface, a stochastic disturbance is added to the calculated response force. The coefficients of an identified auto-regressive moving-average (ARMA) model can recreate realistic vibrations to match that of given textures sampled from the physical world [99].

The general form of the contact normal perturbation takes the form

$$n'_e(s) = n_e + A(s) \tag{4.16}$$

where s is a parametric representation of the IHIP's location on the model's surface used to index into the height map or as input to a stochastic model, and the function $A : \mathbb{R}^{N-1} \to \mathbb{R}^N$ calculates the deviation of the normal from the standard model geometry. To integrate such a scheme into a haptic renderer, the modified normal n'_e would be used in place of the equivalent contact normal n_e.

4.3.3 Friction

In the case of static friction, the IHIP is kept in place until the HIP is moved far enough to overcome the limiting friction and transition to a sliding state, at which point the IHIP is updated [88, 46]. The operator readily perceives the abrupt disappearance of the static friction, although with low force output devices the operator is capable of noticing the motion of the HIP around the static point of contact before dynamic sliding starts. The friction cone algorithm is presented here [100], which assumes that the surface is flat but is sufficient to demonstrate how the technique works.

The system can either be in a static contact state where the friction maintains the position of the tool or in a dynamic sliding state when the tool starts to move. Assuming that the tool is in the static state, the IHIP is maintained at a fixed position $ihip_s$ where the tool originally contacted the surface or last came to rest on the surface, as seen on the left side of Figure 4.9. The projection of the HIP in subsequent time steps is considered the proposed IHIP $ihip_p$. The normal force as exerted by the operator is estimated using the current position error as

$$F_n \approx K_p(hip_k - ihip_p)\,. \tag{4.17}$$

Using this measure of the operator's downwards force, the upper limit on the magnitude of the static friction force is

$$|F_s|_{max} = \mu_s |F_n|\,, \tag{4.18}$$

where μ_s is the static coefficient of friction.

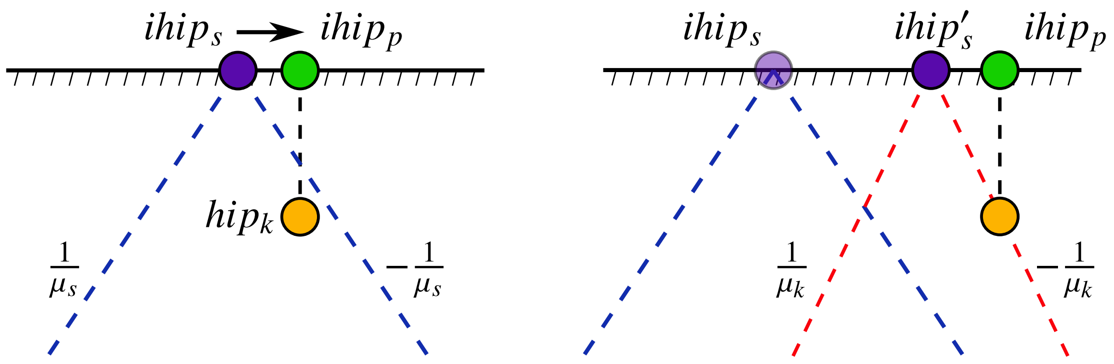


Figure 4.9: The Friction Cone Algorithm in static contact (left) and dynamic sliding (right)

The force exerted by the operator tangent to the surface is

$$F_t \approx K_p(hip_k - ihip_s) - F_n\,. \tag{4.19}$$

If the operator's tangential force is less than the static friction, $|F_t| \leq |F_s|$, then the IHIP remains fixed at $ihip_s$. Because the normal force as well as the static and kinetic friction forces are proportional to the penetration depth of the HIP, the result is that the IHIP remains in place so long as the HIP remains within the static friction cone under surface, as shown in the dashed lines on the left side of Figure 4.9.

The HIP exits the static friction cone when $|F_t| > |F_s|$ at which time the contact enters the dynamic sliding mode. The magnitude of the kinetic friction force is

$$|F_k| = \mu_k |F_n|\,, \tag{4.20}$$

where μ_k is the kinetic coefficient of friction. The kinetic friction force is directed opposite to the tangential component of the operator's motion. In the sliding contact mode, the IHIP is updated such that the HIP is located at the edge of the dynamic friction cone on the right side of Figure 4.9. This ensures that there will be the correct amount of position error in the tangential direction to generate the appropriate kinetic friction.

4.3.4 Contact Impulses

The impulsive forces due to collisions seen in Subsection 3.2.2 can be added to the rendered force as feed-forward pulses upon initial contact to cancel the momentum of the device. The

The expression for the momentum of a robotic manipulator is given in [101]. Adapted to the notation of this thesis, the change in momentum of a manipulator in response to a contact event is

$$\Delta p = -(1+c_r)\left(JM^{-1}J^{\mathsf{T}}\right)^{-1} J\dot{q}_0\,, \tag{4.21}$$

where $\dot{q}_0$ is the initial velocity of the manipulator in the generalized coordinates and c_r is the coefficient of restitution of the collision. Δp represents a change in momentum between the manipulator's pre- and post-collision states that can be supplied by the appropriate joint torques over a short period of time to better simulate the Impulsive Collision Force Phenomenon.

An alternative approach uses pre-recorded impulses that are re-played in an open-loop fashion during contact events to improve the perception of surface stiffness [102]. The pulses have been designed to reproduce the accelerations recorded from physical contact scenarios. The method assumes that the contact forces can be decomposed into a fast, high frequency and slow, low frequency components.

Chapter 5

The Mapping Renderer

The minimum distance and constraint projections from Chapter 4 were shown to contain discontinuities inside the model, leading to the pop-through artifact when they are incorporated into a haptic renderer. With the previous projections, line segments in the model were mapped onto points on its surface. The undesired discontinuities arise where the interior lines of constant projection meet. If the lines of constant projection can be curved such that they meet at only a single point in the model interior, the possibility of pop-through will have been minimized.

This chapter considers an alternate projection function that solves the projection problem by mapping the model onto a canonical model like a circle. The problem is trivial on the circle and singular only at its center. The mapping is formed by generating a coordinate system on the model that corresponds to a similar coordinate system on the canonical model. The model is flowed under the action of a differential equation so that the coordinate lines are based on the model's geometry. Because the resulting projection field consists of curved lines of constant projection, a regulation scheme is proposed to ensure that the response force is oriented in the proper direction.

5.1 The Projection Problem

The projection problem can be difficult to define and computationally intensive on objects of arbitrary shape, but is straightforward on a circle or sphere. The circle's interior points can be projected radially towards the surface, which is well defined everywhere except at the circle's center. If a transformation function can be used to map the model onto a circle, then the projection problem can be solved in a domain where the problem is trivial. The models considered are homeomorphic to the circle, that is models for which there exists a continuous bijection that maps the model onto the circle and that possess a continuous inverse for mapping the circle back onto the model. Based on the discussion from Chapter 4, the ideal projection would possessed the following properties:

Be continuous at all but one interior point

Since pop-through occurs across discontinuities in the projection function, ensuring that there is only one discontinuity minimizes the potential of pop-through occurring.

Be C^2 continuous

The projection field will eventually be incorporated into a control loop involving a second order system. While C^2 is not required for the projection to work smoothly with PD controllers, the future work envisions the incorporation of the projection function into more complex controllers.

Be a function of the current HIP only

If the projection is a function of the HIP alone, as opposed to the constraint projection that also relies on the past IHIP, the projection can be precomputed offline and stored in a look-up table for fast evaluation online.

Result in an error vector perpendicular to the surface for shallow penetrations

The position error vector between the HIP and IHIP should be parallel to the contact normal for small penetrations to preserve the character of the model so that, in the limit, the rendering is driven by the model's geometry.

One source for transformation functions capable of mapping one model onto another are the methods from image registration that determine a correspondence between a source and target image. Image registration is the determination of a geometric transformation that aligns the points of one image with another [103]. The techniques can be classified according to the formulation of the registration problem. Feature-based techniques match distinct points or curves in one image to the next while intensity-based techniques match entire images onto one another.

Since the haptics problem deals purely with the geometric aspects of a model, feature-based techniques are the most appropriate image registration methods. Many techniques in this area model the space as a rubber sheet or a thin plate, calculating the displacements between features according to the physical laws governing elastic solids. The deformations are constructed such that the image features align and such that the bending energy of the system between features is minimized. Linear elastic models are sufficient to accurately model modest changes in shape between the source and target, but have difficulty capturing more significant deformations [104]. To accommodate large deformations and ensure that the transformation is a diffeomorphism, other methods are modelled on viscous fluid flows as opposed to deformable solids [105]. Both the elastic and fluid flow methods generate a transformation function by imposing a physical behaviour on the deformation. Given the importance of the model's geometry in the calculation of the contact force in Chapter 4, methods based on differential geometric are considered instead.

5.2 Differential Geometry and Grid Generation

The problem of finding a transformation function can equivalently be approached as the drawing of coordinate lines on the model. Since the models considered are homeomorphic to the circle, a coordinate system with the same structure as the polar coordinate grid can be established on the model. This type of approach has been taken in differential geometry, where curves are evolved according to their normal vectors and curvatures to generate a series of levels that can be used in gridding applications. This is in contrast to most image registration methods that assume a class of geometric deformations and calculate the parameters that optimize the transformation such that an error metric is minimized.

This section considers how the surface of a model can be viewed as a curve that is acted upon by a differential equation, and how the properties of the differential equation can be tailored for haptic use. The numerical methods required to solve the differential equation are presented. Lines tangential and transversal to the model's surface are generated, forming a grid over its interior. Section 5.3 then goes on to construct a projection function using the resulting transformation function.

5.2.1 Curve Shortening Flows

The behaviour of curves as they evolve under the action of differential equations has been studied in the pure math community and has found application in a variety of areas including fluid dynamics, image enhancement, image segmentation and grid generation [106]. Some basic definitions from differential geometry are presented here to provide the necessary background behind the Curve Shortening Flow (CSF) problem. For more details, a good introductory differential geometry text is [81], while [106] provides a thorough treatment of the CSF problem in particular. A preliminary discussion on the nature of CSFs is presented for parametric curves before switching the discussion over to a level-set representation of the model that is more conducive to a stable numerical solution.

The discussion that follows makes some basic assumptions about the curves that represent the model's surface. It is assumed that the model can be represented as a Jordan curve, which is a simple, closed curve that does not intersect itself and divides the plane into two regions. The curve ϕ can be defined parametrically as the regular curve $\phi : [a,b] \to \mathbb{R}^2$ where $\phi(a) = \phi(b)$. Regularity of the curve means that the derivative of the curve's component functions are continuous, i.e. that the curve does not possess corners or cusps. While the regularity property would seem to limit the class of models to smooth shapes, the final numerical implementation yields finite approximations to the derivatives on the corners of angular models so that this framework requires only C^0 continuity of the surface in practice. It is assumed that the curve has an anticlockwise orientation, which places the interior region to the left of the curve. The orientation of a regular curve with clockwise orientation can always be inverted by a change of parameter.

The tangent and normal vectors of ϕ can be defined using its derivatives. The derivative of the curve with respect to its parameter can be expressed as

$$\frac{d\phi(s)}{ds} = v_c(s)t_c(s) \tag{5.1}$$

where $t_c(s) = \frac{d\phi(s)}{ds} / \left|\frac{d\phi(s)}{ds}\right|$ is the curve's unit tangent vector and $v_c(s) = \left|\frac{d\phi(s)}{ds}\right|$ is the velocity of the curve. The derivative of the unit tangent $t_c(s)$ can be expressed as

$$\frac{dt_c(s)}{ds} = \kappa(s)\left|\frac{d\phi(s)}{ds}\right| n_c(s) \tag{5.2}$$

where $\kappa(s)$ is the curve's signed curvature and $n_c(s)$ its inward unit normal vector. The curvature gives an indication of how quickly the normal vector changes when moving along the curve and thus how sharply the curve bends. Where the tangent line defines a first order linear approximation to the curve at a given point, the osculating circle provides a second order approximation. As well as sharing a point with the curve, it also has the same curvature at the point of approximation. It is tangent to the curve at the point $\phi(s)$ and passes through an additional pair of points on the curve infinitesimally close to $\phi(s)$. The osculating circle is centered at $\phi(s) + (1/\kappa(s))n_c(s)$ and has a radius of $1/\kappa(s)$. Sections of the curve where $\kappa > 0$ ($\kappa < 0$) correspond to convex (concave) regions on the model's surface. These conventions are illustrated in Figure 5.1.

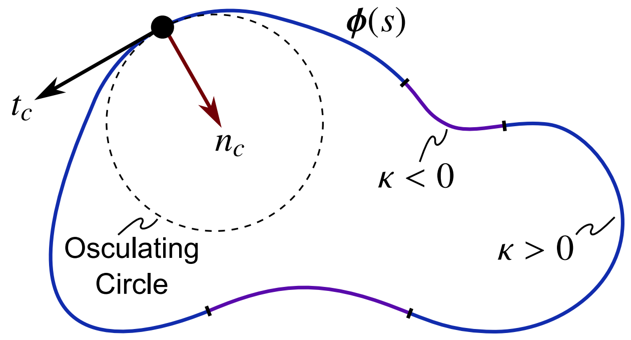


Figure 5.1: A parametric curve and its associated quantities

The CSF problem is formulated as the evolution of the curve $\phi(s)$ according to a differential equation. The curve is augmented with an artificial time variable $\phi(s,t)$, initialized such that the curve $\phi(s,0)$ is the original parametric curve $\phi(s)$. The definitions of the tangent, normal and curvature are also augmented with the artificial time t since they change along with the curve.

The CSF problem is then the initial value problem governed by

$$\frac{\partial \phi(s,t)}{\partial t} = v_f(\kappa) n_c(s,t) \tag{5.3}$$

where $\phi(s,0) = \phi(s)$ and v_f is the velocity of the flow as a function of the curvature κ. The CSF equation (5.3) represents pushing the curve ϕ along its inward normal vector n_c at a speed governed by v_f. For the original CSF problem $v_f(\kappa) = \kappa$, which is also referred to as the flow by mean curvature [107].

Some general results have emerged from the study of the flow by mean curvature. Grayson's Theorem states that the flow under mean curvature takes any closed, embedded curve and flows it to a convex curve in finite time [108]. Furthermore, the Gage-Hamilton Theorem states that convex curves will remain convex under the flow by mean curvature and that the curve will tend to a circle [109]. These theorems imply that any curve will eventually flow down to an infinitesimally small circle under the flow by mean curvature regardless of their original shape.

5.2.2 Numerical Formulation of the CSF Problem

To solve the CSF equation, one might first think of sampling the curve and advancing those sample points according to the differential equation (5.3) in a Lagrangian formulation of the problem. This formulation of the problem was shown to be numerically ill-conditioned, however, so a different approach is required [110]. An Eulerian formulation known as the Level Set Method (LSM) was proposed in [107], where the curve is represented as the zero level set of a function whose values are computed at discrete, fixed points in space. The differential equation (5.3) is then computed over the entire model space, instead of just at its surface, across an artificial time dimension in a dense but numerically stable formulation of the problem. A description of the LSM can be found in [110], with salient details provided in this section.

The curve ϕ can alternatively be represented as the zero level set of an implicit function $\gamma(y) = 0$, as seen in Figure 5.2. The function γ is represented at a set of uniformly spaced sample points. In this thesis, the values between sample points are calculated through bilinear interpolation. A convenient choice for γ in this case is the model's signed distance function, which is negative on the inside and positive on the outside of the model in the conventions adopted here. While other choices for γ exist, the signed distance function is a logical choice because it is straightforward to derive from other model representations and various computationally efficient implementations have already been studied, e.g. [111]. As with the parametric representation, the arguments of γ can be augmented with an artificial time variable as $\gamma(y,t)$, where $\gamma(y,0)$ is the signed distance function of the original model.

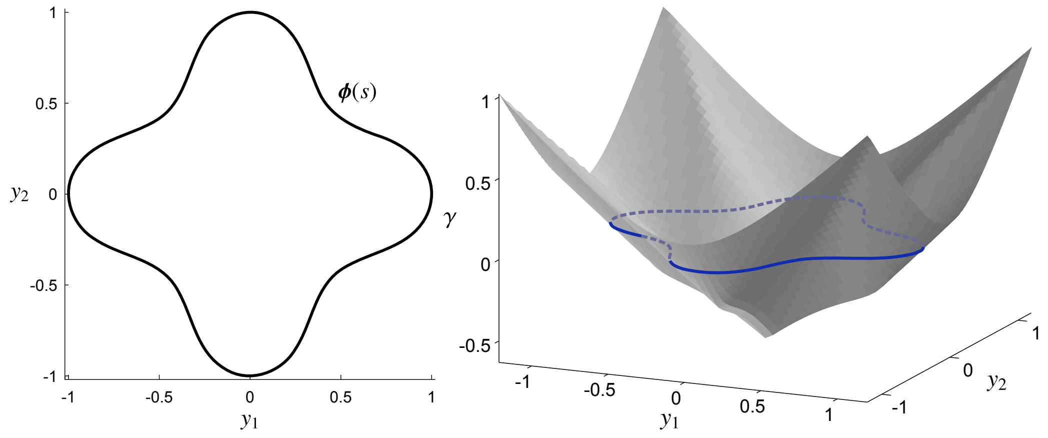


Figure 5.2: Cassinian model as a parametric curve (left) and level set of its signed distance function (right)

The CSF problem can be reformulated using a level set representation as the initial value problem

$$\frac{d\gamma(y,t)}{dt} = v_f(\kappa)|\nabla\gamma|\,, \tag{5.4}$$

where $\gamma(y,0) = \gamma(y)$ and $|\nabla\gamma|$ is the spatial gradient of γ [110]. Under (5.4), the signed distance function is non-uniformly pushed upwards or pulled downwards, eventually shrinking the embedded curve down to a point before disappearing entirely.

Definitions for the normal vector and surface curvature on the implicit model representation, analogous to those from the parametric case, can be defined using the spatial derivatives of γ. The inward normal vector of a signed distance function is the negative of γ's spatial gradient

$$n_c(y,t) = -\frac{\nabla\gamma}{|\nabla\gamma|}\,. \tag{5.5}$$

For the two dimensional case, (5.5) is given by

$$n_c(y_1, y_2, t) = \frac{1}{\sqrt{\frac{\partial \gamma}{\partial y_1}^2 + \frac{\partial \gamma}{\partial y_2}^2}} \begin{bmatrix} \frac{\partial \gamma}{\partial y_1} \\ \frac{\partial \gamma}{\partial y_2} \end{bmatrix}. \tag{5.6}$$

The curvature of a signed distance function is given by the negative of the divergence of the normal vector as

$$\kappa(y, t) = \Delta \cdot \frac{\nabla \gamma}{|\nabla \gamma|}. \tag{5.7}$$

For the two dimensional case, (5.7) is given by

$$\kappa(y_1, y_2, t) = \frac{\frac{\partial \gamma}{\partial y_1}^2 \frac{\partial^2 \gamma}{\partial y_2^2} - 2 \frac{\partial \gamma}{\partial y_1} \frac{\partial \gamma}{\partial y_2} \frac{\partial^2 \gamma}{\partial y_1 \partial y_2} + \frac{\partial \gamma}{\partial y_2}^2 \frac{\partial^2 \gamma}{\partial y_1^2}}{\left(\sqrt{\frac{\partial \gamma}{\partial y_1}^2 + \frac{\partial \gamma}{\partial y_2}^2} \right)^3}. \tag{5.8}$$

As with the parameterized curve representation, κ is positive on convex regions and negative on concave regions.

The spatial derivatives can be numerically approximated using centered differences on the regular grid of the level set representation. The first derivative is given by

$$\frac{\partial \gamma}{\partial y_i} \approx \frac{\gamma(y_{i+1}) - \gamma(y_{i-1})}{2\Delta y_i}, \tag{5.9}$$

and the second derivative by

$$\frac{\partial^2 \gamma}{\partial y_i^2} \approx \frac{\gamma(y_{i+1}) - 2\gamma(y_i) + \gamma(y_{i-1})}{\Delta y_i^2}. \tag{5.10}$$

Higher orders of accuracy for the approximations of the numerical derivatives were found to have little impact on the solutions of the examples in this thesis.

The differential equation in (5.4) can be integrated forward in time using a forward Euler discretization scheme

$$\gamma(y, t + \Delta t) = \gamma(y, t) + v_f(\kappa) |\nabla \gamma(y, t)| \Delta t. \tag{5.11}$$

The time step Δt is limited by the Courant-Friedrichs-Lewy (CFL) condition, which for the differential equation (5.4) with centered difference spatial derivatives and forward Euler temporal integration states that the time step is limited by

$$\Delta t < \frac{1}{2\max(v_f)\left(\frac{1}{\Delta y_1^2} + \frac{1}{\Delta y_2^2}\right)} \tag{5.12}$$

for the two dimensional case [106]. An example of a flow by mean curvature as calculated by (5.4) with $v_f = \kappa$ is shown in Figure 5.3. The ellipse becomes more circular as the flow advances, tending towards an infinitesimal point in the center.

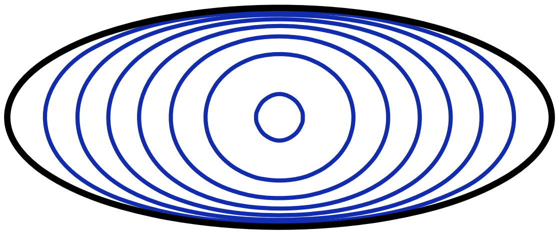


Figure 5.3: The flow under mean curvature, $v_f = \kappa$, of an ellipse

5.2.3 Flow Behaviour and Velocity Functions

While the flow by mean curvature will eventually flow the model down to a point, it is not suitable for grid generation on models with concave regions. The curve will flow outwards beyond the model boundary where $\kappa < 0$ before flowing down to a point, which would extend the coordinate system beyond the model boundary. Figure 5.4 shows the example from [110, p.45], where the curve breaches the model boundary before tending towards a circle. If the simulation were carried further forward in time, the circle would then tend towards a point in the center of the model. To ensure that a homeomorphic transformation function can be created from the flow levels, the curve must pass over each point in space once and only once without the flow reversing its direction.

A generalization of the CSF equation can consider velocity functions $v_f(\kappa)$ other than the mean curvature. For grid generation applications, the velocity should be chosen such that $v_f > 0$

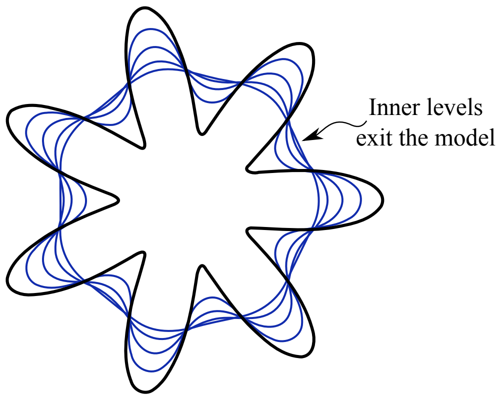


Figure 5.4: Intermediate CSF level curves flowing outside of the original model with $v_f = \kappa$

to ensure that the curve will not flow outwards. A popular case involves fronts that are propagating forward with unit speed, i.e. $v_f = 1$, which yields inset surfaces of the original curve. The example in Figure 5.5 shows two inset surfaces at different depths from the progenitor curve. All but the simplest of curves eventually form cusps or corners on the inner curves at a depth of $1/\kappa$ from the initial curve, as seen on the left side of Figure 5.6. The inner-most curve shown is no longer a Jordan curve, with the swallowtail partitioning the plane into three regions and thus failing to create a suitable set of coordinate lines.

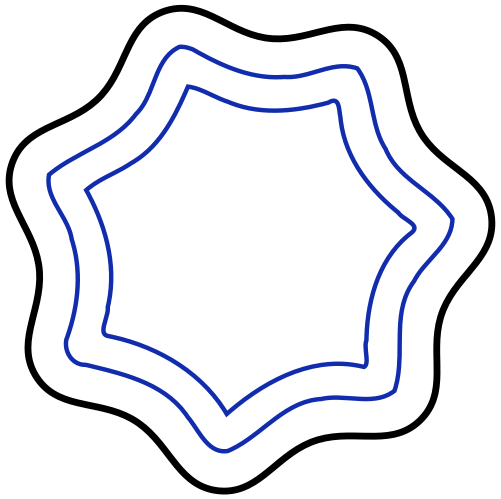


Figure 5.5: Flow with $v_f = 1$ generating inset surfaces

The swallowtail can be trimmed away according to the “entropy condition”, which states that if the curve represents the front of a fire, material should only burn once as the front advances [112]. The entropy condition ensures that the curve does not advance into areas over which the front has already swept and removes the swallowtail, as shown on the right side of Figure 5.6. Where post-processing steps are required to remove such a swallowtail with some methods, the single-valued signed distance function incorporates the entropy implicitly in the LSM's formulation.

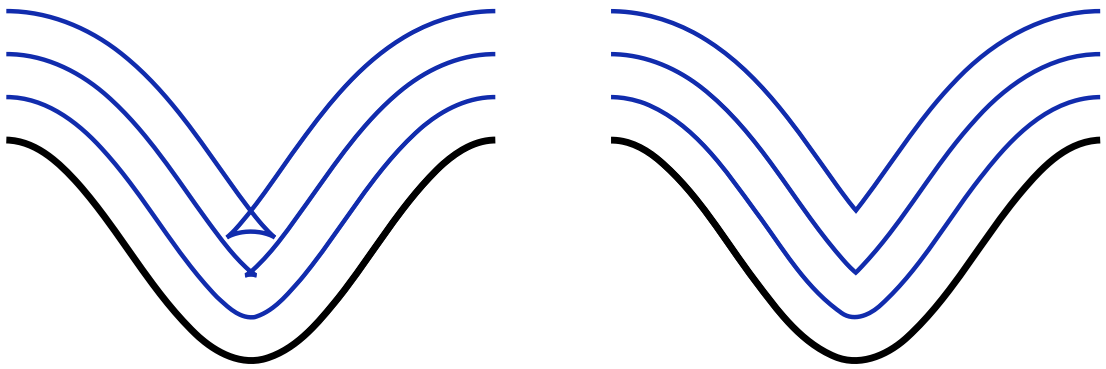


Figure 5.6: CSF with $v_f = 1$ under a Lagrangian formulation (left) and an Eulerian formulation that satisfies the entropy condition (right)

The level curves generated by the LSM are used as a set of tangential coordinate lines in the gridding process. When swallowtails occur and are trimmed away either explicitly in post-processing or implicitly in the LSM, a section of the curve is removed. This makes the generation of transversal coordinate lines more difficult since you can no longer connects points of the same parameter value from different levels as a portion of the curve was removed on an inner level. Heuristic methods can be applied to define transversal lines [106], but the geometric basis for the transversal lines is lost in the process. It is thus preferable to prevent the formation of cusps and corners in the intermediate flow levels.

The cusps in Figure 5.6 arise because the velocity function $v_f = 1$ does not reduce the magnitude of the curvature on convex model features as the curve flows. The flow function for gridding purposes must therefore be a function of the curvature to avoid the formation of cusps, but also remain positive to ensure advancement of the front in the inward direction only. The velocity function

$$v_f^{cap}(\kappa) = \max(\kappa, K_{cap})\,, \tag{5.13}$$

where K_{cap} is the minimum speed at which the curve will move forward, was proposed for grid generation applications [112]. Because v_f^{cap} is always positive the front will only advance forward and because the flow is partially governed by the curvature, the curve will tend towards a circle. Figure 5.7 shows an example of a curve flowing under the velocity (5.13).

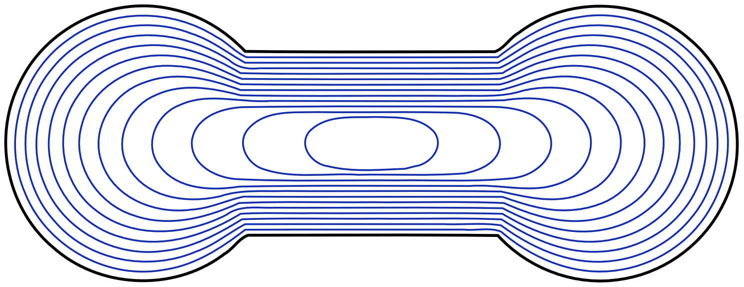


Figure 5.7: CSF of a Dumbbell with velocity $v_f = v_f^{cap}$

When flowing the curve under the velocity function (5.13), $\gamma(y,t)$ will be differentiable in both y and t. The original curve $\gamma(y,0)$ was assumed differentiable in y. The flows in t are constructed by integrating a continuous function, and hence $\gamma(y,t)$ is also differentiable in t so long as the level curves do not self-intersect or approach each other asymptotically. The function γ will not be twice differentiable in t under (5.13), however, as the derivative of (5.13) is discontinuous across points where $\kappa = K_{cap}$. This means that the overall projection function generated from the coordinates lines of γ will have C^1 continuity.

Where a C^1 continuous γ is sufficiently smooth for many applications, the aim here is to incorporate these flow functions into control loops with second order mechanical systems. A velocity function that can produce a C^2 continuous γ is desired to improve the level of continuity of the resulting transformation function and for its eventual incorporation into a feedback linearization scheme as is outline in the future work of Chapter 7. To achieve C^2 continuity, the velocity function v_f must also have a continuous derivative.

Several candidates for v_f were investigated to determine how to best meet the desired behaviours outline in Section 5.1 and it was found that the curve flow began to oscillate if v_f was too large near sharp convex regions. It was determined that the desired v_f is therefore always positive, twice differentiable, near zero for concave regions where $\kappa \ll 0$, and approaches some finite positive value for convex regions where $\kappa \gg 0$ [9]. The arctangent function was designed to provide a suitable velocity function

$$v_f^{arc} = \frac{1}{\pi}\arctan\left(\frac{\kappa}{\kappa_{avg}} - 1\right) + \frac{1}{2}, \tag{5.14}$$

where κ_{avg} is the average curvature given by the weighted average of the curvature with respect to arc length

$$\kappa_{avg} = \frac{\int_a^b \kappa \left|\frac{d\phi}{ds}\right| ds}{\int_a^b \left|\frac{d\phi}{ds}\right| ds}. \tag{5.15}$$

The numerator of κ_{avg} is the total curvature which is always equal to 2π for the Jordan curves considered here [81], while the denominator is the arc-length of the curve. The scaling by κ_{avg} helps ensure that the curve advances at a uniform speed. In the case of a circle with radius r_c, the arc-length of the curve is $2\pi r_c$ and the curvature is $\kappa = 1/r_c$, so the fraction $\kappa/\kappa_{avg} = 2\pi$ is a constant irrespective of the circle's radius. This prevents the flow from accelerating as it tends towards a point and the curvature increases. Maintaining the velocity of the flow simplifies the selection of equally spaced level curves suitable for grid generation. The same model from Figure 5.7 is shown in Figure 5.8 under the new velocity function.

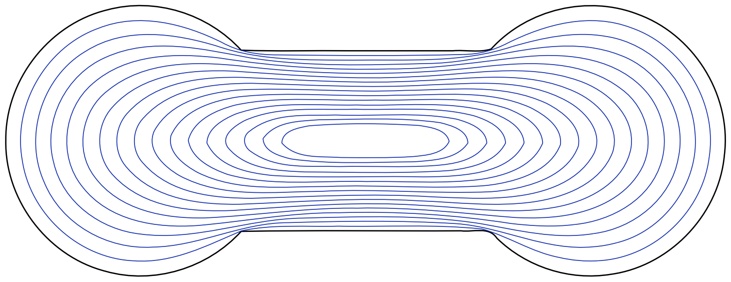


Figure 5.8: CSF of a Dumbbell flowing with $v_f = v_f^{arc}$

While the velocity functions v_f^{cap} and v_f^{arc} are capable successfully flowing some models down to points without generating self-intersection in the level curves, there is no guarantee that the intermediate flows will not undergo topological changes. The Gage-Hamilton and Grayson Theorems applied only when $v_f(\kappa) = \kappa$, so there are no guarantees that the curve will flow down to a point for other velocity functions. In fact, scenarios can be constructed where the curve does undergo a topological change under (5.13) or (5.14), as shown in Figure 5.9.

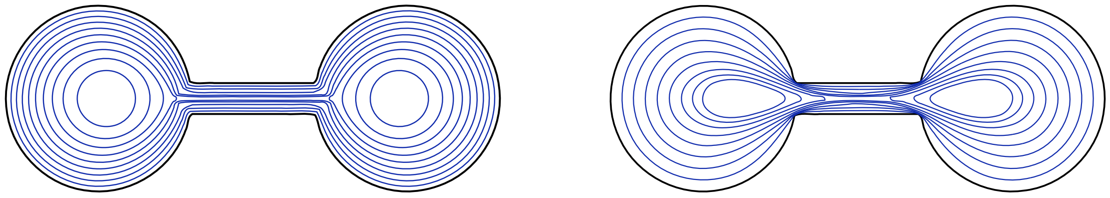


Figure 5.9: Dumbbells that undergo topological changes with v_f^{cap} (left) and v_f^{arc} (right)

The topological changes occur because the velocity function v_f only considers local geometric information, whereas the problem of drawing coordinates lines is a global problem. The global problem will have a solution since one could always draw a set of tangential coordinate lines by hand. Although a suitable set of coordinates lines cannot always be found by the LSM considered here, the flow function v_f^{arc} in (5.14) does work on a variety of model shapes. The examples in Figure 5.10 show the tangential levels for both smooth and angular models with eccentric shapes.

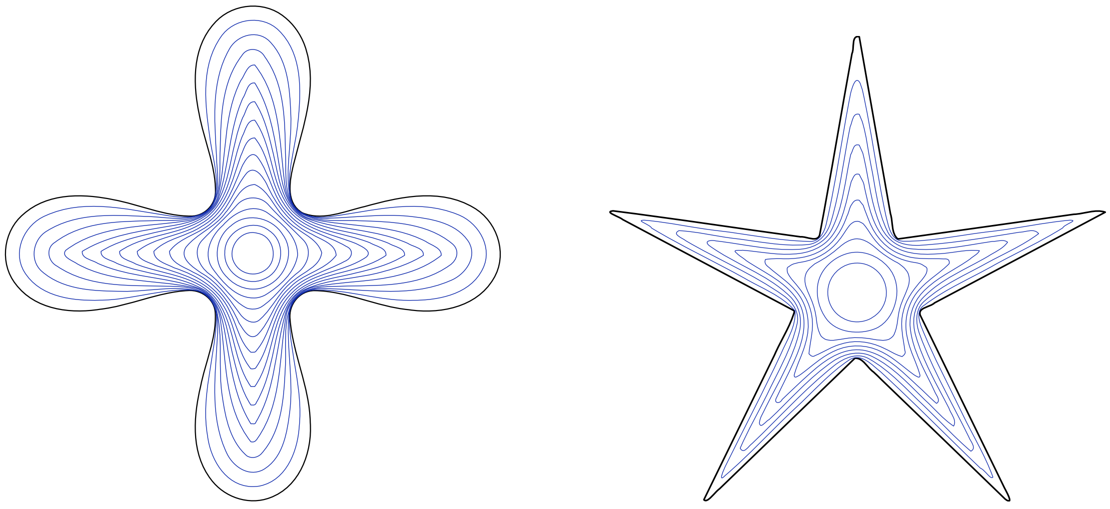


Figure 5.10: CSF with $v_f = v_f^{arc}$ for an eccentric Cassinian (left) and star (right)

5.2.4 Generation of Transversal Lines

The parameterization of the curve is lost when converting it to a zero level set, which means that the transversal grid lines cannot be determined directly by matching points with the same parameter value on different level curves in the flow. Instead, the initial zero level set is sampled with a set of n_p points that are simultaneously flowed with the curve according to an equivalent differential equation as the flow itself [112]. This involves the solving of an additional n_p ordinary differential equations

$$\frac{dX_i}{dt} = v_f(\kappa_i)n_c \tag{5.16}$$

where X_i is the position of the i^{th} point and κ_i is the curvature at that point. A set of n_q points in time are collected for each of the n_p sample points along the curve. The solution to (5.16) will be numerically stable because the values of κ_i are derived from the flows of the LSM, providing a robust means of connecting points between the different levels in the flow.

When the calculations are finished, the $n_p \times n_q$ points obtained by solving (5.16) are the grid points where coordinate lines intersect. The resulting transversal grid lines from (5.16) will intersect the tangential lines at right angles, providing an orthogonal coordinate system [112]. Figure 5.11 shows the grids generated for the Cassinian and star models. The curves in Figure 5.11 have been also be flowed outwards to create a grid over the entire model space. Where running the differential equation (5.4) forward in time makes the curve approach a circle, running the same differential equations backwards would make the curve more eccentric and less circular, causing flows intersect. A separate curve flow is conducted to generate the exterior grid by negating the original level set, evaluating (5.4) on $-\gamma$ and causing the curve to tend towards a far-field circle instead of a central point. Flowing the curves outwards is not strictly necessary for the generation of the transformation function since only the model itself must be transformed, but it facilitates later computational stages in the rendering process.

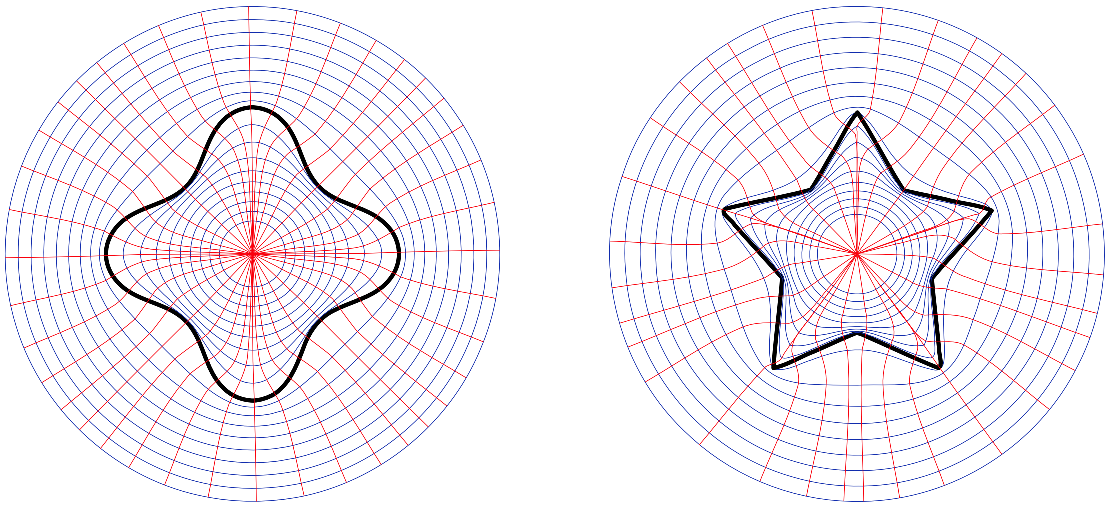


Figure 5.11: Gridding of the Cassinian and star models generated by CSF with v_f^{arc}

5.3 Building Transformation and Projection Functions from the Model Grid

Once a grid has been established over the model, a similar grid can be produced on a circle with the same number of tangential and transversal levels. A correspondence can be drawn between the grid points, which when coupled with an interpolation scheme between grid points defines the forward transformation Ω between the model and circular domains.

The mapping projection is given by

$$\rho_{map}(hip) = \Omega^{-1}\left(r_c \frac{\Omega(hip)}{|\Omega(hip)|}\right) \tag{5.17}$$

where r_c is the radius of the circle. The HIP is transformed into the circular domain, projected to the surface of the circle, and then transformed back into model space. The mapping procedure is shown in Figure 5.12. The inverse transformation Ω^{-1} is defined in a similar way in the opposite direction.

The interpolation scheme is based on the barycentric coordinates of a triangle. Any point within the model interior will be enclosed in a quadrilateral formed by the grid points, with the exception of those points in the center of the model that are contained in triangles. The

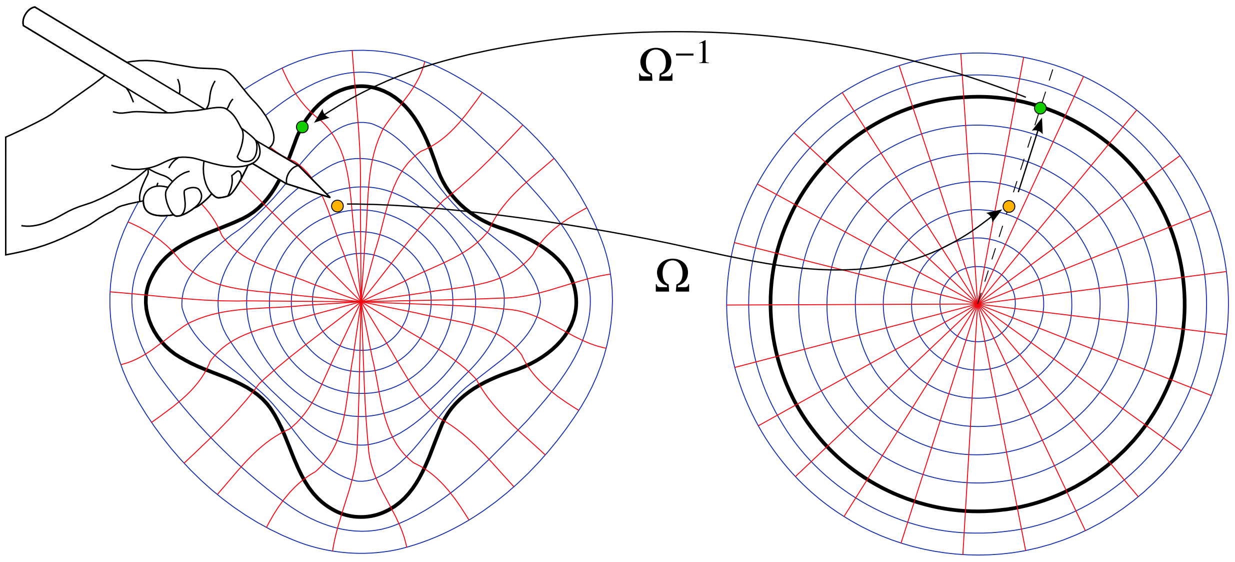


Figure 5.12: Mapping projection based on the transformation Ω between the model domain (left) and circular domain (right)

quadrilaterals are subdivided into four triangles, as shown in Figure 5.13, where c is the point of intersection of the line segments ae and bd. Given a point p_m and the vertices a, b and c in the model space, its barycentric coordinates (A, B, C) are the solution to system of equations

$$p_m = Aa + Bb + Cc\,, \tag{5.18}$$

$$A + B + C = 1\,. \tag{5.19}$$

The mapped point p_c is then given by looking up the corresponding vertices in the circular domain, v, w, and x, and using the same barycentric coordinates from the model domain

$$p_c = Av + Bw + Cx\,. \tag{5.20}$$

The barycentric interpolation provides a one-to-one and onto map from one triangle to another, so the resulting transformation will be a homeomorphism.

While the forward transformation Ω is used to map the model domain onto the circular domain, the inverse transformation Ω^{-1} is only used to map the surface of the circle back onto the surface of the model. The reduction in dimensionality simplifies the barycentric mapping used in

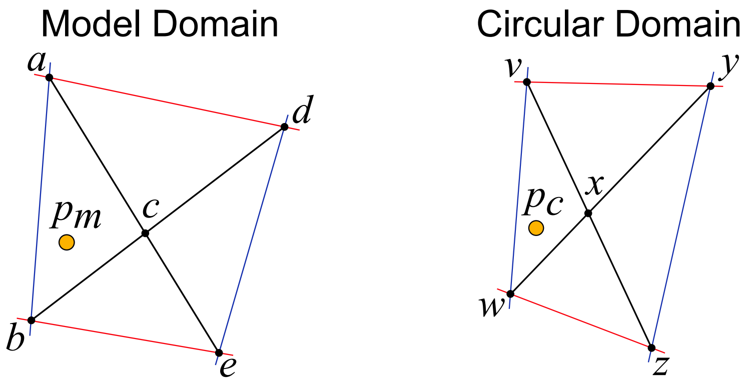


Figure 5.13: Mapping between the grid cells

the forward transformation to a linear interpolation between surface points for the inverse transformation. Although the inverse transform could still use the barycentric mapping, the use of a linear interpolation scheme results in greater accuracy and lower computational costs.

The resulting projection fields for a star and a Cassinian model are shown in Figure 5.14. The projection field lines drawn correspond to the lines of constant parameter in the original curve ϕ as the model is flowed down. The only singularity in the projection is at a point right in the center of the models, which is the hardest point for the operator to reach. The angular features of the star model lead to many field lines converging to a small area on the interior, but the field lines never intersect. The Cassinian's field lines are more evenly distributed thanks to the slowly varying curvature on the model's surface.

5.4 Regulation with Curved Lines of Projection

The curved projection lines require that the surface normal must be used in the regulation scheme. For the minimum distance and constraint projections in Figures 4.5, 4.6, 4.7 and 4.8, the lines of constant projection were straight and intersected the surface at right angles, meaning that e_p and F_r would be perpendicular to the surface at the point of contact. The field lines of the mapping projection in Figure 5.14, however, are curved. When the penetration of the HIP into the model becomes large, the direction of error vector e_p can diverge from the surface's normal vector at the IHIP. This leads to forces that do not reflect the local geometry of the surface.

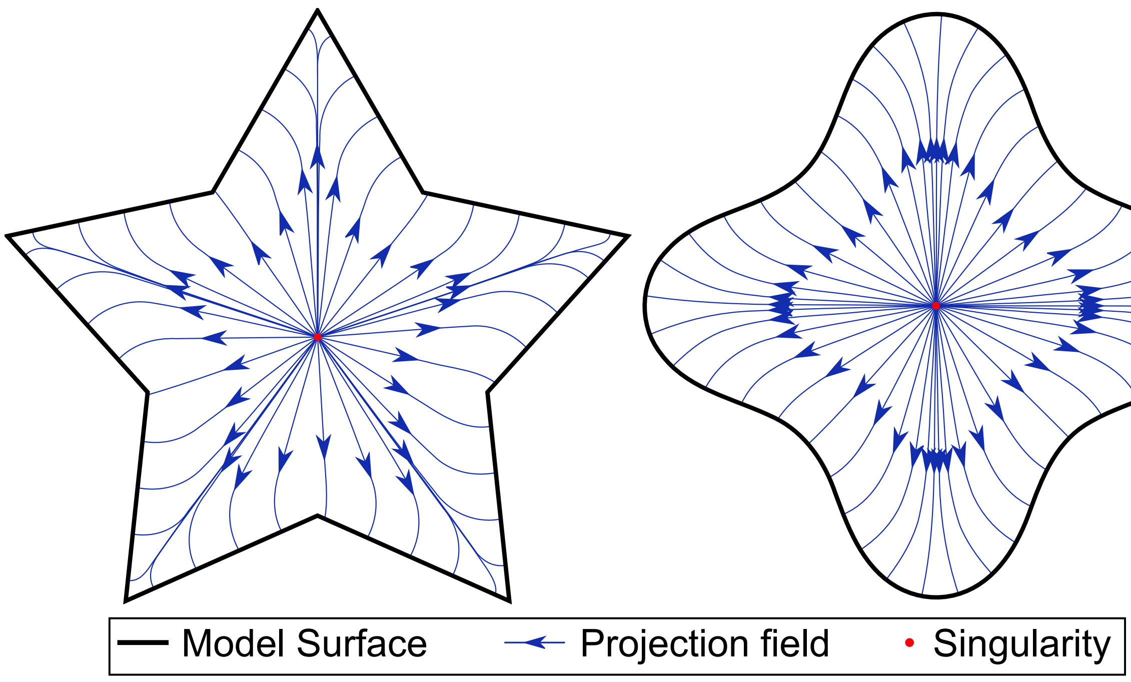


Figure 5.14: Vector field lines of the mapping projection ρ_{map} for the star (left) and Cassinian (right) models

To rectify the situation, a regulation scheme that projects the position error onto the normal vector at the surface contact point is used

$$F_r = K_p \left(n_e \cdot e_p\right) n_e + K_d \left(n_e \cdot \dot{y}\right) n_e \,. \tag{5.21}$$

The projection scheme (5.21) is referred to here as the transversal PD controller since projecting both error quantities onto the normal vector allows control action only in the transverse dimension of the task space. For projections whose lines of constant projection are straight lines that meet the model's surface at right angles, (5.21) is equivalent to the standard PD controller (4.6). A sample projection is shown in Figure 5.15 where it can be seen that the mapping projection would generate a response force in a skewed direction on a flat surface under a standard proportional controller (4.6). The projection onto the normal vector, however, retains the characteristics of the model, improving the perception of model's geometry. This is similar to the force shading algorithm discussed in Subsection 4.3.1, except the normal used in (5.21) is always given directly by the surface geometry without modification.

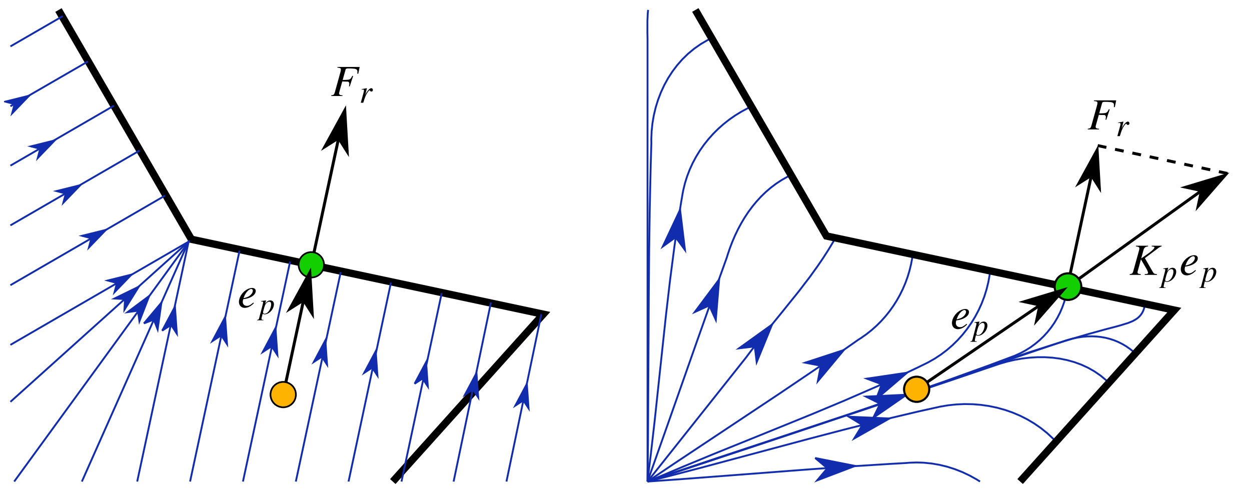


Figure 5.15: Response force for the standard PD controller (4.6) (left) and transversal PD controller (5.21) (right)

The projection onto the surface's equivalent normal vector in (5.21) is necessary to ensure that flat surfaces are rendered with a constant response force direction. If, for example, the position error were rotated to line up with the normal, instead of taking the geometric projection, the magnitude of the response force could change as the operator traversed at a constant penetration depth under a flat surface. The change in magnitude could then be interpreted as curvature when the surface is flat. Because of the geometric projection, the projection fields in Figure 5.14 cannot be viewed as potential fields for the response force direction. The projection fields determine the IHIP, but the final response force magnitude and direction are modulated by the equivalent contact normal.

The new projection function defined in this chapter, along with its accompanying regulation scheme, together form a new haptic renderer. The mapping renderer will be compared against the minimum distance and constraint renderers in the next chapter to explore the relative strengths and weaknesses of each method.

Chapter 6

Comparison and Evaluation of the Haptic Renderers

This chapter evaluates the performance of the minimum distance, constraint and mapping renderers from Chapters 4 and 5 first by considering how a series of trajectories are rendered and then by analysing the closed loop dynamics of the haptic system. The projection schemes are evaluated using a series of test interactions to demonstrate the renderers' capabilities of mimicking the real world qualitatively speaking in various scenarios. It is shown that the different renderers are equivalent when the HIP's penetration into the model is small, but that their performance diverges with increasing penetration depth. In particular, it is shown that the constraint renderer does in fact suffer from pop-through, contrary to previous reports [9].

The closed loop dynamics for the different regulation schemes are analysed, revealing the limitations of the local planar contact approximation inherent in the methods typically found in the haptic literature when applied to curved surfaces. When the model is curved, the controllers are no longer able to separate the control problem into two decoupled subtasks of enforcing the constraint imposed by the model and assigning arbitrary dynamics along its surface. The result is that the centripetal acceleration necessary to track the surface is being supplied by the position error, instead of being explicitly accounted for in the controller. This leads to larger position errors and a distorted experience on curved models.

6.1 Trajectory Based Renderer Comparison

While the projection fields of the minimum distance, constraint and mapping renderers from Sections 4.1.2, 4.1.3 and 5.2.3 are informative, it is difficult to determine the performance of the haptic system from those plots alone. The impact of the different renderer design philosophies on the system's performance is investigated here using a series of test trajectories. The relative benefits and limitations of each approach are the focus of this section. Simulations and experiments are conducted and the results are compared against the phenomena arising in real world interactions. It is shown that the renderers are equivalent for differential penetrations, that they are all subject to pop-through under different circumstances, and that the expected rendering of certain trajectories are unclear when the penetration depth is sufficiently large. But first, the evaluation methodology is covered in greater detail. This section draws extensively from [9].

6.1.1 Evaluation by Test Trajectories

The test cases are comprised of a series of HIP trajectories across different model features including smooth areas, concave corners and convex corners. A series of HIP trajectories are stored in a database, one for each test case. The simulations supply these trajectories to the renderer as if they originated from the robot's encoders in experiment. The resulting IHIP trajectories and response forces calculated by the renderer are stored in a database for analysis and plotting. This methodology isolates the operation of the renderer from the other system components by removing the robot manipulator and the operator, as seen in the modified block diagram of Figure 6.1. The response force and IHIPs are compared across renderers with reference to the physical phenomena named in Section 3.2. The results are drawn using a subset of the HIP and IHIP points in the Cartesian plane to show how the projection onto the surface evolves throughout various interactions. The results are deterministic and noise-free, clearly demonstrating the qualitative behaviour of the renderers.

The feedback-loop between operator, robot and renderer is fundamental to haptic systems and so experiments are necessary to validate that the simulations have not over simplified the problem. The experiments, which also include the operator and haptic device, are presented in

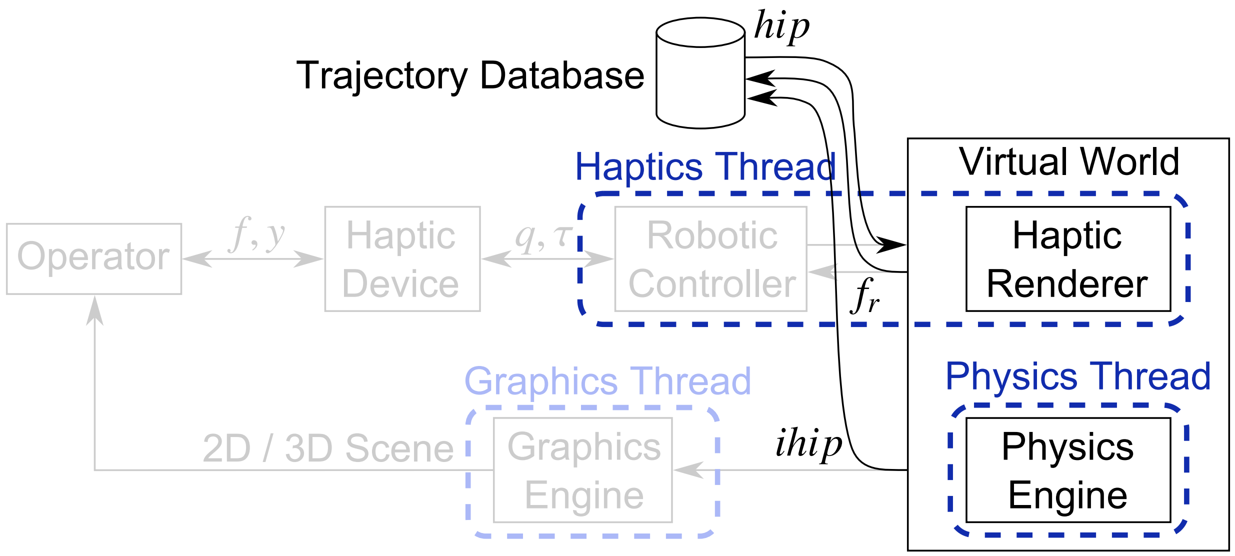


Figure 6.1: Block diagram of the system configuration for evaluating the projection functions in simulation

parallel to demonstrate that the qualitative behaviours described in the simulations are indeed present in the actual system and to describe the sensations that arise in practice. The experiments were conducted using the manipulator pictured in Figure 2.5, calculating the response force using the proportional controller (4.6) for the minimum distance and constraint renderers or the transverse PD controller (5.21) for the mapping renderer. Static model based compensation is employed here, translating the Cartesian response force into joint torques using the Jacobian transpose as in (2.6) from Subsection 2.5.3. The sampling time is $t_s = 1\,\mathrm{ms}$ and the control gains as chosen as $K_p = 1\,\mathrm{kN/m}$ and $K_d = 0$ across the various methods in both simulation and experiment. The HIP, IHIP and force response signals were saved for comparison against the simulated signals and across renderers.

6.1.2 Differential Equivalence of Renderers

Section 2.6 discussed how the HIP will penetrate the model and so position errors are unavoidable, but the magnitude of that penetration will be small when the system is capable of rendering large stiffnesses. The penetration depth on an infinite half-plane under the PD controller (4.6) in static equilibrium is

$$|e_p| = \left|\frac{F_r}{K_p}\right| \quad , \tag{6.1}$$

which is an inversion of (4.6) where the operator's force is equal and opposite to the response force. The depth is therefore proportional to the operator's force and inversely proportional to the proportional gain K_p, which represents the rendered stiffness. Given that the maximum sustainable force from commercially available impedance haptic devices is typically less than 10 N, the operator is capable of achieving significant penetration depths when applying moderate forces [34].

If the system is capable of stably rendering large stiffnesses, the penetration depth will be small and the differences between renderers will be minimized. This is what is referred to here as the differential equivalence of the renderers: that as the penetration depth is decreased, the differences between the IHIPs and response forces of the various renderers becomes less significant. While this is true in a mathematical sense on the smooth surface of the Cassinian, it is shown to be true in a practical sense on the angular star model. All three renderers perform equivalently in the limit as the position error is reduced to zero.

Smooth Model Interactions on the Cassinian

Consider the simulated sliding motion over a smooth model where the HIP is close to the surface, shown in Figure 6.2. The differences between the projections and the response forces of all three methods can only be seen when enlarging the plots significantly. The projections of the mapping renderer are not perpendicular to the surface, and hence the time parameterization of the IHIP trajectory along the Cassinian's surface differs slightly from that of the minimum distance and constraint renderers. The response force of the mapping renderer, however, remains normal to the surface thanks to the regulation scheme in (5.21). As the penetration depth decreases, the position error vector of the mapping renderer approaches the surface normal vector, thus converging to the same error vector as with the minimum distance and constraint projections. This confirms the differential equivalence of the methods on smooth models.

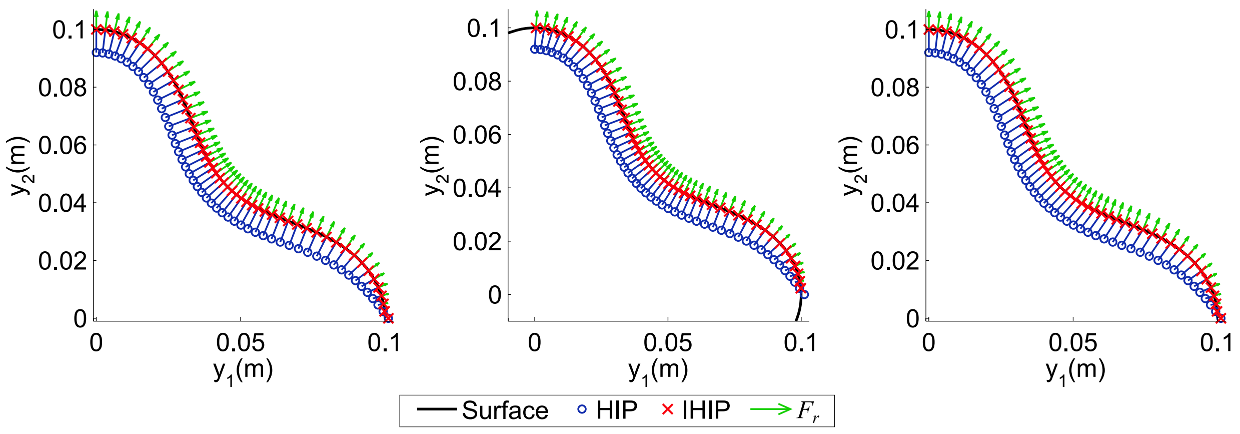


Figure 6.2: Simulated projections for small penetrations: minimum distance (left), constraint (middle) and mapping (right) renderers

The associated response forces are likewise very similar across the methods, as seen in Figure 6.3. The staircase effect in the center of Figure 6.3 with the constraint renderer is due to the discretization of the model used in the polygonal implementation of the constraint renderer (see Subsection 4.1.1). A small pop-through occurs at each discontinuity in the force response as a singularity in the projection function is encountered, which is shown in the projection field of Figure 4.8. The magnitude of these mini pop-throughs diminishes as the model's resolution is increased, receding as the model representation more closely approximates the model's shape. The response force of the minimum distance and mapping renderers on the left and right, on the other hand, are smooth because they are based on a look-up table which calculates the IHIP via interpolation leading to a smooth IHIP trajectory.

The renderings from the experimental case of sliding on the Cassinian are shown in Figure 6.4. The performance between the renderers are very similar, as in the simulations, showing only subtle differences. The penetration depth into the model of 0.8 mm, typical of impedance type haptic renderers. The deviations between the signals are primarily due to the variations in the operator's action in subsequent experimental runs. Approximately the same amount of fluctuation was seen when repeating experiments with a particular renderer as is seen across the different renderers in Figure 6.4. The object feels smooth and its shape is clearly discernible in all cases, successfully recreating the Continuous Contact Point Phenomenon and the

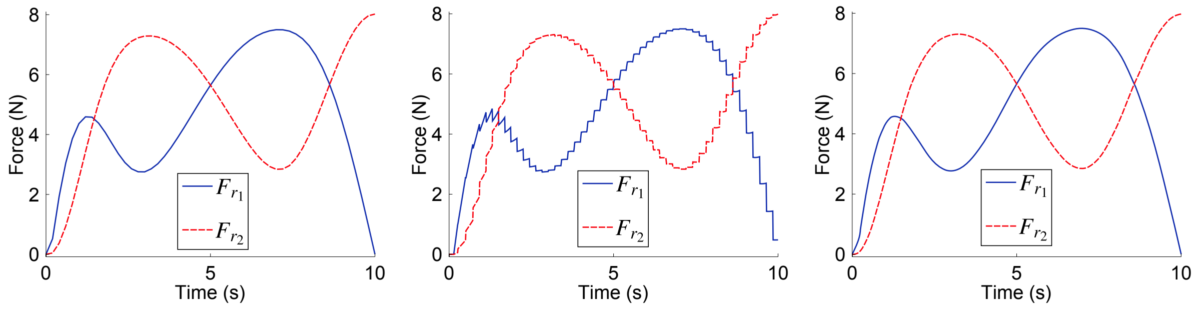


Figure 6.3: Simulated response force for small penetrations: minimum distance (left), constraint (middle) and mapping (right) renderers

Contact Normal Force Phenomenon.

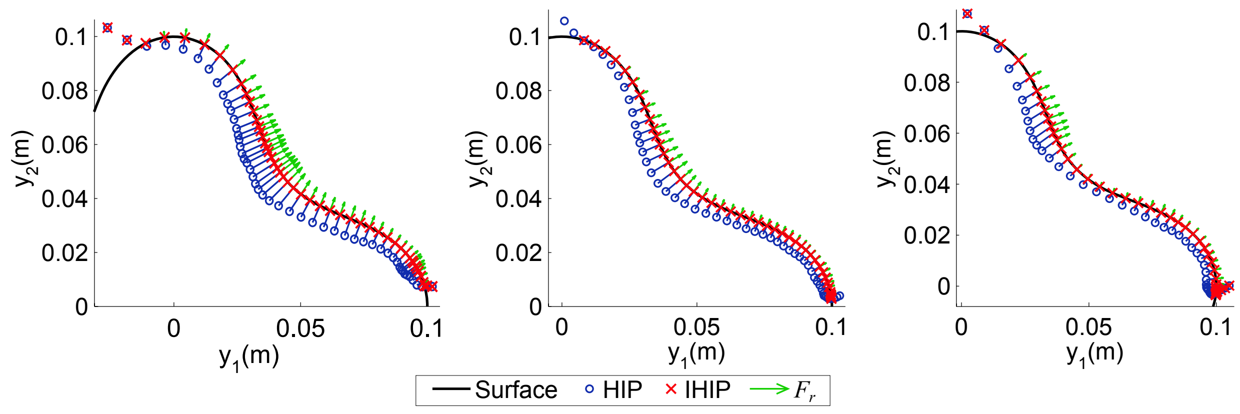


Figure 6.4: Experimental projections for small penetrations: minimum distance (left), constraint (middle) and mapping (right) renderers

The response forces across different simulations could be matched up in time for comparison's sake in Figure 6.3 because they shared the same HIP trajectory as input. The same approach does not work for the experimental results, though, as the operator's actions are irreproducible across experimental runs and therefore do not line up in time. Instead, the polar angle of the HIP is used as an index to line up the datasets across experimental runs. The test cases considered are based on traversing the surface of the model so the angular motion of the HIP with respect to the output space's coordinate system provides a means of associating the IHIPs and response forces between experiments.

The response force is broken down into its direction and magnitude, generating Figure 6.5. The angle of the response force on the left side of Figure 6.5 is very similar across the methods. No perceptible difference was detected between them. This is in agreement with the simulation results of Figure 6.3. While the response force magnitude on the right side of Figure 6.5 varies considerably between the methods, similar variations were observed between experimental runs on any particular method. The differences can be attributed primarily to the differences in the operator's actions between runs.

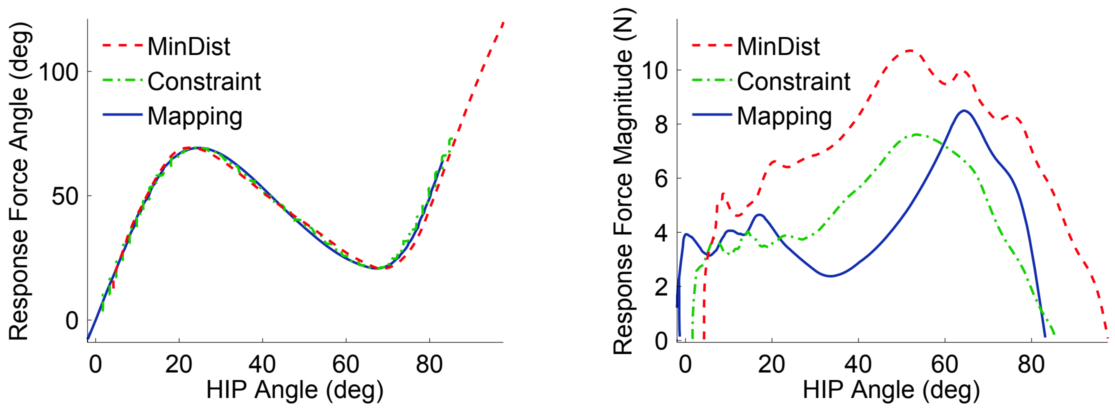


Figure 6.5: Comparison of force response directions and magnitudes for small penetrations

Sliding Across a Concave Corner

The rendering of a traversal over a concave corner, shown in Figure 6.6, also reveals similar performance across the methods. The HIP trajectory can be broken into three portions including straight line segments before and after the corner, connected by an arc under the corner. The renderings of the minimum distance and constraint renderers are identical, mapping the line segments onto the corresponding model faces and the arc onto the corner point. They project a whole range of HIP points in the arc onto the same IHIP at the corner point, where the normal vector is not well defined. This leads to a rounding of the corner as the position error vector gradually changes direction from the normal vector of the first model face to that of the second.

The rounding of the corner can be seen in the sloped line connecting the initial and final directions of the response force in Figure 6.7. It requires several seconds for the transition to occur for the minimum distance and constraint renderers, which would ideally happen instantaneously according to the Abrupt Switch Phenomenon. This rendering is actually equivalent to that of a rounded corner with a certain radius rendered with a higher stiffness. While force shading has been applied to modulate the surface's normal vector to help smooth roughly discretized surfaces [97], these methods cannot sharpen the corners. Since multiple HIPs are mapped onto the same IHIP, they would receive the same modulated normal vector.

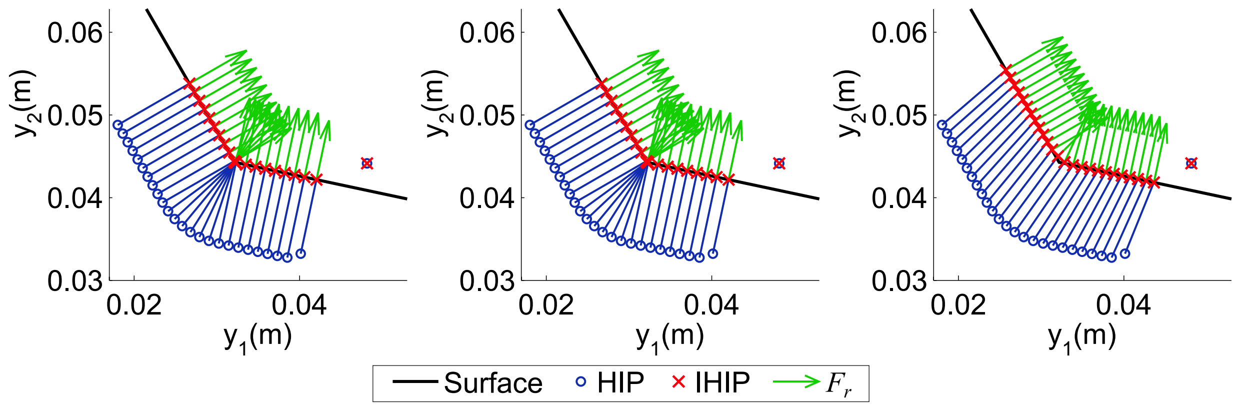


Figure 6.6: Simulated projections around concave corner: minimum distance (left), constraint (middle) and mapping (right) renderers

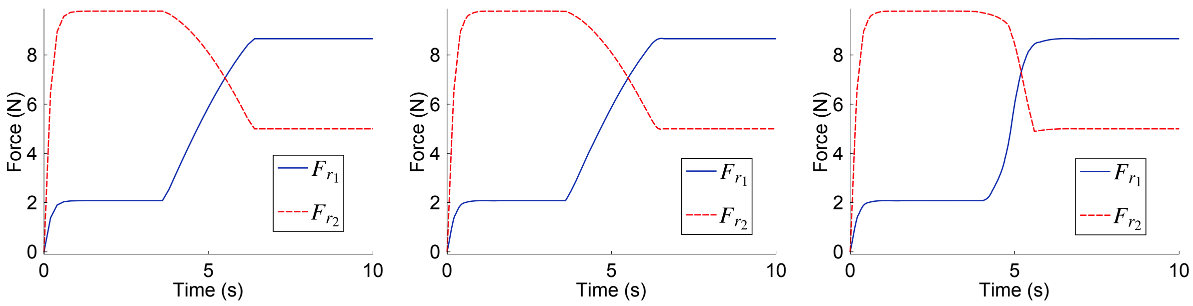


Figure 6.7: Simulated response force around concave corner: minimum distance (left), constraint (middle) and mapping (right) renderers

The mapping renderer, however, distributes the projections of the HIP so that no two HIP points are mapped onto the same IHIP at the surface point. The left and middle plots in Figure 6.7 shows how the position error gradually changes across the corner for the minimum distance and constraint renderers. The response force of the mapping renderer on the right side of Figure 6.7 transitions relatively quickly due to the use of the transversal PD controller (5.21) for regulation. This faster transition better approximates the Abrupt Switch Phenomenon. The transition would happen even more quickly for the mapping renderer as the density of the LUT were increased, which would better capture the change in normal vector direction across the model's surface.

Across all methods, increasing the rendered stiffness and thus reducing the penetration depth makes the transition between response force directions occur more quickly. The size of the arc under the corner diminishes as the HIP trajectory is brought closer to the corner, making the switch between response force directions occur more rapidly both in space and time. This leads to a more accurate rendering of the Abrupt Switch Phenomenon and to a convergence between the methods.

The experimental renderings in Figure 6.8 demonstrate a similar behaviour to the simulation results in Figure 6.6. The experimental HIP trajectories oscillate slightly due to the relatively high proportional gain and lack of a derivative term to introduce damping.

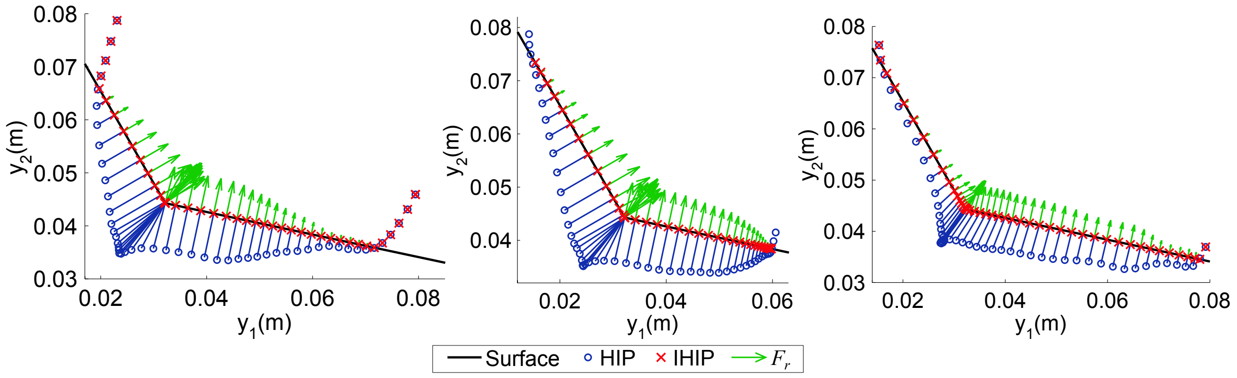


Figure 6.8: Experimental projections around concave corner: minimum distance (left), constraint (middle) and mapping (right) renderers

While the HIP trajectories of the three methods look very similar in Figure 6.8, only the mapping renderer's HIP trajectory loops back over itself. The HIP trajectory for the mapping renderer actually reverses direction temporarily before continuing around the corner, as can be seen in the loop of the peak of the force magnitude on the right side of Figure 6.9. Such a loop shows that the operator's momentum was arrested, which helps sharpen the perception of concave corner by delivering a closer approximation to the Impulsive Collision Force Phenomenon when entering the corner. The force magnitude for the minimum distance and constraint methods contain no such loops. The response force delivered for all methods displayed a minor peaking effect similar to an impulse, but the quality of the impulse would have been improved had they been delivered using other techniques specifically tailored to that phenomenon [102].

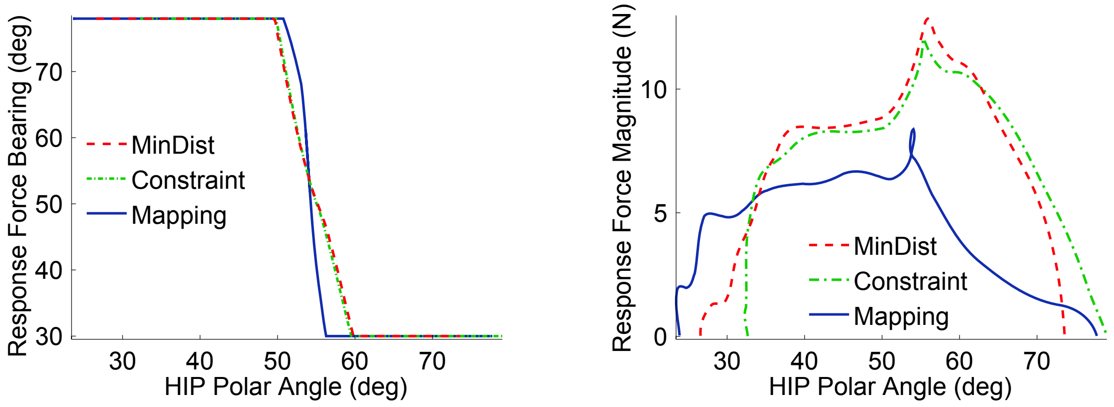


Figure 6.9: Comparison of force response directions and magnitudes around concave corner

The left side of Figure 6.9 shows how the response force of the mapping method transitions between the start and end response force directions faster in space than the other methods. In all cases, it was clear that corners were being rendered, but the experience under the mapping renderer was noticeably sharper than the others. When using a lighter touch across the corner, the HIP's penetration was smaller and the experience between the methods became less tangible, confirming the renderers' equivalence in the limit.

Sliding Over a Convex Corner

The difference between the renderers on a convex corner are more significant, but the methods again converge as the penetration depth decreases. Figure 6.10 shows the simulated sliding off of a convex corner. Only the constraint renderer correctly renders the contact by instantaneously dropping the response force to zero as the corner is cleared, in line with the Broken Contact Phenomenon. The response forces in Figure 6.11 show that the minimum distance and mapping renderers are not instantaneously zeroed, but rather change direction mid-way through the interaction over the corner. F_{r_2} abruptly changes sign at $t \approx 5$ s, while the constraint renderer maintains a constant contact until zeroing the response force at $t \approx 9$ s.

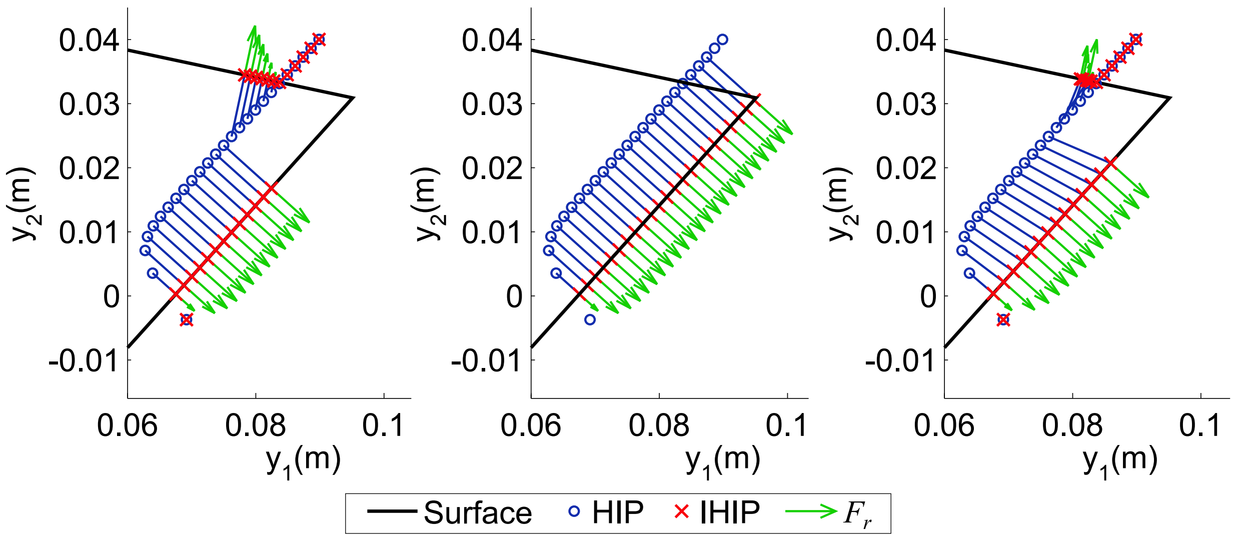


Figure 6.10: Simulated projections around convex corner: minimum distance (left), constraint (middle) and mapping (right) renderers

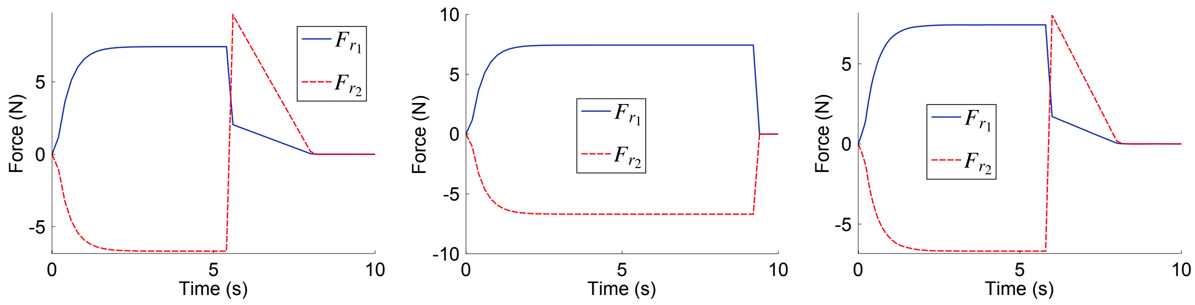


Figure 6.11: Simulated response force around convex corner: minimum distance (left), constraint (middle) and mapping (right) renderers

The occurrence of pop-through in some renderers and not in others would seem to run contrary to the claim of their differential equivalence, but the magnitude of the pop-through is proportional to the penetration depth. Increasing the stiffness rendered will reduce the penetration depth so that force transition time for the minimum distance and mapping renderers will advance from $t \approx 5$ s towards the transition time of the constraint renderer at $t \approx 9$ s. If the erroneous force at the distant model face is presented to the operator for a sufficiently short amount of time, little momentum is transferred to the operator's hand and the pop-through will blend in with the discontinuity associated with the Broken Contact Phenomenon.

The experimental results in Figure 6.12 reproduce the differences between renderers seen in the simulated results. While the differences between the minimum distance and mapping renderers that suffer from pop-through and the constraint renderer that does not are clearly visible in the figure, the difference was minor in practice because the proportional gain was sufficiently high. Increasing the control gain has the effect of reducing the pop-through's impact, again suggesting the differential equivalence of the renderers. Turning down the proportional gain exaggerated the pop-through, confirming that the magnitude of the pop-through is inversely proportional with rendered surface stiffness.

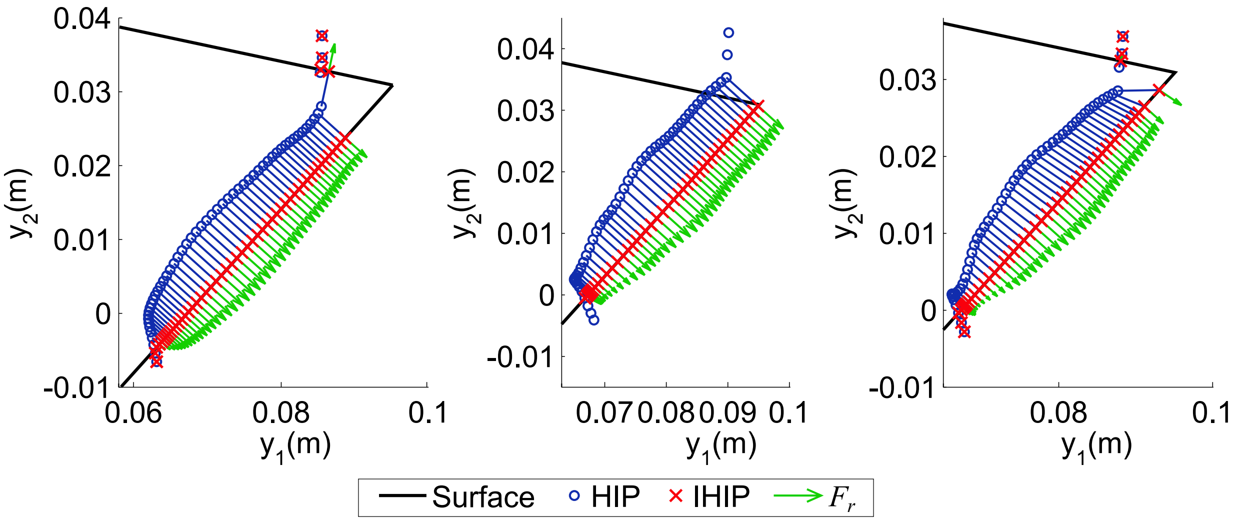


Figure 6.12: Experimental projections around convex corner: minimum distance (left), constraint (middle) and mapping (right) renderers

6.1.3 Pop-through

The studied renderers may be differentially equivalent, but their behaviour diverges in important ways as the penetration depth increases. At some point as the penetration depth increases, the operator will start to perceive being pushed out of the model, especially when popping through a thin portion of the model. Experience with haptic systems indicates that the penetration depth necessary to trigger pop-through on angular models is readily attained in practice, and so the considerations that follow are of practical relevance.

It is well known that the minimum distance method suffers from pop-through [44], which can be seen in Figure 6.10 by the lack of IHIP points near the corner and in Figure 6.11 by the change in the response force direction. This represents a fundamental limitation of the minimum distance renderer that is unaffected by increasing the system's sampling rate, which corresponds to the definition of pop-through as a violation of the Continuous Contact Point Phenomenon.

The mapping renderer also experiences a discontinuity in its force response from Figure 6.11, but in this case it is not due to the pop-through as a violation of Continuous Contact Point Phenomenon. If the sampling rate in the system were increased, a greater number of IHIP points would have appeared around the corner in the projections of Figure 6.10. In the limit, the continuous case would have the IHIP sliding over the entire corner. The increase in the sampling rate would not solve the problem altogether, however, as the operator would still experience a change in response force direction that would fail to simulate the Broken Contact Phenomenon.

What was previously unknown is that pop-through is also possible on curved models for the minimum distance and constraint renderers [9]. The projections fields from Sections 4.1.2, 4.1.3 and 5.2.3 showed where the singularities occur in the model for the different projection functions, with pop-through arising whenever the HIP trajectory passes through a singular point. Figure 6.13 shows the simulation results for a deeper trajectory on the Cassinian, with pop-through occurring for both the minimum distance and constraint renderers. No IHIP points are mapped onto the tip of the Cassinian's lobe, so the operator bypasses this portion of the model's surface, effectively erasing the lobes from the interaction experience. The mapping renderer prevents pop-through by generating a continuous series of IHIP points along the surface (or at least would be made continuous as the sampling time of the system approached infinity), presenting the entire traversed surface to the operator. The effects of pop-through appear in the discontinuities of the force response of Figure 6.14 for the minimum distance method at around $t \approx 5\,\mathrm{s}$ and the constraint method at around $t \approx 6\,\mathrm{s}$. The mapping renderer's forces are smooth signals, providing a closer emulation of real world interactions with a smooth object according to the Continuous Contact Point Phenomenon.

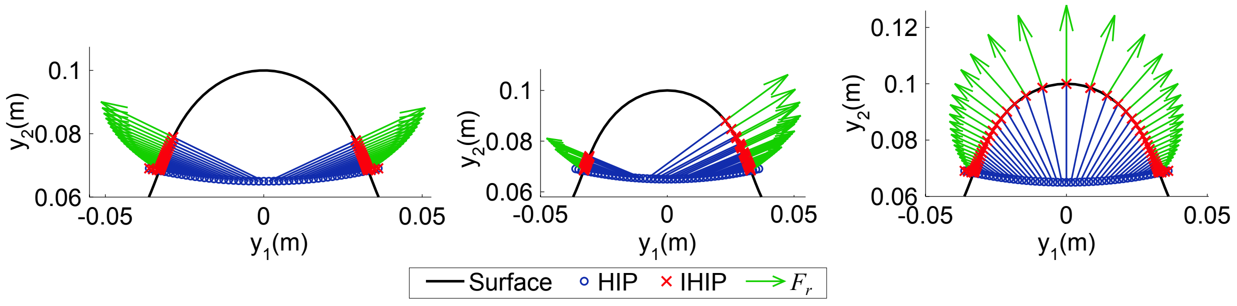


Figure 6.13: Simulated projections for large penetrations: minimum distance (left), constraint (middle) and mapping (right) renderers

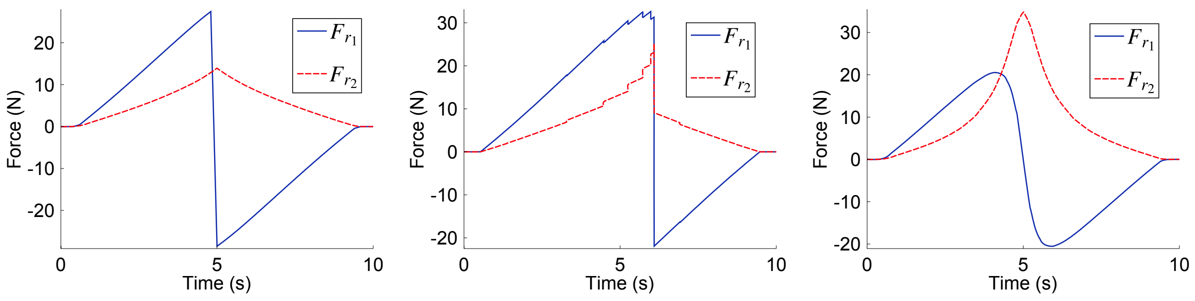


Figure 6.14: Simulated response force for large penetrations: minimum distance (left), constraint (middle) and mapping (right) renderers

To test this trajectory safely in experiments, the control gain was lowered to $K_p = 100\,\text{N/m}$, mimicking a device with limited force output capabilities. The experimental trajectories of Figure 6.15 confirm the existence of pop-through for the minimum distance and constraint renderers that was seen in the simulations of Figure 6.13. The sudden transition between response force directions for the minimum distance and constraint renderers produces an experience similar to passing over a convex corner, distorting the operator's perception of the model's geometry. By inducing pop-through on the Cassnian's lobes, corners have effectively been introduced into the model. With the mapping renderer, however, the operator experiences a smaller version of the Cassinian, including the full range of response force directions as would be expected from such a curved surface according to the normal directions and the Contact Normal Force Phenomenon.

The mismatch between the actual size and perceived size of the model is noticeable because of the large position errors, but the curvature of the model is still readily felt under the mapping renderer.

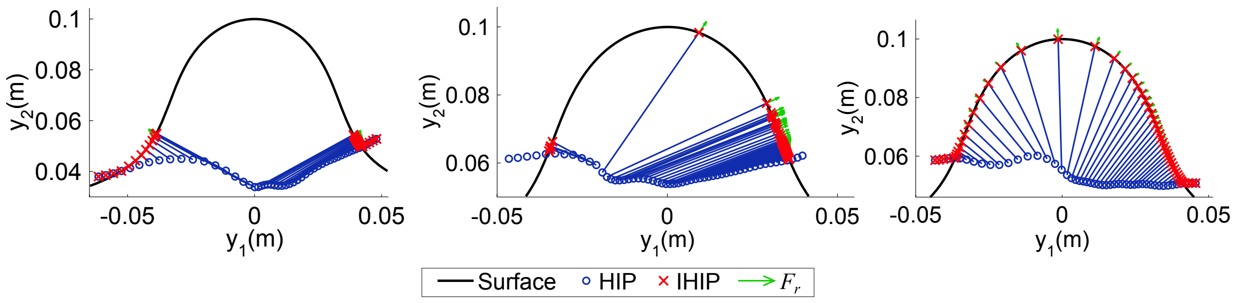


Figure 6.15: Experimental projections for large penetrations: minimum distance (left), constraint (middle) and mapping (right) renderers

The possibility of creating a hybrid method that combines the mapping and constraint renderers was considered in an attempt to solve the pop-through problem on both curved and angular models. One way to create a hybrid method would be to switch between the mapping renderer on curved portions of a model and the constraint renderer on angular portions of a model. This involves essentially cutting the projection fields from Chapters Chapter 4 and Chapter 5 according to a switching law and joining the pieces together. Since the projection fields have different forms, they cannot be smoothly joined and would thus create discontinuities in different places.

The severity of the pop-through changes with penetration depth, becoming progressively worse for the minimum distance renderer, but only exists over a certain range of penetration depths for the constraint renderer. Figure 6.16 shows the rendering on the Cassinian for all three renderers as the penetration depth is increased. As the HIP's trajectory becomes deep enough to cross the model's medial axis, the minimum distance renderer starts to violate the Continuous Contact Point Phenomenon and pop-through occurs. The pop-through becomes progressively more severe as the HIP's depth is increased, with the IHIP skipping over larger and larger portions of the model's surface.

The constraint renderer exhibits the minor zig-zag pop-through for the shallow penetration in depth in Figure 6.16, as previously seen in Figure 6.3. At a certain point, the severity of the pop-through increases significantly as the IHIP skips over several faces on the model in a single time step as seen in the second row, middle column of Figure 6.16. If the model were continuous, significant pop-through would arise when the depth was sufficient to cross the medial axis as in the projection fields from Figure 4.7, but occurs at a greater penetration depth in the discretized model case here as can be deduced from Figure 4.8. As the penetration depth is further increased, the pop-through ceases to occur as the IHIP becomes constrained between inflection points on the surface.

While the minimum distance and constraint renderers suffer from pop-through on the smooth model of Figure 6.16, the mapping renderer remains continuous. The mapping renderer was specifically designed to address this case, ensuring that the Continuous Contact Point and the Contact Normal Force phenomena would be satisfied.

6.1.4 Ambiguous Interactions

While the pop-through discussed in Subsection 6.1.3 always constitutes an undesired haptic artifact, the proper way to render a specific interaction scenario cannot always be determined solely by the HIP's trajectory. The nature of the task at hand impacts the conduct of the operator and their expectations of the scene's rendering. The operator acts under a hybrid position / force control mode analogous to the force control schemes in control theory in certain situations [113, 53]. The relative emphasis between position versus force control depends on the interaction scenario at hand. It is unclear from the HIP signal alone under which mode the operator is currently acting and therefore the correct rendering of a scene cannot always be determined from sensor input. A sample scenario is considered in this section to demonstrate the problem at hand.

Similar HIP trajectories can arise when trying to accomplish different types of tasks, each requiring a different set of response forces to properly render the interaction. The HIP trajectory from the bottom row of Figure 6.16 could represent a primarily position control task, like wood carving. In this case, the operator will try to shape the wood while counteracting components of the response force that push the HIP off of the operator's intended trajectory. The constraint renderer provides the proper rendering, maintaining the IHIP near the initial contact point. This provides the resistance expected when the operator is pushing straight into the model with a large

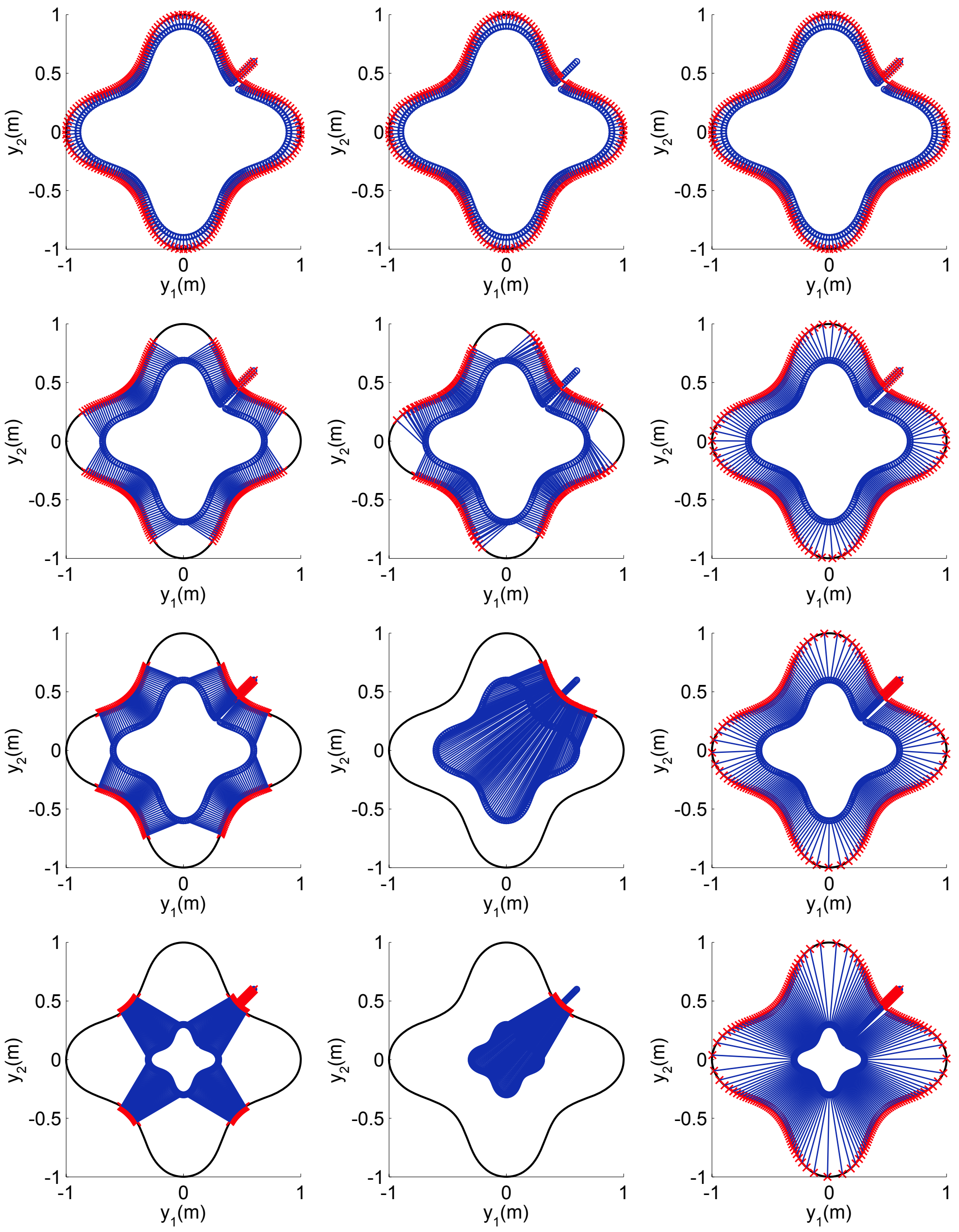


Figure 6.16: Projections on the Cassinian for increasing penetration depths under the minimum distance (left column), constraint (middle column) and mapping (right column) renderers

force. The IHIP trajectory of the mapping renderer, however, would incorrectly allow the tool to pass through the piece.

On the other hand, the task in the bottom row of Figure 6.16 could be a primarily force control task, like the plaque removal on the surface of a tooth [114]. In this case, the operator is trying to apply a consistent normal force while using the surface's geometry to guide their traversal over the surface. The proper rendering would allow the tool to traverse the surface while the pressure is maintained, which is exactly what the mapping renderer displays. The constraint renderer, however, cannot generate a smooth IHIP trajectory around the entire surface when the penetration depth is large. The IHIP remains stuck on concave model features, as seen in the bottom middle of Figure 6.16. If the operator tries to push the IHIP to another part of the model's surface by force, then pop-through will result like in Figure 6.15 or the contact will be broken according to the projection field in Figure 4.8.

6.1.5 Summary of the Projection Function's Performance

The differences between renderers diminish with penetration depth, so the choice of method matters little for systems capable of rendering very high stiffnesses. In many practical cases, the HIP's depth will be sufficient so that the dissimilarities between methods will be significant. The pop-through from Subsection 6.1.3 and ambiguous interactions from Subsection 6.1.4 can occur when the HIP penetrates sufficiently into the model. The model's medial axis played a role in determining both where pop-through and ambiguous renderings would arise, describing the conditions under which aberrant system behavior may occur.

6.2 Computational Considerations

The computational complexity of the renderer is also an important consideration in the performance of a haptic renderer. Both the processing power and memory requirements are significant factors, although processing power is generally of primary consideration due to its impact on the system's sampling rate, which in turn impacts system stability as discussed in Section 2.6. The renderers can all be implemented using algorithms that run entirely online, but it is advantageous if a portion of the computations can be pre-executed offline. The minimum distance and mapping renderers are well suited to shifting the burden of computation offline since their projection functions depend only on the HIP, whereas the constraint renderer's projection uses the past IHIP as well.

The main disadvantage of the constraint method is that it requires online collision detection for its initialization, which is challenging to perform at robotic force control rates for complicated models. For the scenarios considered in this paper, the collision between the model and a line segment formed by the previous and current HIP is required. When hierarchical collision detection schemes like an Oriented Bounding Box tree are used, the computational complexity is $O(\log n)$ where n is the number of triangles in the model [13]. While it is possible to perform such calculations at rates of 1 kHz for complicated models, these calculations still demand significant processing power that may be required for simulating other aspects of the virtual world. It is thus beneficial to minimize the computational load deriving from the haptic renderer. Once the collision has been detected, specialized data structures allow for the efficient update of the constraint projection [45].

The minimum distance and mapping projections, however, can be computed offline for a number of points throughout the model space whose results are stored in a look-up table (LUT) for later interpolation online. The model space is sampled uniformly and ρ is evaluated offline at each point. The resulting projection (or lack thereof if the grid point is outside the model) is stored to form the LUT. The LUT based implementation allows the bulk of the computational costs to be incurred offline. The renderer would then index into the LUT and interpolate between sample points online, which is an $O(1)$ operation with respect to the model's complexity. The online evaluation is trivial, enabling very fast update rates. The LUT also incorporates the functionality of the collision detection algorithm, further reducing the computational load on the system. The use of a LUT will allow the methods to run at rates in excess of 1 kHz, enabling future research into the effects of sampling rates on the limits of haptic stability. For very large virtual environments that need to sampled at very fine resolutions, the memory requirements could exceed the available main memory. In this case, a pre-fetching caching scheme would have to be implemented to load portions of the LUT based on the current output and velocity. In this case, a hierarchical decomposition of the output space would be required and the algorithm would be $O(\log n)$ asymptotically speaking, although the actual cost would grow very slowly with n.

The LUT is generated by uniformly sampling the model space and repeatedly performing the projection procedure offline for later interpolation online. The IHIP location as well as the normal vector at the IHIP are stored, each being represent using N doubles, where N is the number of dimensions in the output space. Each sample point thus requires $2 \times 8 \times N$ bytes, since a double is 8 bytes and both the IHIP and normal vector are stored. The total memory required in bytes is therefore

$$M = 16N \prod_{n=1}^{N} \frac{D_i}{\Delta_i} \tag{6.2}$$

where D_i is the size of the object and Δ_i is the distance between samples in the i^{th} dimension. In general, the sampling resolution should be fine enough to capture the salient details of the model, but need not be so fine as to capture the surface's texture that can be better incorporated using other specialized techniques [98]. The experiments from Section 6.1 used a spatial resolution of $\Delta_i = 0.5$mm, which was sufficient to prevent the appearance of discretization artifacts in the experimental results. For the cases considered in the previous section, the objects measured 200mm across on each side and a spatial resolution of 0.5mm was used in each direction. This yields a storage requirement of

$$M_2 = 32\frac{200^2}{0.5^2} \approx 5.12\text{MB}$$

for the 2D case presented in this paper. The important question to consider is whether this method would scale to the 3D case whose data storage requirements are greater. For the 3D case, M_2 becomes

$$M_3 = 48\frac{200^3}{0.5^3} \approx 3\text{GB}$$

Given the availability of inexpensive storage in the terabyte range, it is possible to construct a world composed of many virtual objects using this method. The calculation in (6.2) uses a worst-case scenario of dense sampling. Alternate structures like octrees can hierarchically decompose the space, sampling more coarsely over regions that change little while still capturing the salient details in highly variable regions [115]. This captures the model details while reducing the overall storage requirements. To further reduce the storage requirements, the LUT could incorporate higher order interpolation schemes to better capture the model's curvature using fewer data points.

6.3 Dynamic Properties of the Renderers

The evaluations of Section 6.1 addressed effects that were mainly independent of the HIP's velocity, studying HIP and IHIP trajectories without reference to durations and velocities. It is impossible to distinguish between the contributions of the centripetal and normal contact forces without considering the velocity of the HIP since both forces act in the same direction and are combined in F_r. It is only by including the HIP's velocity in the analysis that it can be verified whether or not both the Contact Normal Force and Centripetal Force phenomena are being properly simulated. It is therefore important to supplement the investigations from Section 6.1 to more fully evaluate the performance of the renderers.

Where the artifacts of the previous section arose primarily due to the projection function, this section focuses on the performance of the regulation scheme. The closed-loop dynamics of the haptic system under the regulation schemes based on the PD controller (4.6) with the velocity definitions of the projected proportional (PP) in (4.7), PP with viscous damping (PP+V) in (4.9), PP with derivative (PP+D) in (4.11) and PP with projected derivative (PP+PD) (4.13) from traditional haptic implementations (summarized in Table 4.1) as well as the transversal PD controller projected (5.21) are compared, with simulations conducted to demonstrate how well they simulate the frictionless sliding case in the absence of the operator. Any deviation from the desired behaviour under these idealized conditions reveals haptic artifacts introduced by the regulation scheme. It will be shown that all of the renderers considered rely on position errors to supply both the contact and centripetal force, leading to larger position errors when the HIP moves swiftly across curved models. The conclusions from this section were first presented in [8].

Modifications are necessary to the testing framework from Section 6.1 to perform this analysis. The system is now treated as an automatic control system instead of a human-machine cooperative task. This converts the operator's influence from an integral component of the system to an unmodelled external disturbance. The haptic renderer is converted from a unilateral to a bilateral constraint, now considering both the model's exterior and interior to be "inside" of the object. Figure 6.17 shows the evaluation topology consisting of the feedback loop between the haptic device, robotic controller and haptic renderer used throughout the rest of this chapter. The analysis that follows considers the ability of the haptic controllers to maintain the output on the surface and regulate its dynamics along the surface. If the system is unable to achieve the desired behaviour without the operator's disturbance, its performance will only degrade as the operator is re-introduced. In this sense, the loop in Figure 6.17 can provide necessary but not sufficient conditions for the haptic systems to be able to satisfy their objectives.

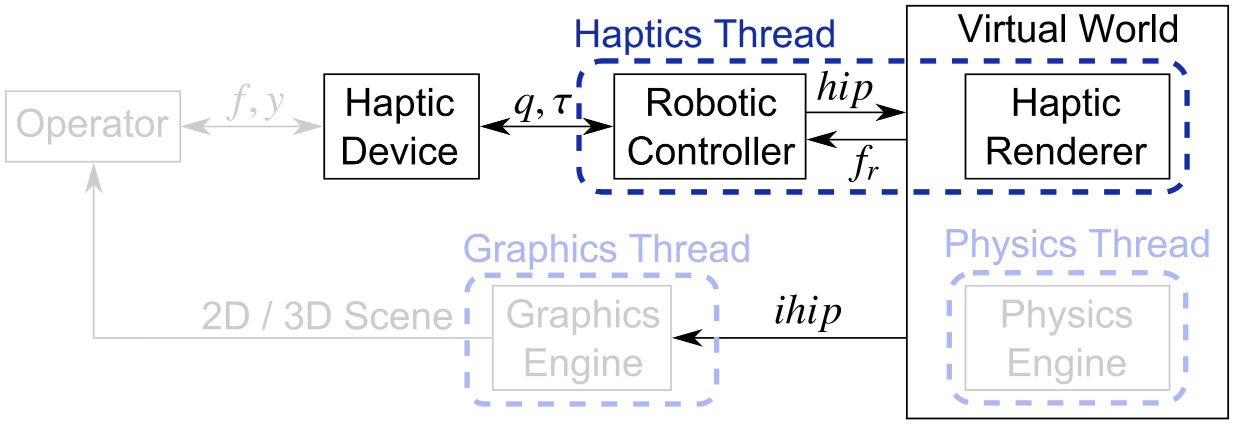


Figure 6.17: Block diagram of the system used for the dynamic testing and analysis

Haptic renderers represent a special case of a virtual constraint. Virtual constraints implement a desired behaviour in a system by imposing relationships on its state and/or output through the action of controller acting in addition to the natural constraints arising from the physics of the system itself. They are analogous to the mechanical constraints that arise in rigid body dynamics [116]. A haptic renderer can be seen as a unilateral constraint on the output of the system, where the system should satisfy $\gamma(y) \geq 0$, where y is the HIP and γ represents the model as a zero level set [74]. Unilateral constraints are difficult to analyse, however, because they give rise to systems of switched differential equations since the virtual tool transitions between being constrained by contact with the surface and moving freely in space. When the constraint is viewed as a bilateral constraint, on the other hand, with $\gamma(y) = 0$, the analysis techniques for continuous differential equations can be applied. This is essentially the approach that was adopted in [67], with the assumption that the contact remains unbroken instead of reformulating the constraint as bilateral.

The regulation schemes seek to satisfy two objectives simultaneously: driving the HIP onto the constraint and implementing the desired friction dynamics along the surface. This constitutes separating the control action along the dimensions transversal and tangential to the surface of the model. When considering the haptic renderer as a bilateral constraint, the transversal controller is desired to stabilize the constraint, such that the HIP will approach and remain on the constraint. In the frictionless sliding case, the desired tangential dynamics are those of a double integrator representing a mass driven by a force without resistance. It is according to these performance objectives that the renderers are evaluated here.

Expressions for the virtual object, regulation scheme and haptic device are required to form the closed-loop dynamics of the operator-less subsystem of Figure 6.17. The virtual objects for this section's test cases were chosen to test the effects of constraint curvature on the performance of the regulation scheme. The line

$$y_2 = 0 \tag{6.3}$$

and circle

$$y_1^2 + y_2^2 = r_c^2 \quad , \tag{6.4}$$

where $r_c = 0.1\,\mathrm{m}$ is the radius of the circle, are used for the analyses that follow. The projection function for all of the renderers happens to be identical for the line and for the circle in 2D, so the tests here are able to separate the impact of the regulation scheme from that of the projection function in the haptic renderer.

The regulation schemes from Table 4.1 are compared here. Because of the particular form of line (6.3) and circle (6.4) constraints, the projections onto the constraints produce position error vectors that are normal to the constraint at the point of projection. Thanks to this property, the transversal PD controller (5.21) is equivalent to the standard PD controller (4.6) on which the controllers of Table 4.1 are based. For this reason, the regulation scheme (5.21) is not included in the comparative analysis that follows since its performance is equivalent here to the PP+PD controller.

Since it is desired to test the capabilities of the regulation schemes listed in Table 4.1 with regards to the constraint geometry, it is important to compensate for the dynamics of the robotic device. These dynamics have been neglected until now and if left uncompensated would impact the tracking performance of the overall system. Feedback linearization controllers are used to linearize the haptic device, taking into account its dynamics so that any tracking errors can be attributed to the regulation scheme.

6.3.1 Feedback Linearization

Feedback linearization is a control technique capable of compensating for the nonlinear dynamics of robot manipulators. It establishes an inner-loop controller system that provides a new outer-loop control input so that the system is linear between its output and this new input. If an accurate dynamic model of the robot can be found and the robot has sufficient degrees of actuation, then the robot can be assigned a desired set of dynamics. If the system with dynamics (2.14) yields a full vector relative degree with respect to its output (2.16), then a transformation function $\Xi : U \to \Xi(U)$ can be constructed of the form

$$\Xi(x) = \xi = \begin{bmatrix} h_1(x) \\ L_f h_1(x) \\ \vdots \\ h_m(x) \\ L_f h_m(x) \end{bmatrix} \tag{6.5}$$

where Ξ is a diffeomorphism on U [62].

According to the State Space Exact Linearization problem, the dynamics (2.14) can be expressed as

$$\dot{\xi} = L_f^2 h(x)\Big|_{x=\Xi^{-1}(\xi)} + L_g L_f h(x)\Big|_{x=\Xi^{-1}(\xi)} u \tag{6.6}$$

under the transformation function $\Xi(x)$, where $L_f h(x)$ is the Lie derivative $L_f h(x) = \frac{\partial h}{\partial x} f(x)$ [62]. Choosing the inner-loop controller as

$$u = \left(L_g L_f h(x)\right)^{-1} \left(-L_f^2 h(x) + v\right) \tag{6.7}$$

results in a linear system of the form

$$\dot{\xi} = A\xi + Bv \tag{6.8}$$

$$y = C\xi \tag{6.9}$$

where v is the outer-loop control input and the state matrices are in the Brunovsky normal form which for 2nd order mechanical systems has the block structured

$$A = \begin{bmatrix} A_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_m \end{bmatrix}, A_i = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.10}$$

$$B = \begin{bmatrix} B_1 \\ \vdots \\ B_m \end{bmatrix}, B_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{6.11}$$

$$C = \begin{bmatrix} C_1 & \cdots & C_m \end{bmatrix}, C_i = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{6.12}$$

The structure of the linearized system comprised of (6.8) and (6.9) results in the inner-loop dynamics of

$$\ddot{y} = v \tag{6.13}$$

The outer-loop controller is then chosen as

$$v = \frac{1}{m_{tool}} F_r \tag{6.14}$$

where m_{tool} is the mass of the virtual tool and F_r is the response force generated by the renderer. Combining (6.13) and (6.14) yields the closed-loop virtual tool dynamics of

$$m_{tool}\ddot{y} = F_r \tag{6.15}$$

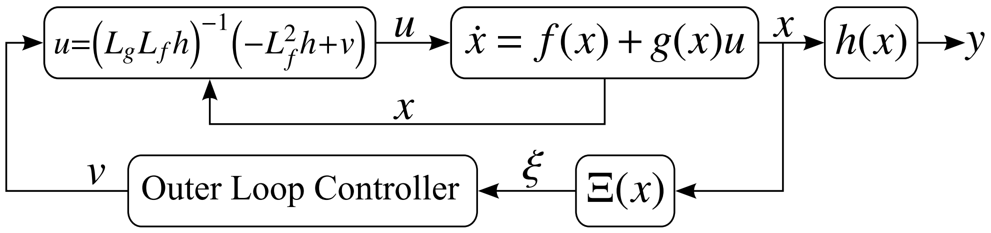


Figure 6.18: Block diagram of the state feedback linearization controller (6.7)

which is of the same form as the desired dynamics of the virtual tool in (3.1). The block diagram for the haptic system under feedback linearization control is shown in Figure 6.18.

The feedback linearization controller (6.7) with transformation function (6.5) will be well defined at those points where the system (2.14) with output (2.16) has a full vector relative degree [62]. According to Proposition 2.1 and Definition 2.4, the system will be feedback linearizable when the state is in the functional configuration space $\mathcal{D}_{x^\star}$.

Feedback Linearization of the Point Mass Robot

Because of the simple nature of this system, the feedback linearization is equally straightforward. Section 2.5.5 showed that the point robot's functional configuration space is all of $\mathbb{R}^n$ so that the feedback linearization controller is well defined everywhere. The feedback linearization controller is

$$u = m_{point} v \tag{6.16}$$

which is paired with the trivial feedback transformation function $\Xi(x) = x$. This controller is used in the simulations results throughout the rest of this chapter.

Feedback Linearization of the Parallelogram Robot

In Section 2.5.5, the functional configuration space $\mathcal{D}^{para2d}$ of the parallelogram robot was found and was shown to not be empty so that the robot is indeed feedback linearizable within a region of its workspace. For the parallelogram manipulator operated in the 2 DOF mode with dynamics

(2.35) and output (2.36), the feedback linearization controller (6.7) evaluates to

$$u = \begin{bmatrix} -\frac{\cos(x_2)d_2}{l_1\cos(x_1-x_2)} & -\frac{\sin(x_2)d_2}{l_1\cos(x_1-x_2)} \\ -\frac{\sin(x_1)d_3}{l_4\cos(x_1-x_2)} & \frac{\cos(x_1)d_3}{l_4\cos(x_1-x_2)} \end{bmatrix} \left(\begin{bmatrix} -\sin(x_1)l_1x_3^2 + l_4\cos(x_2)x_4^2 - \frac{\cos(x_1)l_1b_2x_3}{d_2} - \frac{l_4\sin(x_2)b_3x_4}{d_3} \\ \cos(x_1)l_1x_3^2 + l_4\sin(x_2)x_4^2 - \frac{\sin(x_1)l_1b_2x_3}{d_2} + \frac{l_4\cos(x_2)b_3x_4}{d_3} \end{bmatrix} + v \right) \tag{6.17}$$

which is well defined over $\mathcal{D}^{para2d}$.

The controller in (6.17) is used in the experimental results of the rest of this chapter. The control gains were chosen as $K_p = 500\,\mathrm{s}^{-1}$ and $K_d = 44.7\ \mathrm{s}^{-2}$ to produce a critically damped system likely to remain within the actuator's bounds of saturation, the sampling time was $t_s = 1\,\mathrm{ms}$, the fastest allowable by the hardware used, and the radius of the circle constraint in (6.4) was $r_c = 0.1$ m.

6.3.2 Performance on a Line Model

The dynamic performance for the various controllers are evaluated on the line constraint (6.3) using to the system's closed loop dynamics and its performance in simulations and experiments. This determines a baseline system performance against which the circle constraint case in Subsection 6.3.3 can later be compared. The closed loop system dynamics and simulations demonstrate the best possible capabilities of the various projective controllers, while the experiments validate the behaviour of the controllers when subjected to real world conditions.

Theoretical Performance on a Line Constraint

To form the closed-loop dynamics, the point of projection on the constraint and the normal / tangent vectors at that point are required. While analytical expressions for these quantities cannot be found for arbitrary models, the simple structure of the line constraint admits tractable expressions. For the line (6.3), the projection onto the line ρ_{line}, the normal vector at the point of projection n_{line} and the associated tangent vector t_{line} are:

$$\rho_{line} = \begin{bmatrix} y_1 \\ 0 \end{bmatrix}, \quad n_{line} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad t_{line} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{6.18}$$

The various pieces can all be combined to get an expression for the closed-loop dynamics of the end-effector. Taking the ideal point robot model with dynamics (2.23) and output (2.24) and applying the PD controller (4.6) with the various error definitions for the PP (4.7), PP+V (4.9), PP+D (4.11), and PP+PD (4.13) controllers each in their turn given the line model (6.3) and its associated quantities (6.18), the various sets of closed-loop dynamics for y are shown in Table 6.1 are obtained.

Table 6.1: Closed-loop dynamics of the point robot from Section 2.5.5 and line model (6.3)

Name	**Regulation Scheme**	**Closed-loop dynamics**
PP	$F_r = -K_p e_p$	$\ddot{y}_1 = 0$ $\ddot{y}_2 = -K_p y_2$ (6.19)
PP+V	$F_r = -K_p e_p - K_d \dot{y}$	$\ddot{y}_1 = -K_d \dot{y}_1$ $\ddot{y}_2 = -K_p y_2 - K_d \dot{y}_2$ (6.20)
PP+PD, PP+D	$F_r = -K_p e_p - K_d \dot{e}_p$ $F_r = -K_p e_p - K_d (n_e \cdot \dot{y}) n_e$	$\ddot{y}_1 = 0$ $\ddot{y}_2 = -K_p y_2 - K_d \dot{y}_2$ (6.21)

The dynamics in y_1 and y_2 are related to the tangential and transversal motion in the system, respectively. Since this is a frictionless surface, ideally $\ddot{y}_1 = 0$ which implies that the end-effector slides along the surface at constant velocity as determined by the initial conditions or accelerates according to the forces exerted by the external disturbance. This tangential behaviour is achieved by all but the PP+V controller, which induces a deceleration of the tangential motion according to the derivative control gain K_d. In the transversal dimension, it is desired to have the end-effector approach and remain on the line, which translates to having an equilibrium point in the system at $y_2 = 0$. Given the expressions in Table 6.1 and the fact that the control gains are positive, the PP+V, PP+PD and PP+D controllers all produce asymptotically stable transversal dynamics, while the PP controller produces dynamics that are only marginally stable. In practice, however, the system will typically be stable under the PP controller because of unmodelled (and therefore uncompensated) friction effects that dissipate energy. In fact, this is the controller that was used in the experiments of Section 6.1.

The closed-loop dynamics from Table 6.1 are simulated for the initial condition $y(0) = (-1, 0)$ and $\dot{y}(0) = (3, 10)$, which corresponds to a configuration on the constraint with velocity that will instantaneously cause the output to leave the constraint. The position in the plane and the tangential velocity are shown in Figure 6.19. The marginal stability of the proportional only controller PP appears in the oscillations around the constraint with an associated oscillation of the tangential velocity, both of which are undesirable. The PP+V controller's damping approaches the constraint, but the damping introduces a viscous friction force in the tangential dynamics that decelerates the motion. The PP+D and PP+PD controllers are collinear in Figure 6.19, both asymptotically approaching the constraint while maintaining a constant tangential velocity. The PP+D and PP+PD controllers are therefore the only ones capable of separating the transversal and tangential control dimensions to achieve the desired results.

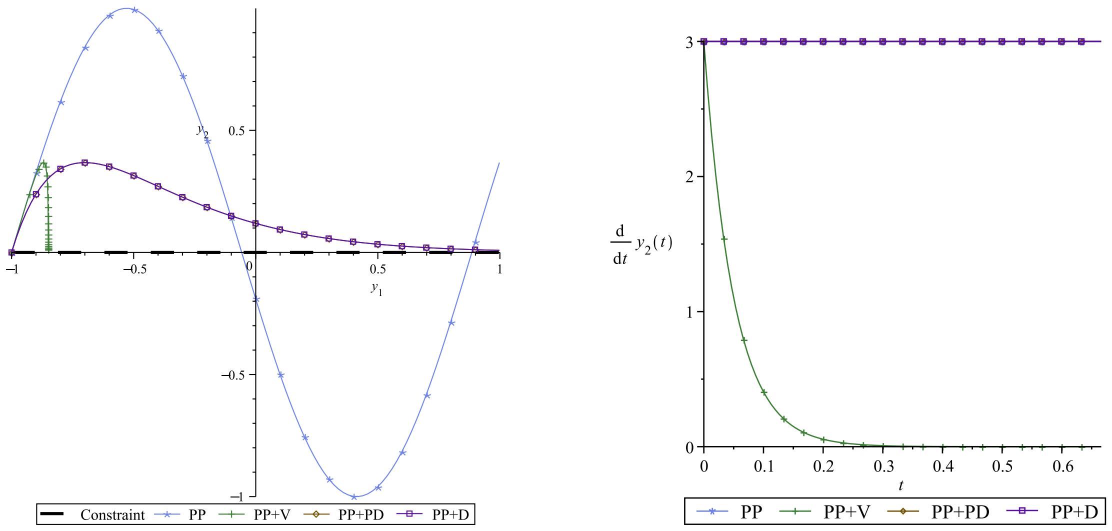


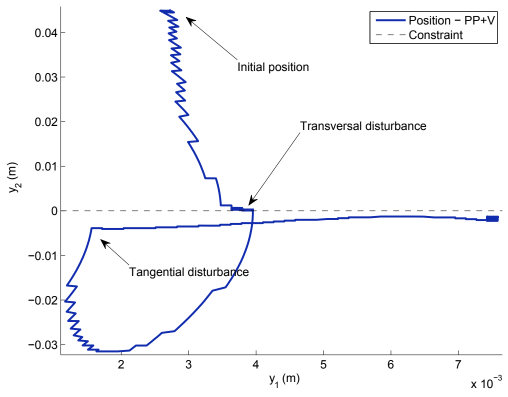
Figure 6.19: Simulation of the dynamics from Table 6.1 for the initial condition $y(0) = (-1, 0)$ and $\dot{y}(0) = (3, 10)$

Experimental Performance on the Line Constraint

The experiments were run by starting the manipulator in a configuration off of the constraint and letting the controller draw the manipulator onto the constraint. A transversal disturbance was then manually applied to verify the response of the system. When the system settled, a tangential disturbance was subsequently applied at the end-effect. The transversal control can be evaluated in the initial tracking phase of the manipulator and the desired double integrator tangential dynamics can be evaluated from the velocity profile of the system after the tangential disturbance. The experiments were run for the PP, PP+V and PP+D outer controllers under the inner-loop feedback linearization controller (6.17). Separate experiments were not conducted for the PP+PD controller on the line constraint because it was shown to be equivalent to the PP+D controller for this case in Table 6.1.

The implementation of the PP controller was found to be unstable in practice, resulting in gradually increasing end-effector velocities. This is likely due to the time delay inherent in the sampled data nature of the control system. This delay was not modelled in (6.19) and its introduction would deteriorate the marginal stability of the system into outright instability. Had the end-effector been grasped by an operator, their grip would have added damping that could have stabilized the system. The PP+V and PP+D / PP+PD controllers yielded stable systems, whose recorded positions are shown in Figure 6.20. The two systems behaved quite similarly in the first phase of the motion, with the same degree of tracking error when returning to the constraint. This confirms the theoretical findings from Table 6.1 that the two systems have the same transversal behaviour. Neither of the controllers manage to converge to zero error because of the nonlinear, configuration dependent joint frictions that were not fully captured in the robot's dynamic model (2.26). The zig-zag effects on the left side of Figure 6.20 are the result of signal quantization by the encoders and the relatively fine scale of the y_1 axis.

The main differences between the PP+V and PP+D/PP+PD methods arise in the tangential dimension. The vertical scales in Figure 6.20 are almost identical, but the horizontal scales are approximately an order of magnitude different. The transversal damping introduced by the PP+V controller greatly decelerates the tangential motion, which can be seen by the quick return to zero of $\dot{y}_1$ in the velocity plot on the left side of Figure 6.21, contrary to the system's frictionless sliding goal. The PP+D/PP+PD controllers, however, provide less resistance to the sliding along the constraint, as seen on the right side of Figure 6.21. Ideally, $\dot{y}_1$ would be a horizontal line after the disturbances are applied, but the dynamic model used sometimes over- and sometimes under-compensates the friction in the system, leading to minor accelerations and decelerations



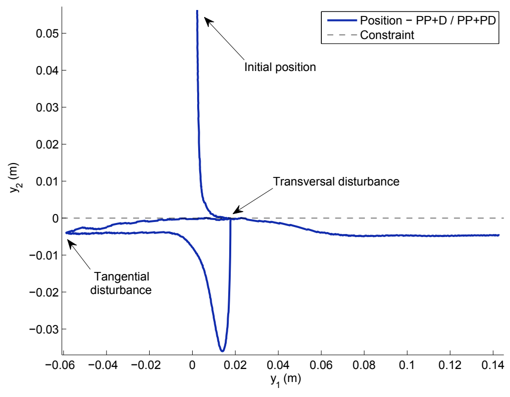
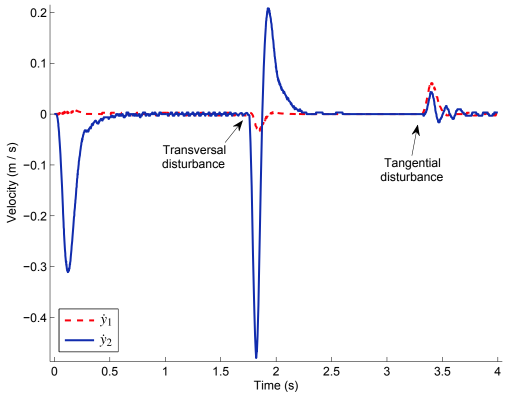


Figure 6.20: Experiments on the line constraint under the PP+V (left) and PP+D / PP+PD (right) controllers - Positions

in the tangential velocity profile as seen between $t \approx 4\,\text{s}$ and $t \approx 6\,\text{s}$ on the right of Figure 6.21.



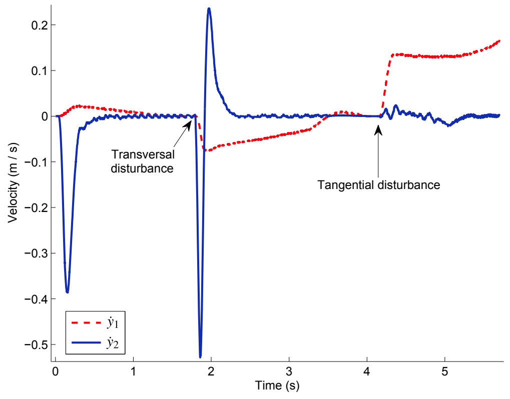


Figure 6.21: Experiments on the line constraint under the PP+V (left) and PP+D / PP+PD (right) controllers - Velocities

6.3.3 Performance on a Circle Model

The same methodology as with the line constraint from Subsection 6.3.2 is followed here to demonstrate the theoretical and practical differences between a flat and curved constraint for the various rendering strategies listed in Section 4.2.

Theoretical Performance on a Circle Constraint

The functions of interest for the circle constraint (6.4) are the projection function ρ_{circ}, the normal vector at the projection point n_{circ}, and associated tangent vector t_{circ} given by

$$\rho_{circ} = r_c \frac{y}{||y||_2}, \quad n_{circ} = \frac{y}{||y||_2}, \quad t_{circ} = \frac{1}{||y||_2} \begin{bmatrix} -y_2 \\ y_1 \end{bmatrix}. \tag{6.22}$$

The same ideal point mass robot with dynamics (2.23) and output (2.24) and PD controller (4.6) with the various error definitions for the PP (4.7), PP+V (4.9), PP+D (4.11), and PP+PD (4.13) controllers as in the line constraint case are used again. For the circle constraint, the closed-loop dynamics can be better interpreted in polar coordinates, so the change of variables

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} \tag{6.23}$$

is used. The radial coordinate represents the transversal system while the angular coordinate is the tangential system. The resulting closed-loop dynamics for the various controllers are listed in Table 6.2.

Ideally, the controllers would be able to isolate the transverse (radial) and tangential (angular) subsystems, separating the tasks of driving the end-effector onto the model and implementing desired dynamics on the model's surface. For all of the cases in Table 6.2, the expressions are a coupled system of differential equations where the radial and angular dynamics are interdependent. The radial accelerations $\ddot{r}$ contain a term dependent on the angular velocity, namely $r\dot{\theta}^2$, in all cases. This means that the radial dynamics will only have an equilibrium point at $r = r_c$ when the radial velocity $\dot{\theta}$ is zero.

Simulations of the closed-loop dynamics on the circle help illustrate their properties. The dynamics from Table 6.2 were simulated given an initial condition of $y(0) = (1,0)$ and $\dot{y}(0) = (5,0)$, representing a point on the circle with a velocity tangent to the constraint. Since there are no external disturbances from the operator, the controllers would ideally keep the end-effector on the circle, but that is not the case. The position plot on the left side of Figure 6.22 shows that the robot's trajectory leaves the circle for all of the controllers considered. These controllers do

Table 6.2: Closed-loop dynamics of the point robot from Section 2.5.5 and circle model (6.4)

Name	Regulation Scheme	Closed-loop dynamics
PP	$F_r = -K_p e_p$	$\ddot{r} = -K_p(r - r_c) + r\dot{\theta}^2$ $\ddot{\theta} = -\frac{2\dot{r}\dot{\theta}}{r}$ (6.24)
PP+V	$F_r = -K_p e_p - K_d \dot{y}$	$\ddot{r} = -K_p(r - r_c) + r\dot{\theta}^2 - K_d\dot{r}$ $\ddot{\theta} = -\frac{(2\dot{r} + K_d r)\dot{\theta}}{r}$ (6.25)
PP+PD	$F_r = -K_p e_p - K_d(n_e \cdot \dot{y})n_e$	$\ddot{r} = -K_p(r - r_c) + r\dot{\theta}^2 - K_d\dot{r}$ $\ddot{\theta} = -\frac{2\dot{r}\dot{\theta}}{r}$ (6.26)
PP+D	$F_r = -K_p e_p - K_d \dot{e}_p$	$\ddot{r} = -K_p(r - r_c) + r\dot{\theta}^2 - K_d\dot{r}$ $\ddot{\theta} = -\frac{(2\dot{r} + K_d(r - r_c))\dot{\theta}}{r}$ (6.27)

not provide the required radial acceleration required for a circular orbit, relying on the position error to induce an inwards acceleration. The PP+V and PP+D controllers both approach the circle constraint through the damping they induce in the tangential dimension, which slows the angular velocity. This allows the radial dynamics in (6.25) and (6.27) to converge to the desired equilibrium point at $r = r_c$ by asymptotically reducing the impact of centripetal effects. The reduction in angular velocity, however, is contrary to the desired frictionless behaviour on the circle. The PP+V and PP+D controllers are thus both asymptotically stable, but couple the transversal and tangential systems.

The PP and PP+PD controllers both orbit the circle without ever settling on its surface. The PP controller exhibits oscillations in both the transversal and tangential dynamics much like in the line case due to the marginally stable nature of the controller. The PP+PD controller enters a stable orbit at a radius greater than r_c so that the radial position error balances out the centripetal term $r\dot{\theta}^2$ in the radial dynamics.

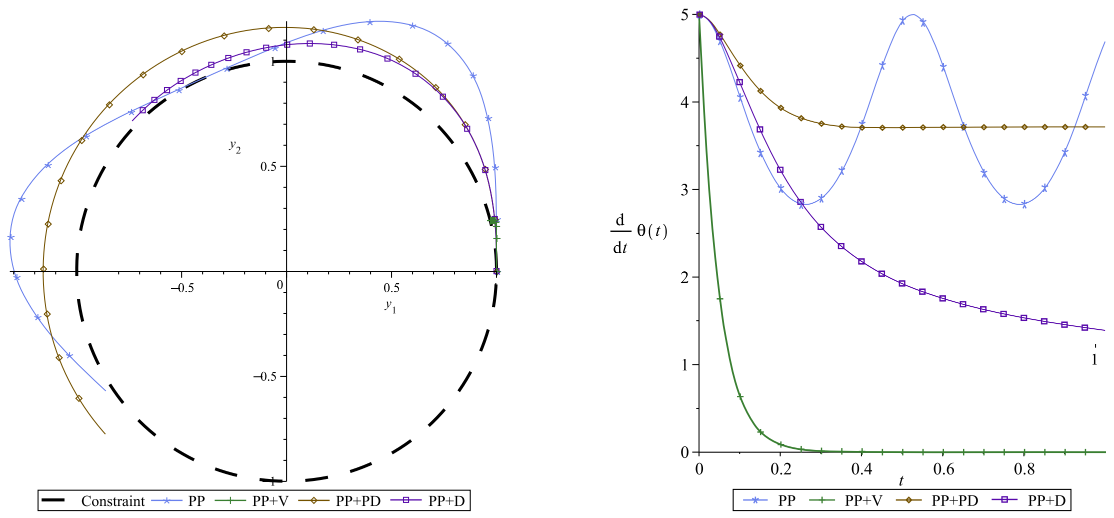


Figure 6.22: Simulation of the dynamics from Table 6.2 for the initial condition $y(0) = (1, 0)$ and $\dot{y}(0) = (0, 5)$

Experimental Evaluation on the Circle Constraint

The experimental protocol on the circle constraint begins with the end-effector on the circle. A tangential disturbance is manually applied to the end-effector at $t = 0\,\mathrm{s}$ and the subsequent motion is recorded, corresponding to the same situation as in the simulation results of Figure 6.22. The results are plotted in Cartesian space to show the transversal error's properties and the angular velocities are plotted to evaluate the tangential velocity, where the introduction of the disturbance occurs. The experiment was conducted for the PP, PP+V, PP+D and PP+PD outer controllers under the feedback linearization controller (6.17). The PP+D and PP+PD are not equivalent on the circle constraint as was the case on the line constraint, and so they are tested separately here.

The PP controller was stable on the circle constraint while it was unstable on the line constraint. The opposite is true for the PP+V controller, being stable on the line but unstable on the circle. This is contrary to the goal of creating a haptic renderer capable of simulating models with arbitrary geometry. The positions and tangential velocities for the PP, PP+D and PP+PD methods are shown in Figure 6.23. The performance of all the methods is relatively similar. They all exhibit radial oscillations around the circle constraint of roughly the same magnitude. The experiments were run much longer then the single revolution shown in Figure 6.23, with subsequent rotations around the constraint demonstrating a continuation of the oscillations without decay or increase. Although the simulated results from Subsection 6.3.3 showed that the PP+V, PP+D and PP+PD controllers achieved asymptotic stability, this is clearly not the case in practice.

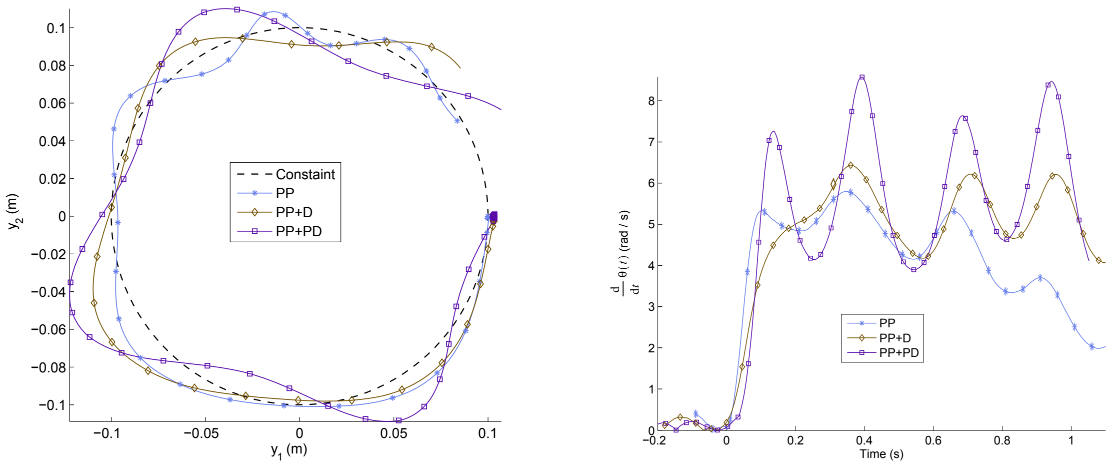


Figure 6.23: Experiments on the circle constraint under various controllers - positions (left) & tangential velocities (right)

Similar oscillations can be seen in the tangential velocities on the right of Figure 6.23. The oscillations are in sync with those of the radial error, with greater velocities in the tangential direction occurring when the end-effector is farthest from the circle's center. Ironically, the only controller whose tangential velocity shows signs of slowing down is the PP controller, which is only theoretically marginally stable in the transversal dimension. For all of the controllers, the oscillatory behaviour is contrary to the goal of frictionless sliding as it introduces periods of acceleration and deceleration as the tangential velocity fluctuates.

6.3.4 Summary of the Impact of the Regulation Scheme on Performance

While the PP+D and PP+PD controllers can achieve the desired objectives on the line constraint, they fail to so on the circle constraint. The controllers presented thus far were all formulated using a local planar approximation to the surface at the point of contact. This planar simplification of the surface works well for linear constraints like the line, but fails to take into account the higher order characteristics of curved constraints like the circle. The curvature of the constraint introduces a nonlinearity into the control problem, which is unaccounted for by the linear controllers. It is for this reason that the none of the controllers were able to achieve both asymptotic stability in the transversal dimension and frictionless sliding in the tangential dimension on the circle.

Relying on the position error to provide the velocity dependent centripetal acceleration distorts the character of the model. The tests conducted for small penetrations on the Cassinian model in Figure 6.4 experienced an increase in position error after passing under the valley between lobes. In reality, this region would supply an outwards centripetal force so that the tool would follow the curve of the surface. Since the controller does not provide this outwards centripetal force, the HIP position tends to drift inwards. This has an impact on the perception of the surface's curvature, making concave regions feel more exaggerated and convex regions feel more flat.

6.4 Summary of Renderer Evaluations

There is no single renderer that performs perfectly across all of the interaction scenarios considered. The pop-through artifact that renders curves as corners or allows the tool to traverse through the model were due to singularities in the projection function of the renderer. While the renderers will all perform well when the haptic system is capable of rendering very large stiffnesses, minimizing the penetration depth of the IHIP, the renderers exhibit various artifacts as the penetration depth increases. Table 6.3 provides an overall summary of the renderers' performance in different situations.

The constraint renderer performs well on angular models, where the enforcement of local constraints aligns well with the form of the model. On curved models, however, the constraint renderer exhibits pop-through whose severity increases with penetration depth. The mapping renderer maintains a smooth response force on curved models, avoiding pop-through for greater penetration depths than the constraint renderer. The mapping renderer is still subject to pop-through across corners or on thin model segments, however. The minimum distance renderer

Table 6.3: Summary of the Renderers' Performance

	Minimum Distance	Constraint	Mapping
Resists Pop-Through on Curves	No	No	Yes
Resists Pop-Through on Corners	No	Yes	No
Suitable for LUT	Yes	No	Yes
Accounts for Centripetal Acceleration	No	No	No

performed no better than the constraint or mapping renderer in any of the test scenarios. It presents no benefits in terms of better simulating phenomena, but offers a much simpler implementation on certain models forms, especially for the rendering of implicit models.

The regulation schemes all performed well on flat surfaces, but were found to suffer from velocity dependent errors on curved surfaces. They failed to explicitly provide the centripetal acceleration necessary to maintain the haptic device on the model surface, using the PD controller to compensate for ensuing position errors. This distorts the curvature of the model, reducing the fidelity of the haptic simulator. Without explicitly accounting for model curvature, it was impossible to separate the system into transversal and tangential subsystems. Such a separation is required to meet the control goals of stabilizing the surface and implementing the desired friction dynamics along the surface. The next chapter introduces a nonlinear controller better suited to address the nonlinear nature of curved constraints.

While all of the regulations schemes, except for the PP controller, performed well on the line constraint, none of them produced an asymptotically stable constraint in the transversal dimension and none of them could recreate the dynamics of a double integrator in the tangential dimension on the circle constraint. The curved nature of the constraints introduce additional dynamics related to the spatial variations of the constraint that must also be taken into account, as will be shown in the next chapter.

Chapter 7

Set Stabilization and Haptics

The haptic renderers presented in Chapters 4 and 5 project onto the surface of the model and then regulate to that projection point. Because their definition is so intimately linked to the projection function, these renderers will be referred to as the projective renderers or projective controllers. They use a planar approximation to the surface at the point of projection, incorporating the surface's normal and / or tangent vectors, but ignores higher order characteristics like curvature. Neglecting the surface's curvature resulted in the uncompensated terms in the closed-loop systems formed by the projective controllers of Table 6.2. An alternative method is therefore required that incorporates the curvature information into the controller.

An alternate approach to the haptics problem is proposed in this chapter, referred to as a set stabilization controller. It draws from the virtual constraints literature where the effects of constraint curvature have been considered in the trajectory and path following problems. One application for virtual constraints are human-machine cooperative tasks that share many similarities to haptic simulators. Their adaptation to the haptics problem is considered, with the presentation here drawing from previous publications [8, 10].

The set stabilization removes the separation of design elements that was used in previous chapters. The problem was separate into an inner-loop controller that made the haptic device appear Cartesian and calculate the response force using the haptic renderer that could ignore the device's characteristics. The set stabilization approach merges the device's control with the task's constraint so that the controller can generate the centripetal accelerations on the part of the haptic device necessary to follow the surface's curvature. Similar results could be obtained by adding a centripetal type feed-forward term to the regulation schemes of previous haptic renderers, but the set stabilization method is framed to better take advantage of previous control works. Formulating the problem in the context of differential geometry and the geometric understand of the control problem allows previous results to be leveraged in proving the stability and in effectuating the design of the controller.

The set stabilization procedure is introduced first for one dimensional constraints in a two dimensional space in Section 7.3. The controller linearizes the system with respect to a virtual output function composed of a transversal and a tangential coordinate. This technique decouples the haptic system into transversal and tangential subsystems so that the control objectives can be met. Section 7.4 extends these results to two dimensional constraints in three dimensions where a single set of tangential coordinates no longer suffices to cover the surface. The controller must then switch between different sets of tangential coordinates to form a controller that is well defined over the entire model surface.

7.1 Virtual Constraints

Virtual constraints consist of a desired relationship among the states and / or output of a system to be enforced by the controller, represented by a set of equations. They are virtual because the desired relationship does not arise from the physics of the system, but stems from the application's objectives. The constraints formalize the task requirements mathematically so that they may be incorporated into control design and system analysis. Virtual constraints appear in a variety of areas in the robotics literature including hybrid position / force control [117], teleoperation [118], haptic simulators [44], cooperative manipulation [119] and assisted manipulation [120]. All of these applications involve conceptually separating the control objectives according to task-based dimensions parallel and perpendicular to the constraint.

Virtual holonomic constraints, which have the form $z(x) = 0$ where x is the system's state vector and z is the constraint function, have been investigated by various authors as a tool for control design [121, 122]. The haptics problem considers unilateral constraints of the form $z(x) > 0$, such that the controller is enforcing the constraint formed by the model's surface only when the output is inside of the model. This would involve the switching between two different control schemes on either side of the model's boundary. Inside the model, the controller would try to reduce the penetration depth and assign the desired surface dynamics while outside the model, the system would attempt to recreate the virtual tool dynamics. Even though each individual controller would be stable, care must be taken when switching between them to ensure that the overall system is stable [123]. The proof of stability in the face of switching forms part of the future work. Like in Section 6.3, the control design here is performed for the holonomic case $z(x) = 0$, with the conversion to a unilateral constraint involving simply turning the controller off at a certain point (or switching to another free space controller that would compensate for the device's friction and inertia).

A common approach to implementing virtual constraints in human-robot cooperative tasks is to project the system output onto the set defined by constraint functions. The local geometric properties of the set defined by the constraint functions naturally generate transversal and tangential directions of motions with respect to the constraint set. This is the approach commonly found in the projective controllers in the existing haptics works from Chapter 4 and Chapter 5. Admittance controllers commands velocities that are compatible with the constraint based on the local normal and tangent vector at the point of projection [124] while impedance controllers exert forces aligned with the task's dimensions to drive the end-effector towards the projection point [46]. Many examples consider only linear constraints [124, 46] or linearize the constraint functions by neglecting higher order terms in nonlinear constraints [47], but the importance of incorporating the constraint's second order differential properties is well known [125]. Without accounting for the curvature of the constraint, the controller relies on the position error to induce the centripetal accelerations in the haptic device necessary to follow the constraint, precluding asymptotic stability in certain cases as seen in Section 6.3 and as reported in [8].

Path following, which is more commonly found in mobile robotics, has some affinity to virtual constraints. The path following problem consists of making a control system's output approach and follow a path with no *a priori* time-parameterization. It can be viewed as the stabilization of a one dimensional constraint, i.e. curves in the system's output space. One approach to the path following problem is to view the path as set to be stabilized, which allows for the formalization of many concepts which have been discussed in less precise terms in previous chapters. From a set stabilization point of view, the controller can ensure output invariance of the path, meaning that if the output is initialized on the path appropriately, it remains on the path indefinitely.

Path following using set stabilization has been applied to cranes [126], bipedal robots [127, 128], and bicycle riding [129]. One approach to set stabilization is transverse feedback linearization (TFL). Given a target set, TFL seeks to feedback linearize a system's transversal dynamics, i.e. that portion of a system's dynamics that determine whether or not the system is approaching the target set. TFL has been applied to planar vertical take-off and landing aircraft [130], magnetic levitation positioning systems [131], and manipulators with flexible links [132]. Despite the single sided name, TFL seeks to linearize both the transversal and tangential subsystems through a state feedback controller where permitted by the system's structure and degree of actuation [133].

The rest of this chapter builds on the results of [132], which considered the stabilization of a parameterized curve in the output space. Both transversal and tangential dimensions are simultaneously feedback linearized. Where [132] considered velocity tracking in the tangential dimension, dynamic assignment on the tangential dimension is performed here. The work of [132] applied to one dimensional constraints that can be covered by a single coordinate chart, while the latter part of this chapter extends this work to closed two dimensional constraints that require switching between multiple coordinate charts [10].

7.2 Set Stabilization

Three separate spaces arise in the haptics problem: the joint space, the output space and the task space. Robotic control is performed most naturally in the joint space since the hardware measures joint configurations and commands joint torques and since the manipulator's singularities are a function of the robot's configuration. Virtual objects are naturally modelled in the output space, which is typically the space where the models are crafted or acquired. The contact forces are most naturally expressed in yet another space that separates the system's dimensions according to the directions across (transversal) or along (tangential) to the constraint. The projective controllers of Chapters 4 and 5 incorporate information from the task space by using the normal vector at the point of contact to separate the position and velocity errors into components aligned with the surface. The curvature at the point of contact, however, constitutes additional information from the task space that is ignored by the projective controllers.

Set stabilization presents an alternative approach where the surface of the model is considered a set of points to be stabilized, fully incorporating the task space information into the control formulation. The model's surface comprises a set of points $\mathcal{S} \subset \mathbb{R}^N$ in the output space of the haptic device. Functions representing the distance to the model and coordinates along the model's surface assimilate the model's structure into the controller. These quantities are pulled back from the task and output spaces through nonlinear, differentiable maps into the joint space, providing a state feedback controller with the necessary information to solve the control problem posed in this section. The end result is a nonlinear state feedback linearizing controller that takes into account the model's 2nd order differential properties.

The control problem can only be solved if it is well formulated. Since the models represent physical objects, their surfaces are closed sets, that is compact and boundaryless [134]. From a geometric point of view, the set $\mathcal{S}$ is a closed curve for a model in a two dimensional output space and a closed surface in a three dimensions output space. As a basic necessity, the model must lie within the functional workspace of the manipulator so that the haptic device can be used over the domain of interest.

Assumption 7.1. *Let $\mathcal{S}$ be the set of points forming the constraint. There exists an $x^\star = (x_c^\star, x_v^\star) \in \mathbb{R}^n$ such that $\mathcal{S} \subset \mathcal{F}_{y^\star} = h(\mathcal{D}_{x^\star}, x_v^\star)$ for some $\mathcal{D}_{x^\star} \subseteq \mathbb{R}^N$. In other words, the constraint to be stabilized is a subset of the functional workspace of the system with dynamics (2.14) and output (2.16).*

Assumption 7.1 guarantees that for each y in a neighborhood of $\mathcal{S}$, there exists a unique configuration $x_c \in \mathcal{D}_{x^\star}$ such that $y = h(x_c, \cdot)$ and that $\mathcal{S}$ does not contain any singular points of $\frac{\partial h}{\partial x}$. The set stabilization approach as applied here seeks to simultaneously stabilize the set $\mathcal{S}$ and assign the system's dynamics along the set. The control problem is formally defined as

Problem 7.1. *Given a mechanical system of the form (2.14) with output (2.16), a set $\mathcal{S}$ that satisfies Assumption 7.1 and a set of desired dynamics for the output $y(t)$ restricted to the set $\mathcal{S}$*

$$(\ddot{y} - a(y, \dot{y}))\big|_{\mathcal{S}} = 0 \tag{7.1}$$

with $a : \mathcal{S} \times \mathbb{R}^p \to \mathbb{R}^p$ a C^1 function. Find, if possible, a smooth feedback control law $u : \mathbb{R}^n \to \mathbb{R}^m$ such that

(a) There exists an open set $U \subseteq \mathbb{R}^n$ such that $\mathcal{S} \subset h(U)$ and for each initial condition $x(t_0) \in U$, the solution $x(t)$ to the corresponding closed-loop system is such that $h(x(t)) \to \mathcal{S}$ as $t \to \infty$.

(b) The set $\mathcal{S}$ is output invariant for the closed-loop system.

(c) For each initial condition $x(t_0) \in U$ the closed-loop output trajectory $y(t) = h(x(t))$ satisfies $\ddot{y}(t) \to a(y(t), \dot{y}(t))$ as $t \to \infty$.

The rest of this section is devoted to designing a state feedback controller $u(x)$ that solves Problem 7.1.

Mapping the various quantities and sets back and forth across three different domains generates a plethora of function definitions. Figure 7.1 illustrates the relationships among the various functions to be formally defined in this chapter. The sets labelled in Figure 7.1 are the configuration space $\mathcal{C}$, reachable workspace $\mathcal{R}$, the functional configuration space $\mathcal{D}_0$, and the functional workspace $\mathcal{F}_0$ of the manipulator defined in Section 2.5.

The set stabilization approach considers the motion of the system transversal to the model through a function $s : \mathbb{R}^N \to \mathbb{R}$. The function s provides a measure of "distance" from the output y (the HIP) to the set $\mathcal{S}$ (model's surface). The surface is thus embedded in s as the zero level set

$$\mathcal{S} = \left\{ y \in \mathbb{R}^N : s(y) = 0 \right\}, \tag{7.2}$$

where s arises directly out of the implicit definition of model or is calculable for other model representations. The "distance" defined by s is not necessarily a Euclidean distance, e.g. the square of the distance is used in [63] to simplify later calculations. One candidate for s is the

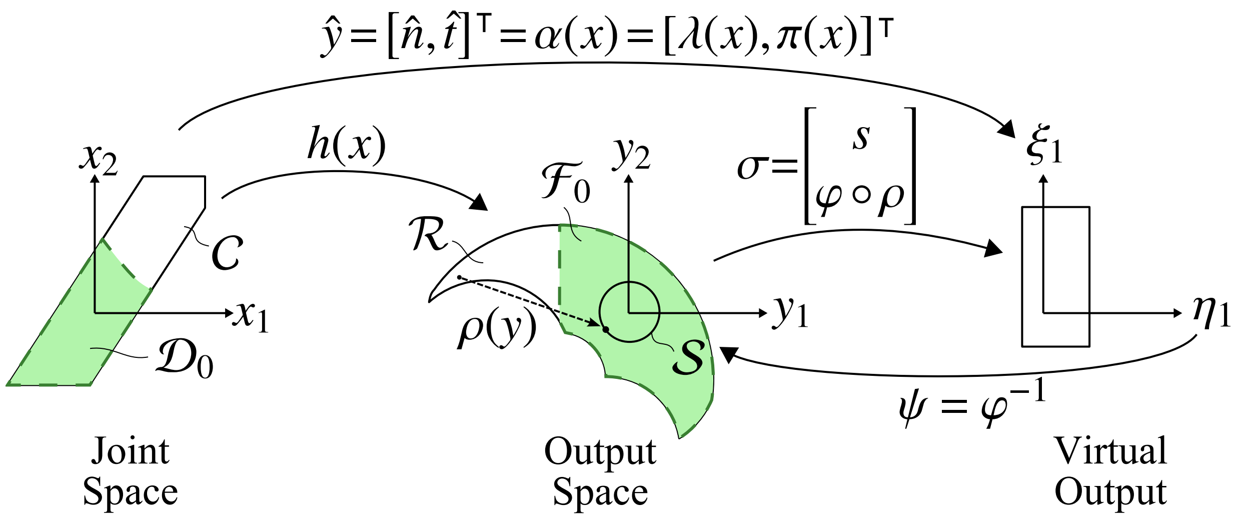


Figure 7.1: The relationships between the defined sets, functions and spaces for the set stabilization controllers

signed distance function of the model. The distance function with respect to the state space $\lambda : \mathcal{D} \to \mathbb{R}$ is defined as

$$\hat{n} = \lambda(x) = s \circ h(x), \tag{7.3}$$

and gives the system's transversal coordinate $\hat{n} = \lambda(x)$. The transversal dynamics are then comprised of $\hat{n}$ and its derivatives.

The model's surface is assumed to admit a coordinate system (or set of coordinates systems later in Section 7.4) that describes the location of the contact point on $\mathcal{S}$. The tangential coordinate $\hat{t}$ is given by the function $\varphi : \mathcal{S} \to \mathbb{R}^{N-1}$, which is only defined on the surface of the model. To determine the tangential coordinate of an arbitrary point in the functional workspace $\mathcal{F}$, φ is composed with a projection function $\rho : \mathbb{R}^N \to \mathcal{S}$. Composed with the forward kinematics, the function $\pi : \mathcal{D} \to \mathbb{R}^{N-1}$ maps from the state variable x to the tangential coordinate $\hat{t}$.

$$\hat{t} = \pi(x) = \varphi \circ \rho \circ h(x). \tag{7.4}$$

Set stabilization is based on feedback linearizing the haptic device with respect these coordinates, grouped together to form the virtual output $\hat{y}$

$$\hat{y} = \begin{bmatrix} \hat{n} \\ \hat{t} \end{bmatrix} = \sigma(y) = \begin{bmatrix} s(y) \\ \varphi \circ \rho(y) \end{bmatrix} \tag{7.5}$$

$$= \alpha(x) = \begin{bmatrix} s \circ h(x) \\ \varphi \circ \rho \circ h(x) \end{bmatrix} = \begin{bmatrix} \lambda(x) \\ \pi(x) \end{bmatrix}, \tag{7.6}$$

where $\sigma : \mathcal{F} \to \mathbb{R}^N$ maps the functional workspace space and $\alpha : \mathcal{D} \to \mathbb{R}^N$ maps the functional configuration space to the virtual output space. The virtual state variables ξ and η are related to the virtual output as $\xi_1 = \hat{n}$ and $\eta_1 = \hat{t}_1$, $\eta_3 = \hat{t}_2$, etc., since the virtual state variables are comprised of both positions and velocities. The functions $\lambda(x)$ and $\pi(x)$ define transversal and tangential coordinates whose separation of the task space arises from the structure of $\mathcal{S}$. By feedback linearizing the dynamics of the robot in (2.14) with respect to the virtual output (7.6), the objectives of Problem 7.1 can be addressed in a straightforward manner in the linearized coordinates.

The system (2.14) must have a full vector relative degree with respect to the virtual output (7.6) in order to apply a feedback linearization controller.

Proposition 7.1. *Mechanical systems with dynamics (2.14) have a full vector relative degree with respect to the virtual output (7.6) for $x \in (\mathcal{D} \cap \mathcal{M}) \times \mathbb{R}^N$ where*

$$\mathcal{M} = \left\{ x_c \in \mathbb{R}^N : \operatorname{rank}\left(\frac{\partial \sigma(y)}{\partial y} \right)\Bigg|_{y=h(x_c,\cdot)} = N \right\} \tag{7.7}$$

Proof. The proof follows that of Proposition 2.1 closely, applying the chain rule and the block structure of the system's matrices to derive the full vector relative degree conditions specific to this application. A new nonlinear state space system is defined with the same dynamics (2.14) but with a new output consisting of $\alpha(x)$ in (7.6). According to Definition 2.2, if $L_g\alpha = \mathbf{0}$ then the system will have a full relative degree where $L_gL_f\alpha$ is full rank.

The block structure of the mechanical system's matrices in (2.14) ensures that

$$\begin{aligned} L_g\alpha =& L_g\sigma(h(x)) \\ =& \left.\frac{\partial\sigma(y)}{\partial y}\right|_{y=h(x)} \frac{\partial h(x)}{\partial x} g(x) \\ =& \mathbf{0} \end{aligned}$$

for all $x \in \mathbb{R}^n$.

The decoupling matrix will have the form

$$\begin{aligned} L_gL_f\alpha =& L_g \left.\frac{\partial\sigma(y)}{\partial y}\right|_{y=h(x)} \frac{\partial h(x)}{\partial x} f(x) \\ =& L_g \left.\frac{\partial\sigma(y)}{\partial y}\right|_{y=h(x)} \frac{\partial h(x)}{\partial x_c} x_v \\ =& \begin{bmatrix} \left(\frac{\partial}{\partial x_c}\left(\left.\frac{\partial\sigma(y)}{\partial y}\right|_{y=h(x)} \frac{\partial h(x)}{\partial x_c} x_v\right)\right) & \left(\left.\frac{\partial\sigma(y)}{\partial y}\right|_{y=h(x)} \frac{\partial h(x)}{\partial x_c}\right) \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ g_v(x_c) \end{bmatrix} \\ =& \left.\frac{\partial\sigma(y)}{\partial y}\right|_{y=h(x)} \frac{\partial h(x)}{\partial x_c} g_v(x_c) \end{aligned} \tag{7.8}$$

According to Proposition 2.1, the product of the two rightmost matrices have full rank when $x \in \mathcal{D}$. Therefore, $\operatorname{rank}\left(L_gL_f\alpha\right) = N$ for $x \in \mathcal{D}$ and the system will have a full relative degree when

$$\operatorname{rank}\frac{\partial\sigma(y)}{\partial y} = N. \tag{7.9}$$

□

Since $\mathcal{M}$ requires a non-singular Jacobian of the virtual output $\hat{y}$ with respect to the output y, this means that the functions s and ρ need to have at least C^1 continuity. Chapter 4 discussed how the minimum distance projection is discontinuous across the medial axis of the model, so the set $\mathcal{M}$ will normally be only a subset of the model interior. The maximization of the set $\mathcal{M}$, and hence the minimization of the non-smooth points within the model, is discussed in the future research of Section 8.2.

The set where the virtual output has full vector relative degree $\mathcal{E}$ is therefore given by the intersection of the points in $\mathcal{D}$ with those in $\mathcal{M}$, i.e. $\mathcal{E} = \mathcal{D} \cap \mathcal{M}$. It is the set where the robot is singularity free and the transversal and tangential coordinates are well defined. The exact input-output feedback linearization procedure from Subsection 6.3.1 can then be applied to the system with dynamics (2.14) with respect to the virtual output $\hat{y}$ for $x \in \mathcal{E} \times \mathbb{R}^N$. Due to the different control objectives in the transversal and tangential dimensions from Problem 7.1, the linearized system is organized into separate subsystems for the set stabilization approach. The coordinate transformation associated with the set stabilization feedback linearization Ξ_{SSC}, partitioned according to the transversal coordinate ξ and the tangential coordinate(s) η, is

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \Xi_{SSC}(x) = \begin{bmatrix} \lambda(x) \\ L_f\lambda(x) \\ \pi_1(x) \\ L_f\pi_1(x) \\ \vdots \\ \pi_{N-1}(x) \\ L_f\pi_{N-1}(x) \end{bmatrix}. \tag{7.10}$$

The feedback linearizing set stabilization controller parallels that of the feedback linearization controller in (6.7), exchanging y for $\hat{y}$ to yield

$$u = \left(L_g L_f \alpha(x)\right)^{-1} \left(-L_f^2 \alpha(x) + v\right). \tag{7.11}$$

The dynamics of the resulting inner-loop system can be grouped according to the transversal and tangential elements in Ξ_{SSC}, forming the linearized subsystems

$$\begin{aligned} \dot{\eta} &= A^{\parallel}\eta + B^{\parallel}v^{\parallel} \\ \dot{\xi} &= A^{\perp}\xi + B^{\perp}v^{\perp} \end{aligned}, \tag{7.12}$$

where $(A^{\perp}, B^{\perp})$ and $\left(A^{\parallel}, B^{\parallel}\right)$ are controllable pairs in the Brunovsky normal form and the outer-loop control input v is structured as $v = [v^{\parallel}, v^{\perp}]^{\intercal}$, with transversal $v^{\parallel} \in \mathbb{R}$ and tangential $v^{\perp} \in \mathbb{R}^{N-1}$ elements [61]. The block diagram for the set stabilization controlled system is given by Figure 7.2, which closely resembles the standard feedback linearization system in Figure 6.18.

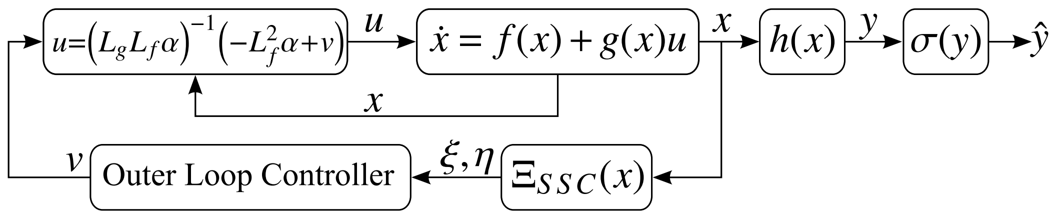


Figure 7.2: Block diagram of the set stabilization controller (7.11)

The transversal subsystem has only one input and output and so the ξ dynamics simplify the SISO system

$$\ddot{\xi}_1 = v^{\perp}\,. \tag{7.13}$$

When the outer-loop transversal controller is chosen as the PD controller

$$v^{\perp} = K_p \xi_1 + K_d \xi_2\,, \tag{7.14}$$

the transversal dynamics are stabilized for any choice of $K_p, K_d > 0$. It should be noted that the sampled data nature of the system has not been modelled here, so there will be an upper limit on the gains K_p and K_d in practice. The outer-loop controller (7.14) is independent of the robot dynamics and the virtual output and so remains constant across all of the applications of the set stabilization approach in this thesis.

The stabilization of the transversal dynamics in (7.12) is equivalent to making the set $\mathcal{S}$ output invariant [61]. This means that once the output reaches the set $\mathcal{S}$, it will remain on the set indefinitely unless acted upon by an external disturbance. In the state space, making $\xi = 0$ an equilibrium point corresponds to making the set Γ

$$\Gamma = \Big\{x : \lambda(x) = L_f\lambda(x) = 0\Big\}, \tag{7.15}$$

invariant. Γ is called the maximal invariant controlled set and is composed of the model's surface $\lambda(x) = 0$ and the points on the surface with velocity tangent to the constraint $L_f\lambda(x) = 0$. It constitutes the path following manifold formally defined in [133].

The tangential subsystem from (7.12) can be assigned the desired dynamics since $\left(A^{\parallel}, B^{\parallel}\right)$ is also a controllable pair. The desired tangential dynamics on the constraint have the form

$$\ddot{y} = a(y, \dot{y}),\; y \in \mathcal{S}\,. \tag{7.16}$$

The function $a(y,\dot{y})$ models the physical phenomena related to the sliding contact and is expressed in the output space. This facilitates the control design as the friction model can be applied directly here without modification to suit the control method. For example, the desired tangential dynamics for frictionless sliding are

$$a(y,\dot{y}) = 0\,, \tag{7.17}$$

while the viscous friction dynamics, in contrast, are

$$a(y,\dot{y}) = \frac{K_v}{m_{tool}}\dot{y}\,, \tag{7.18}$$

where m_{tool} is the mass of the virtual tool and K_v the coefficient of viscous friction.

To determine the form of the outer-loop tangential controller, the general form of the dynamics in the output space needs to be translated into the domain of the tangential controller to design $v^{\parallel}$. To this end, the inverse of the tangential coordinate $\psi = \varphi^{-1}$ is defined, which finds the projection point on $\mathcal{S}$ based on the tangential coordinate, i.e. $y = \psi(\hat{t})$. Since φ forms part of the diffeomorphism (6.5) defined on U, its inverse ψ will be well defined.

The velocity and acceleration in the output can be expressed with respect to the tangential virtual output as

$$\frac{dy}{dt} = \frac{d\psi}{d\hat{t}}\frac{d\hat{t}}{dt}\,, \tag{7.19}$$

$$\frac{d^2y}{dt^2} = \frac{d^2\psi}{d\hat{t}^2}\left(\frac{d\hat{t}}{dt}\right)^2 + \frac{d\psi}{d\hat{t}}\frac{d^2\hat{t}}{dt^2}\,. \tag{7.20}$$

Isolating for the acceleration in the tangential coordinates in (7.20) gives

$$\frac{d^2\hat{t}}{dt^2} = \left(\left(\frac{d\psi}{d\hat{t}}^{\intercal}\frac{d\psi}{d\hat{t}}\right)^{-1}\frac{d\psi}{d\hat{t}}^{\intercal}\right)\left(\frac{d^2y}{dt^2} - \frac{d^2\psi}{d\hat{t}^2}\left(\frac{d\hat{t}}{dt}\right)^2\right). \tag{7.21}$$

Since the linearized tangential subsystem is equivalent to $\frac{d^2\hat{t}}{dt^2} = v^{\parallel}$, choosing the outer-loop controller as

$$v^{\parallel} = \left(\left(\frac{d\psi}{d\hat{t}}^{\intercal}\frac{d\psi}{d\hat{t}}\right)^{-1}\frac{d\psi}{d\hat{t}}^{\intercal}\right)\left(a\left(\psi(\hat{t}),\frac{d\psi}{d\hat{t}}\frac{d\hat{t}}{dt}\right) - \frac{d^2\psi}{d\hat{t}^2}\left(\frac{d\hat{t}}{dt}\right)^2\right) \tag{7.22}$$

yields the desired relationship (7.16). The outer-loop tangential controller (7.22) is independent of the robot's dynamics and output function again, but unlike the transversal controller is dependent on the virtual output (through the inverse of the tangential coordinates ψ) and is thus model-specific.

7.3 Application of Set Stabilization to a Curve in 2D

The set stabilization approach outline abstractly in Section 7.2 is now applied to a one dimensional constraint curve in a two dimensional output space. The development here parallels that of Section 6.3 so that a comparative analysis can be conducted. The set stabilization controller is investigated theoretically for the line constraint, where it is shown to be identical to the PP+D and PP+PD controllers, reinforcing the equivalency of the controllers on flat constraints. The differences between the methods appear when considering the circle constraint, where it is shown theoretically that only the set stabilization controller manages to decouple the two control objectives, successfully stabilizing the constraint and implementing the desired dynamics on the constraint. Experiments confirm that the set stabilization controller outperforms the other controllers, exhibiting smaller errors when sliding around a circle.

7.3.1 Set Stabilization on the Line

The line constraint (6.3) has simple expressions for the distance, projection and tangential coordinate functions. The signed distance to the line is $s_{line}(y) = y_2$, the projection function is $\rho_{line}(y) = [y_1, 0]^\top$ and the coordinate along the line is $\varphi_{line}(y) = y_1$. The virtual output for the point robot on the line constraint is therefore

$$\hat{y} = \sigma_{line}(y) = \begin{bmatrix} s_{line} \\ \varphi_{line} \circ \rho_{line} \end{bmatrix} = \begin{bmatrix} y_2 \\ y_1 \end{bmatrix}. \tag{7.23}$$

The Jacobian of the virtual output (7.23) with respect to the output y, $\frac{\partial \sigma_{line}}{\partial y}$, has full rank for all $x \in \mathbb{R}^4$. Therefore $\mathcal{M}^{line} = \mathbb{R}^2$ and it is only the robot's singularities that will restrict the domain of the set stabilization controller according to Proposition 7.1.

The outer-loop controllers depend only on the constraint and will therefore be the same for the point or parallelogram robots in the sections that follow. The transversal outer-loop controller is given by the PD controller in (7.14). The tangential outer-loop controller is based on the inverse of the tangential coordinate function

$$\psi_{line}(\hat{t}) = \varphi_{line}^{-1}(\hat{t}) = [\hat{t}, 0]^\intercal \,, \tag{7.24}$$

and the frictionless sliding dynamics (7.17). The outer-loop tangential controller (7.22) in this case simplifies to

$$v^{\parallel} = 0\,. \tag{7.25}$$

The outer-loop tangential controller (7.25) applies for both the theoretical analysis with the point robot and the experimental results on the parallelogram robot. The inner-loop feedback linearization controller will change based on the target robotic platforms considered next.

Theoretical Performance on the Line Constraint

The inner-loop set stabilization controller (7.11) for the point mass robot with dynamics (2.23), output (2.24) and virtual output (7.23) is

$$u = m_{point} \begin{bmatrix} v^{\perp} \\ v^{\parallel} \end{bmatrix}. \tag{7.26}$$

The controller (7.26) is well defined for $x \in \mathcal{E}_{line}^{point} \times \mathbb{R}^4$, where

$$\mathcal{E}_{line}^{point} = \mathcal{M}_{line} \cap \mathcal{D}^{point} = \mathbb{R}^2 \tag{7.27}$$

according to Proposition 7.1 since the constraint always has a full rank Jacobian and the point robot has a full vector relative degree everywhere (see Section 6.3.1).

The resulting closed-loop dynamics for the point mass robot with dynamics (2.23) and virtual output (7.23) under the inner-loop controller (7.26) with the outer-loop transversal and tangential outer-loop controllers (7.14) and (7.25) are

$$\begin{aligned} \ddot{y}_1 &= 0 \\ \ddot{y}_2 &= -K_p y_2 - K_d \dot{y}_2 \end{aligned}\,. \tag{7.28}$$

The dynamics in (7.28) are identical to those of the PP+D and PP+PD controllers previously seen in Subsection 6.3.2. When the constraint is linear, the set stabilization approach offers no performance advantage over traditional controllers. The performance of the SS controller is therefore equivalent to that of the PP+D, PP+PD controllers whose simulation results were shown in Figure 6.19.

Experimental Evaluation on the Line Constraint

The set stabilization controller is also equivalent to the PP+D and PP+PD controllers on the line constraint when a more complicated robot model is considered. The inner-loop set stabilization controller (7.11) for the parallelogram robot with dynamics (2.35) and output (2.36) with the virtual output (7.23) for the line constraint is

$$u = \begin{bmatrix} -\frac{\sin(x_2)d_2}{l_1\cos(x_1-x_2)} & -\frac{\cos(x_2)d_2}{l_1\cos(x_1-x_2)} \\ \frac{\cos(x_1)d_3}{l_4\cos(x_1-x_2)} & -\frac{\sin(x_1)d_3}{l_4\cos(x_1-x_2)} \end{bmatrix} \left(\begin{bmatrix} \cos(x_1)\,l_1 x_3^2 + l_4\sin(x_2)\,x_4^2 - \frac{\sin(x_1)l_1 b_2 x_3}{d_2} + \frac{l_4\cos(x_2)b_3 x_4}{d_3} \\ -\sin(x_1)\,l_1 x_3^2 + l_4\cos(x_2)\,x_4^2 - \frac{\cos(x_1)l_1 b_2 x_3}{d_2} - \frac{l_4\sin(x_2)b_3 x_4}{d_3} \end{bmatrix} + v \right) \tag{7.29}$$

The controller (7.29) is well defined for $x \in \mathcal{D}^{para2d} \times \mathbb{R}^2$

$$\begin{aligned} \mathcal{E}_{line}^{para2d} &= \mathcal{D}^{para2d} \cap \mathcal{M}_{line} \\ &= \mathcal{D}^{para2d} \end{aligned}$$

according to Proposition 7.1. The constraint always has a full rank Jacobian, so the domain of the controller is only limited by the robot's functional configuration space $\mathcal{D}^{para2d}$ defined in (2.39).

The set stabilization controller here is effectively the same as the standard feedback linearization controller in (6.17), with the only difference being that the outer-control loop has the control inputs in the opposite order. The result is that the rows of the pre-multiplying decoupling matrix in (7.29) and the columns of the matrix inside the parentheses are reversed relative to (6.17). Because the SS controller is identical to the PP+D and PP+PD controller, the experiments were not repeated for this case. The performance of the SS controller is essentially that of the experimental results on the right sides of Figures 6.20 and 6.21.

7.3.2 Set Stabilization on the Circle

The circle constraint (6.4) is now considered to determine the impact of constraint curvature on the controller's performance. For the circle case, the virtual output is defined as

$$\hat{y} = \sigma_{circ}(y) = \begin{bmatrix} s_{circ} \\ \varphi \circ \rho \end{bmatrix} = \begin{bmatrix} ||y||_2 - r_c \\ \arg(y_1 + iy_2) \end{bmatrix} . \tag{7.30}$$

The constraint is well defined where the Jacobian of the virtual output $\sigma_{circ}(y)$ with respect to the output y is full rank. For the circle constraint, σ_{circ}'s Jacobian is given by

$$\frac{\partial \sigma_{circ}(y)}{\partial y} = \begin{bmatrix} \frac{x_1}{\sqrt{y_1^2+y_2^2}} & \frac{y_2}{\sqrt{y_1^2+y_2^2}} \\ -\frac{y_2}{y_1^2+y_2^2} & \frac{y_1}{y_1^2+y_2^2} \end{bmatrix} . \tag{7.31}$$

The matrix (7.31) has a determinant of

$$\det\left(\frac{\partial \sigma_{circ}(y)}{\partial y}\right) = \frac{1}{\sqrt{y_1^2 + y_2^2}} , \tag{7.32}$$

so the Jacobian is full rank for $y \in \mathbb{R}^2/(0,0)$. The set $\mathcal{M}_{circle}$ is therefore

$$\mathcal{M}_{circle} = \left\{ x_c \in \mathbb{R}^2 : h(x_c, \cdot) \neq 0 \right\} . \tag{7.33}$$

The expression in (7.33) says that the virtual output is not well defined in the center of the circle. Neither the projection function nor the tangential coordinate are well defined in this case because it is not clear what surface point would be associate with the center of the circle, which forms the circle's medial axis.

The outer-loop transversal controller is again given by the PD controller in (7.14). The tangential controller is again based on ψ, the inverse of the tangential coordinate, which is given by

$$\psi_{circ}(\hat{t}) = \begin{bmatrix} r_c \cos \hat{t} \\ r_c \sin \hat{t} \end{bmatrix} . \tag{7.34}$$

The outer-loop tangential controller (7.22) is again

$$v^{\parallel} = 0\,, \tag{7.35}$$

for the frictionless surface dynamics (7.17).

The inner-loop controllers for the point and line robots are now derived for simulation and experimental evaluation with the point and parallelogram robots.

Theoretical Performance on the Circle Constraint

The inner-loop set stabilization controller (7.11) for the point mass robot with dynamics (2.23) and output (2.24) for the virtual output (7.30) for the circle constraint is

$$u = \begin{bmatrix} \frac{x_1 m_{point}}{\sqrt{x_1^2+x_2^2}} & -x_2 m_{point} \\ \frac{x_2 m_{point}}{\sqrt{x_1^2+x_2^2}} & x_1 m_{point} \end{bmatrix} \left(\begin{bmatrix} -\frac{(-x_2x_3+x_1x_4)^2}{\left(x_1^2+x_2^2\right)^{3/2}} \\ \frac{2\left(-x_1x_3^2x_2+x_3x_1^2x_4-x_3x_2^2x_4+x_1x_4^2x_2\right)}{\left(x_1^2+x_2^2\right)^2} \end{bmatrix} + v \right). \tag{7.36}$$

The domain of the controller is

$$\begin{aligned} \mathcal{E}^{point}_{circle} &= \mathcal{D}^{point} \cap \mathcal{M}_{circle} \\ &= \left\{ x_c \in \mathbb{R}^2 : h(x_c, \cdot) \neq 0 \right\} \end{aligned} \tag{7.37}$$

according to Proposition 7.1. With the controller now fully defined, its performance can be investigated.

The closed-loop system dynamics are formed by combining the inner-loop controller (7.36) with the outer-loop transversal and tangential controllers (7.14) and (7.35). Expressed with respect to the polar coordinates (6.23), the resulting system dynamics are

$$\begin{aligned} \ddot{r} &= -K_p(r - r_c) - K_d \dot{r} \\ \ddot{\theta} &= 0 \end{aligned}\,. \tag{7.38}$$

The tangential and transversal dimensions are decoupled without angular terms in the radial dynamics and vice versa. The radial dynamics will be stable for any choice of positive K_p and K_d, while the angular motion is only driven by the external disturbances. It is interesting to note that the mass of the point robot does not appear in (7.38) since the controller is automatically compensating for it.

The system's performance was simulated given the same initial condition as in Section 6.3.3 of $y(0) = (1,0)$ and $\dot{y}(0) = (0,5)$. This corresponds to a point on the set with velocity tangent to the circle. The radius of the constraint was taken as $r_c = 1$. Figure 7.3 shows the performance of the set stabilization controller in relation to the other regulation schemes. The set stabilization controller is the only one capable of maintaining zero radial error throughout the simulation, demonstrating the invariance of its output. As such, it is the only controller presented in this thesis capable of successfully addressing Problem 7.1.

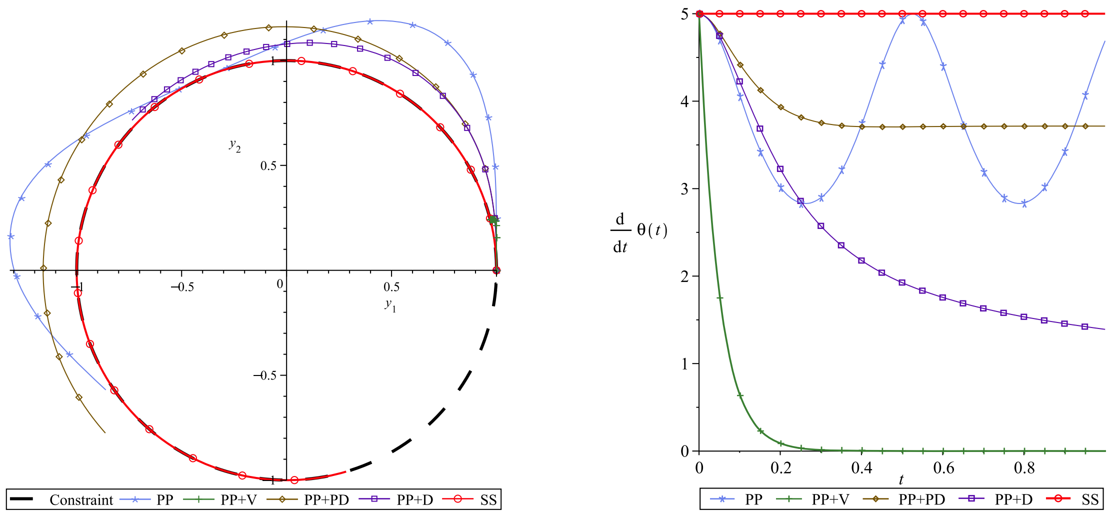


Figure 7.3: Simulation of the set stabilization dynamics (7.38) along with those from Table 6.2 for the initial condition $y(0) = (1,0)$ and $\dot{y}(0) = (5,0)$

The projective controllers all exert zero force at $t = 0$ in the simulations of Figure 7.3 so that $u(0) = 0$. In contrast, the set stabilization controller exerts the force

$$u(0) = \begin{bmatrix} -25m_{point} \\ 0 \end{bmatrix} \tag{7.39}$$

towards the center of the circle. This corresponds to the centripetal acceleration $\frac{m_{point}\dot{y}^2}{r_c}$ necessary to maintain the output on the constraint, emulating the Centripetal Force Phenomenon.

Where the projective controllers require an accumulated transversal error to induce a centripetal acceleration, the set stabilization controller explicitly accounts for it.

Experimental Performance on the Circle Constraint & Sources of Error

The previous set stabilization controllers resulted in expressions for the inner-loop controller compact enough to include in this thesis. The set stabilization controller (7.11) for the parallelogram robot with dynamics (2.35), output (2.36) and virtual output (7.30), however, yields a very lengthy expression that does not lend itself well to interpretation. And this is in spite of the relatively simple robot dynamics and output function, with the parallelogram robot specifically designed to possess simple dynamics for use in advanced robotics control research [65]. This case demonstrates how quickly the size of the algebraic expression for the controller grow when performing feedback linearization with respect to virtual outputs.

While the expressions may become too unwieldy to calculate by hand for most practical combinations of robots and constraints, the use of symbolic computation tools ensures that set stabilization is still a practical control option. The inner-loop controller was generated symbolically using Maple, whose results were then be exported to MATLAB code for inclusion in a Simulink based control environment. The resulting code for the inner-loop controller was approximately one thousand lines of MATLAB code, with each line usually consisting of a single algebraic operation due to the nature of Maple's code optimization process. Despite the size of the expressions, the Maple worksheet requires just a few seconds to calculate the controller for a given robot model and constraint. The controller is well defined for the domain

$$
\begin{aligned}
\mathcal{E}^{para2d}_{circle} &= \mathcal{D}^{para2d} \cap \mathcal{M}_{circle} \\
&= \left\{ x \in \mathcal{D}^{para2d} \times \mathbb{R}^2 : h(x) \neq 0 \right\}
\end{aligned}
\tag{7.40}
$$

according to Proposition 7.1.

The same experiment as in Section 6.3.3 was run, manually providing a tangential disturbance to the robot whose output is initially on the constraint. The results of the transversal and tangential performance are seen in Figure 7.4. The transversal oscillations of the set stabilization controller exhibit both a lower frequency and lower amplitude over the projective controllers. In addition to superior tracking results in the transversal dimension, the tangential motion is closer to the desired frictionless sliding case. Ideally, the tangential velocity would have the form of a step function that rises when the impulsive external disturbance is applied and then remains constant. The tangential velocity is steadier with the set stabilization controller, showing gradual fluctuations that are most likely due to unmodelled configuration dependent friction effects. The theoretical advantages of the set stabilization method therefore translate into better performance under real world conditions.

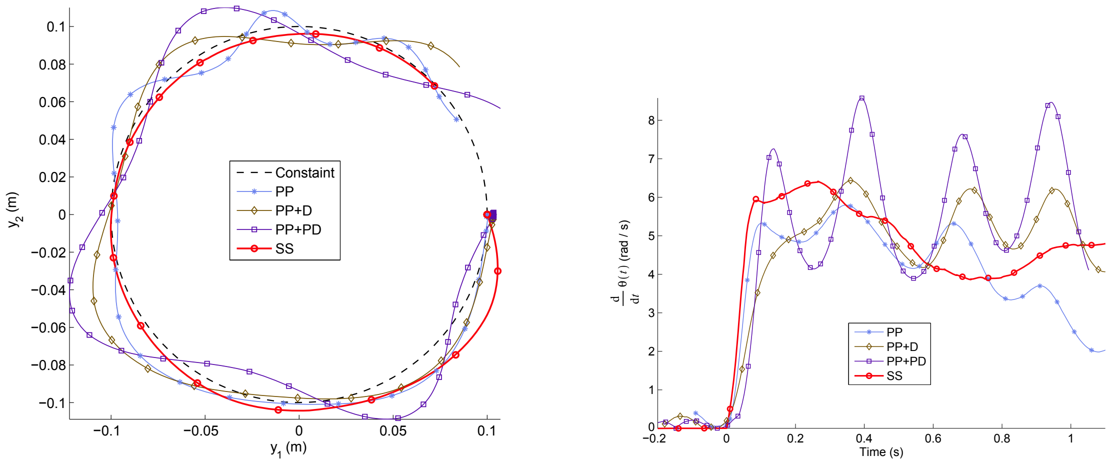


Figure 7.4: Experimental positions (left) and tangential velocities (right) of the set stabilization controller on the circle, along with the projective controller results from Section 6.3.3

The tracking error in Figure 7.4 can be attributed to a variety of sources. The majority of these sources impact both the projective haptic controllers and the set stabilization controller equally, which include errors in the model form, the model's parameter estimates, the accuracy of the position encoders, the quality and delay of the velocity estimations, and the delay and quantization introduced by the sampled-data nature of the control system. The haptic projective controllers are also subject to errors based on the lack of asymptotic tracking for curved constraints demonstrated in Subsection 6.3.3. While the operator's force does not enter into these tests, it is modelled as an external disturbance and when present would contribute a significant source of tracking error. The future research in Section 8.2 discusses how to incorporate the operator's force to improve the performance of the haptic simulator.

The model likely includes unmodelled configuration dependent friction. The parallelogram's links are not exactly parallel so that the joints experience either a compressive or tensile force in the axial direction, depending on the robot's configuration. The change in loading on the joints impacts their friction properties and hence the degree to which the friction is being compensated through the feedback linearization controller. The general trend of the fluctuations in both the transversal and tangential components in Figure 7.4 aligns across subsequent rotations, further indicating that the error is configuration dependent.

7.4 Application of Set Stabilization to a Surface in 3D

The model's surface in a two dimensional space has thus far been a one dimensional constraint. In practical applications, the model's surface forms a closed two dimensional constraint in a three dimensional output space. The extension to a closed set in a higher dimensional space is not as straightforward as redefining the size of the vectors and matrices from Section 7.3. Where a closed curve in 2D can be covered by a single coordinate chart, the same cannot be said of closed surfaces in 3D [134]. This difficulty can be overcome by switching between a collection of coordinate systems whose domains cover different pieces of the constraint. The use of a switching controller allows for the generalization of the control design procedure from the last section to systems with higher dimensional constraints.

The set $\mathcal{S}$ is now assumed to consist of a closed surface, i.e. a compact, oriented, boundaryless two dimensional manifold [133]. The points in $\mathcal{S}$ are still the zero level set of the function s, but the domain of s is now in $\mathbb{R}^3$ instead of $\mathbb{R}^2$. As in the 2D case, a single function s suffices to provide the transversal coordinate as a "distance" to the set. It is only the tangential coordinate functions that cannot be defined over the entirety of $\mathcal{S}$ with a single coordinate chart.

From a differential geometry point of view, the tangential coordinates form coordinate charts on the surface $\mathcal{S}$. For a basic introduction on differential geometry see [81] or for a more advanced presentation see [134].

Definition 7.1. *A coordinate chart of a set $\mathcal{S}$ is a pair (W, φ), where W is a subset of $\mathcal{S}$ and $\varphi : W \to U$ is a bijection from W onto the domain $U \subset \mathbb{R}^{N-1}$.*

Since the coordinate function φ is not defined over the entire set $\mathcal{S}$, a collection of coordinate charts (W_i, φ_i) can be combined to form an atlas $\mathcal{A}$, which is a set of compatible coordinate charts that cover the entire set.

Definition 7.2. *Two coordinate charts (W_1, φ_1), (W_2, φ_2) are C^k-compatible if*

(i) $\varphi_1(W_1 \cap W_2)$, $\varphi_2(W_1 \cap W_2)$ are open sets in $\mathbb{R}^{N-1}$ and

(ii) when $W_1 \cap W_2 \neq \varnothing$, the coordinate changes

$$\varphi_1 \circ \varphi_2^{-1} : \varphi_2(W_1 \cap W_2) \to \varphi_1(W_1 \cap W_2), \tag{7.41}$$

$$\varphi_2 \circ \varphi_1^{-1} : \varphi_1(W_1 \cap W_2) \to \varphi_2(W_1 \cap W_2) \tag{7.42}$$

are both of differentiability class C^k.

The compatibility of the charts from Definition 7.2 means that there is an overlap between the charts on $\mathcal{S}$ such that one set of tangential coordinates can be translated to the other. This property is essential in designing a controller that can switch smoothly between different virtual outputs based on these tangential coordinates.

Feedback linearization controllers are designed for each chart in the atlas $\mathcal{A}$ that covers $\mathcal{S}$. For each chart, a separate virtual output is defined as

$$\hat{y}_{(i)} = \begin{bmatrix} \hat{n} \\ \hat{t}_{(i)} \end{bmatrix} = \sigma_{(i)}(y) = \begin{bmatrix} s(y) \\ \varphi_{(i)} \circ \rho(y) \end{bmatrix} \tag{7.43}$$

$$= \alpha_{(i)}(x) = \begin{bmatrix} s \circ h(x) \\ \varphi_{(i)} \circ \rho \circ h(x) \end{bmatrix} = \begin{bmatrix} \lambda(x) \\ \pi_{(i)}(x) \end{bmatrix}, \tag{7.44}$$

where the subscript (i) indicates the coordinate chart on which the output is defined. Subscripts without parentheses indicate an element within a vector / matrix as before.

Each virtual output yields a separate feedback linearization controller

$$u_{(j)} = \left(L_g L_f \hat{y}_{(i)}\right)^{-1} \left(-L_f^2 \hat{y}_{(j)} + v_{(j)}\right),\ j \in \{1, \ldots, n_c\}, \tag{7.45}$$

where n_c is the number of coordinate charts in the atlas $\mathcal{A}$. The overall switching controller is given by

$$u(x) = u_{(i)}(x),\ i \in \{1, \ldots, n_c\}, \tag{7.46}$$

where $i : x \mapsto \{1, \ldots, n_c\}$ is the switching law. If each virtual output $\hat{y}_{(i)}$ yields a full vector relative degree on the domain of the associated coordinate chart W_i, then an appropriate switching law i can be designed to ensure that the overall switching controller is well defined [10]. The details of the proof are in [10], but intuitively it is reasonable that a smooth controller would result when switching between the various $u_{(i)}$'s. The transversal component is the same across the controllers and will thus be smooth. The tangential components are designed to achieve the desired closed-loop dynamics $\ddot{y} = a(y, \dot{y})$ on the surface $\mathcal{S}$ where the controller is simply expressing these dynamics with respect to different coordinate systems to make sure that the problem is well defined. The desired output dynamics can readily be re-expressed across coordinate charts in the overlap of their domains thanks to their compatibility.

The purpose of switching is to avoid the representational singularities beyond the boundaries of the tangential coordinates' domain. As such, there are many possible switching laws that would successfully transition between charts so long as the switching takes place somewhere in the overlap between charts. The only requirement on the switching law is that for any given state configuration, the switching law $i(x)$ satisfies

$$\rho(h(x)) \in W_{(i)}, \tag{7.47}$$

i.e. that the projected output for the current chart lies within the domain of $\varphi_{(i)}$. The current chart can be selected to ensure that the output is as far from the boundary around $W_{(i)}$ as possible according to

$$i(x) = \sup_{i \in \{1,..,n_c\}} \left\{ \inf_{\omega \in \Omega_i} \|\psi_i(h(x)) - \omega\| \right\} \tag{7.48}$$

where $\Omega_i = \mathcal{S} \backslash W_i$. The controllers are less well conditioned numerically speaking near the edge of $\varphi_{(i)}$'s domain, so the switching law (7.48) helps ensure that the controller is well behaved. In practice, (7.48) may be time consuming to calculate and so taking advantage of the structure of the charts for particular applications is advantageous since it yields much simpler expressions for $i(x)$. This is the case here for the problem on the sphere, as will be seen in the construction of (7.56).

7.4.1 Control Design for a Set Stabilization Controller on the Sphere

The switching set stabilization approach is now applied to a spherical constraint to demonstrate the control design procedure. The process consists of defining the constraint along with the appropriate transversal and tangential coordinates. An atlas $\mathcal{A}$ is formed of two sets of tangential coordinate functions, each with their own associated virtual output and feedback linearization controller. A switching law is designed to determine when each controller is applied.

The constraint is the sphere centered at the origin of the output space with radius r_c given implicitly by

$$y_1^2 + y_2^2 + y_3^2 = r_c^2 \,. \tag{7.49}$$

The constraint to be stabilized is therefore the set

$$\mathcal{S} = \left\{ y \in \mathbb{R}^3 : y_1^2 + y_2^2 + y_3^2 = r_c^2 \right\} . \tag{7.50}$$

The signed distance to the sphere provides a natural transversal coordinate

$$s_{sphere}(y) = \sqrt{y_1^2 + y_2^2 + y_3^2} - r_c \,. \tag{7.51}$$

One choice for $\mathcal{A}$ consists of two coordinate charts based on the angular components of a spherical coordinate system. The coordinate charts formed from spherical coordinates are not defined at the north and south poles. To ensure that $\mathcal{S}$ is covered by the charts, two versions of the spherical coordinates rotated 90° relative to each other are used. The atlas is defined as $\mathcal{A} = \{(W_1, \varphi_1), (W_2, \varphi_2)\}$ where the domain of the two charts are

$$W_1 = \mathcal{S} \backslash \{(0, 0, \pm r_c)\}, \tag{7.52}$$

$$W_2 = \mathcal{S} \backslash \{(\pm r_c, 0, 0)\} \tag{7.53}$$

and the tangential coordinate functions $\varphi_{(1)} : W_1 \to (-\pi,\pi)\times(0,\pi)$, $\varphi_{(2)} : W_2 \to (-\pi,\pi)\times(0,\pi)$ are defined as

$$\varphi_{(1)}(y) = \begin{bmatrix} \varphi_{(1),1}(y) \\ \varphi_{(1),2}(y) \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{y_2}{y_1}\right) \\ \arccos\left(\frac{y_3}{\sqrt{y_1^2+y_2^2+y_3^2}}\right) \end{bmatrix}, \tag{7.54}$$

$$\varphi_{(2)}(y) = \begin{bmatrix} \varphi_{(2),1}(y) \\ \varphi_{(2),2}(y) \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{y_3}{y_2}\right) \\ \arccos\left(\frac{y_1}{\sqrt{y_1^2+y_2^2+y_3^2}}\right) \end{bmatrix}. \tag{7.55}$$

The switching law is designed so that the controller (7.45) uses each coordinate chart only on its domain

$$i(x) = \begin{cases} 1\,, & \frac{\pi}{4} \le \varphi_{(1),2} \circ \rho \circ h(x) \le \frac{3\pi}{4} \\ 2\,, & \text{otherwise} \end{cases}. \tag{7.56}$$

The region where each chart is active on $\mathcal{S}$ is shown in Figure 7.5. The switching occurs far from the edge of the domain of either chart, which helps ensure that the controller is numerically well conditioned in practice.

The virtual output is formed by the tangential coordinates composed with a projection function. For the sphere, it is natural to project radially to its surface

$$\rho_{sphere}(y) = r_c \frac{y}{\sqrt{y_1^2+y_2^2+y_3^2}}, \tag{7.57}$$

which is well defined for all points except the center of the sphere. The virtual outputs are formed by composing (7.55) with the projection (7.57) as

$$\sigma_{(1)}(y) = \begin{bmatrix} s(y) \\ \varphi_{(1)} \circ \rho(y) \end{bmatrix}, \tag{7.58}$$

$$\sigma_{(2)}(y) = \begin{bmatrix} s(y) \\ \varphi_{(2)} \circ \rho(y) \end{bmatrix}. \tag{7.59}$$

According to Proposition 7.1, these virtual outputs will have a full vector relative degree on the points $x \in \mathcal{D}_0^{para3d} \times \mathbb{R}^3$ from (2.33) where the Jacobians of the virtual outputs with respect

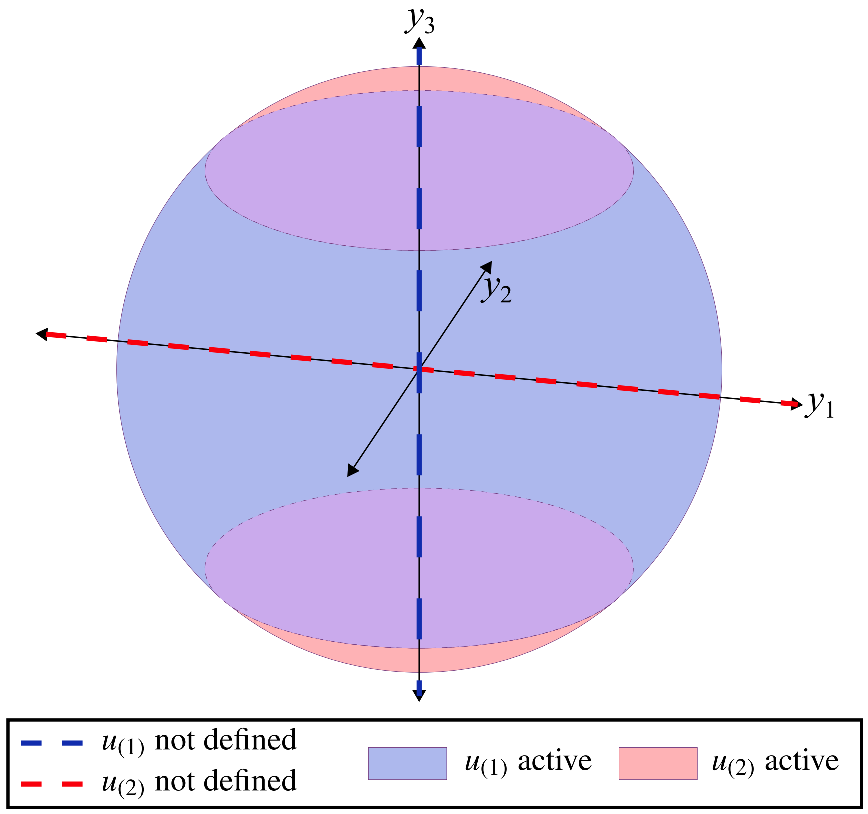


Figure 7.5: Regions of the sphere's surface $\mathcal{S}$ where each coordinate chart is used according to the switching law (7.56)

to y have full rank. In other words, the set $\mathcal{M}_{sphere}$ must be determined so that the controller's domain $\mathcal{E}^{para3d}_{sphere}$ can be formed as the intersection of $\mathcal{M}_{sphere}$ and $\mathcal{D}^{para3d}_0$. The Jacobian for the first virtual output (7.58) is

$$\frac{\partial \sigma_{(1)}}{\partial y} = \begin{bmatrix} \frac{y_1}{\sqrt{y_1^2+y_2^2+y_3^2}} & \frac{y_2}{\sqrt{y_1^2+y_2^2+y_3^2}} & \frac{y_3}{\sqrt{y_1^2+y_2^2+y_3^2}} \\ -\frac{y_2}{y_1^2+y_2^2} & \frac{y_1}{y_1^2+y_2^2} & 0 \\ \frac{y_3 y_1}{\left(y_1^2+y_2^2+y_3^2\right)\sqrt{y_1^2+y_2^2}} & \frac{y_3 y_2}{\left(y_1^2+y_2^2+y_3^2\right)\sqrt{y_1^2+y_2^2}} & -\frac{\sqrt{y_1^2+y_2^2}}{y_1^2+y_2^2+y_3^2} \end{bmatrix}. \tag{7.60}$$

The determinant of (7.60) is

$$\det\left(\frac{\partial \sigma_{(1)}}{\partial y}\right) = -\frac{1}{\sqrt{y_1^2+y_2^2}\sqrt{y_1^2+y_2^2+y_3^2}}, \tag{7.61}$$

which is non-zero for $y \in U_1$ where

$$U_1 = \mathbb{R}^3 \backslash \text{span} \left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}. \tag{7.62}$$

This corresponds to all points in the output space except those that lie on the y_3 axis. A similar process shows that the virtual output $\hat{y}_{(2)}$ yields a full rank Jacobian on the domain

$$U_2 = \mathbb{R}^3 \backslash \text{span} \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\}, \tag{7.63}$$

which includes all points except those that lie on the y_1 axis.

The switching law (7.56) ensures that appropriate virtual output is used so that $U = U_1 \cup U_2$ for the switching controller. The set U expressed in the joint space as $\mathcal{M}_{sphere} \subset \mathbb{R}^3$ is thus

$$\mathcal{M}_{sphere} = \{x_c \in \mathbb{R}^3 : h(x_c, \cdot) \in U\}. \tag{7.64}$$

The controller is therefore defined for $x \in \mathcal{E}_{sphere}^{para3d} \times \mathbb{R}^3$ where

$$\mathcal{E}_{sphere}^{para3d} = \mathcal{M}_{sphere} \cap \mathcal{D}_0^{para3d} \tag{7.65}$$

If the size of the sphere is small enough to fit within the functional workspace as in Assumption 7.1, then (7.65) is simply $\mathcal{M}_{sphere}$. The only point where the controller is undefined is then the center of the sphere.

The inner-loop controller is given by (7.46) for the robot dynamics in (2.26) with output (2.28) and virtual outputs (7.58) and (7.59). As in the 2D case on the circle, the expression for the inner-loop controller is too large to include here. The outer-loop transversal controller is again the PD controller (7.14). The outer-loop transversal controller is based on the inverse of the tangential coordinate functions (7.55)

$$\psi_{(1)}(\hat{t}) = \begin{bmatrix} r_c \cos(\hat{t}_1) \sin(\hat{t}_2) \\ r_c \sin(\hat{t}_1) \sin(\hat{t}_2) \\ r_c \cos(\hat{t}_2) \end{bmatrix}, \quad \psi_{(2)}(\hat{t}) = \begin{bmatrix} r_c \cos(\hat{t}_2) \\ r_c \sin(\hat{t}_1) \sin(\hat{t}_2) \\ r_c \cos(\hat{t}_1) \sin(\hat{t}_2) \end{bmatrix}. \tag{7.66}$$

The outer-loop tangential controller is then formed by using (7.66) in the general tangential controller (7.22). The desired tangential dynamics are given by (7.18), which includes viscous friction. The presence of friction here demonstrates how non-trivial friction dynamics affect the resulting tangential controller. It so happens that the tangential controllers have the same form on both charts. With respect to the linearized state variable η, the controller is

$$v_{(i)}^{\|} = \begin{bmatrix} -\frac{\left(b_d \sin\left(\eta_{(i),3}\right)+2\cos\left(\eta_{(i),3}\right)\eta_{(i),4} m_{tool}\right)\eta_{(i),2}}{m_{tool}\sin\left(\eta_{(i),3}\right)} \\ \frac{\cos\left(\eta_{(i),3}\right) m_{tool}\sin\left(\eta_{(i),3}\right)\eta_{(i),2}^2 - b_d \eta_{(i),4}}{m_{tool}} \end{bmatrix} \tag{7.67}$$

7.4.2 Evaluation of the Set Stabilization Controller on the Sphere

An experiment was constructed to evaluate the controller designed in the previous section. The robot was initialized away from the surface $\mathcal{S}$. Once the controller had pulled the output onto the constraint, the end-effector was grasped and moved along the great circle that contains the prime meridian of the first coordinate chart, which separates its east and west hemispheres. This path ensures that the controller would switch several times between $u_{(1)}$ and $u_{(2)}$. The sphere's radius was chosen as $r_c = 0.1\,\mathrm{m}$ so as to lie entirely within the robot's functional workspace. The desired dynamics here introduced an element of damping as in (7.18) where $m_{tool} = 0.1\,\mathrm{kg}$ and $b_d = 0.3\,\mathrm{kg/s}$. The control gains were chosen as $K_p = 200$ and $K_d = 28.3$ to achieve critical damping in the transversal dimension.

The ensuing transversal position and velocity are shown in Figure 7.6. The manipulator quickly snaps to the set stabilized by the transversal controller. The external disturbance starts around $t \approx 2.5\,\mathrm{s}$ as the end-effector is grasped, which causes the fluctuations in the transversal subsystem (i.e. ξ_1 and ξ_2) about the origin that follow from that point onwards. The peak of the transversal error appears at around $t \approx 4.5\,\mathrm{s}$ when traversing over the top of the sphere. The sphere is an invisible constraint in space and so the change in direction required to traverse over the top of the pole is difficult to anticipate, leading to a larger force from the operator which exacerbates the tracking error.

Figure 7.7 shows the tangential coordinates in each chart, with the shaded regions indicating where a particular chart is currently inactive. During the motion around the sphere, four switches

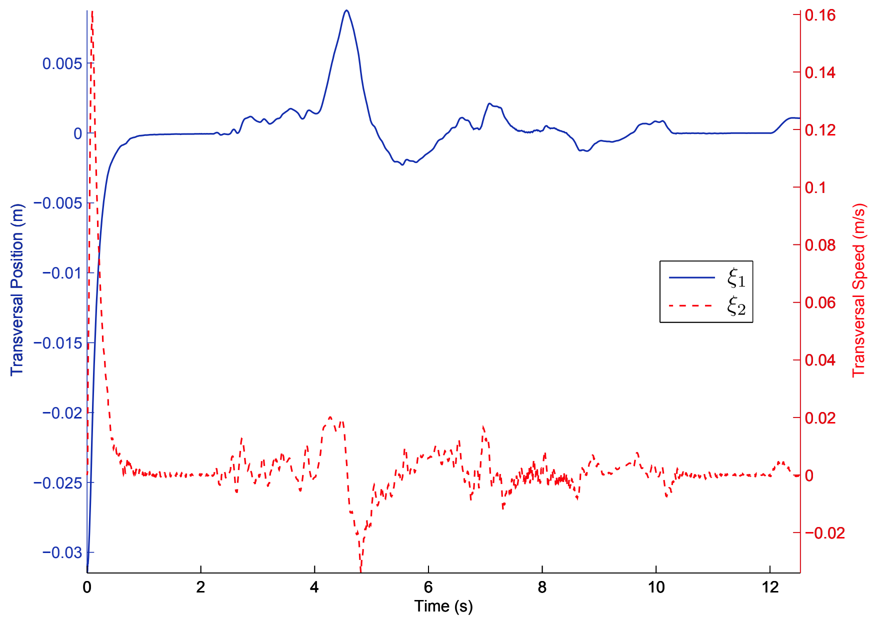


Figure 7.6: Transversal coordinates while the manipulator is driven around the sphere by an external disturbance.

between coordinate charts can be observed at $t \approx 3$, 5, 7 and 9 seconds. Several abrupt changes in the longitudinal coordinates of each chart, $\eta_{(1),1}$ and $\eta_{(2),1}$, can be seen in the shaded areas where those charts are not active, highlighting the necessity of the switching control law.

The controller u can be seen to remain smooth during switching. Figure 7.8 shows the control effort for the 2$^{\text{nd}}$ joint as an example. The vertical dashed black lines show where the transition between coordinate charts take place, with the active chart indicated at the top of the figure. The two control values are identical at most points in time, diverging when the robot's output approaches the border of the inactive chart's domain. Four such divergent areas can be seen in Figure 7.8 at $t \approx 0.5$, 4, 7.5 and 9.5 seconds. In each case, the control signal of the inactive chart spikes in the numerically ill-conditioned region near the boundary of its domain while the active controller remains well behaved. The signals converge again before the next coordinate chart switch as the switching points were chosen far from the singularities of both tangential coordinate functions.

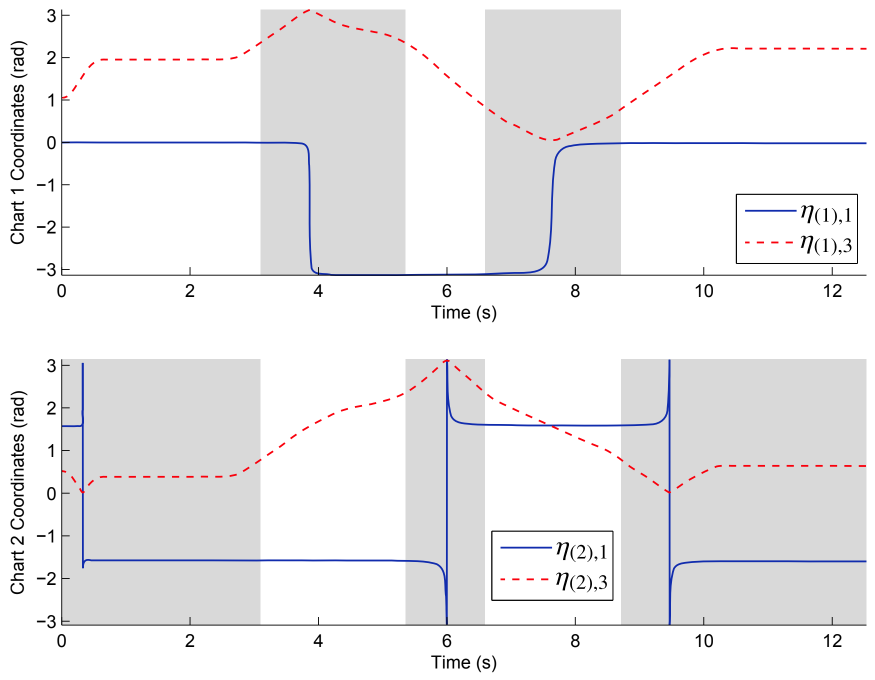


Figure 7.7: Tangential coordinates while the manipulator is driven around the sphere by an external disturbance

7.5 Summary

Where the set stabilization controller is equivalent to the other regulation schemes on flat constraints, it is the only controller in this thesis capable of asymptotically stabilizing the circle constraint and decoupling the transversal from the tangential dynamics. Only the set stabilization controllers contain terms to induce the centripetal accelerations required to follow a curved surface. The projective controllers of Chapter 4 instead rely on the position error in order to loosely follow the curve of the constraints. Not only will the set stabilization technique provide better tracking than tradition haptic renderers, but the addition of centripetal effects provide a better simulation of the Centripetal Force Phenomenon, improving the perception of the surface's curvature.

For models in three dimensions, the controller was extended to use a series of coordinate charts forming an atlas over the model's surface. Switching between a series of linearizing controller based on these charts produced a smooth, well defined controller capable of achieving

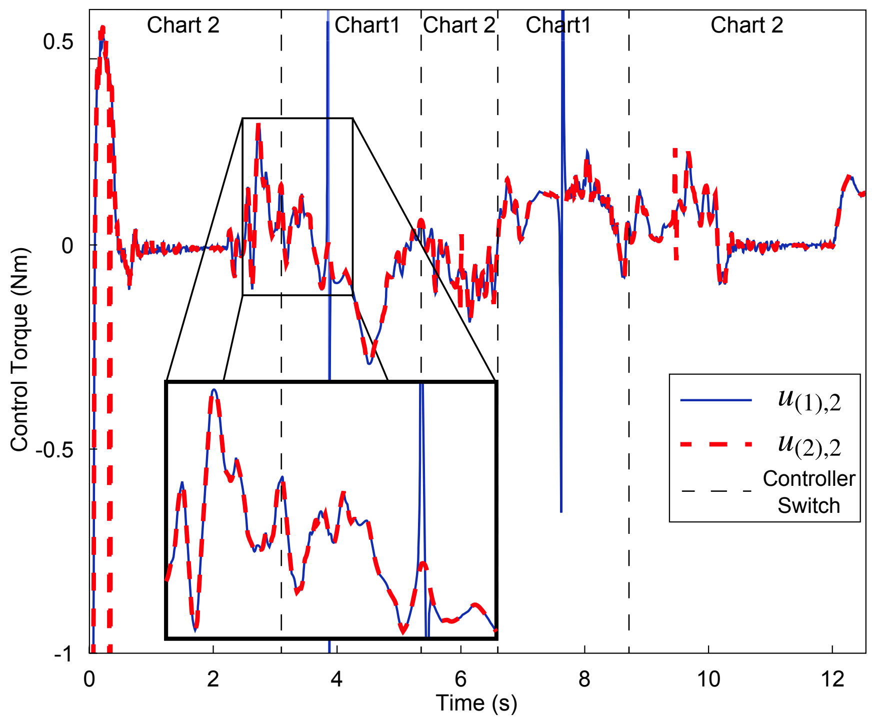


Figure 7.8: Control torques on the 2nd joint for the motion around the sphere. The enlarged region shows the convergence of $u_{(1),2}$ and $u_{(2),2}$ at the switching point.

the same level of performance in three dimensions as was shown in the two dimensional case.

Chapter 8

Conclusions & Future Work

This work has advanced the current understanding of the haptic rendering of rigid objects and proposed ways to improve the performance of haptic simulators. This chapter summarises the conclusions that can be drawn from the present work and outlines further research directions for the haptics community.

8.1 Conclusions

The performance of haptic renderers in rendering rigid objects was analysed, especially in light of the limits on renderable stiffnesses that may permit significant penetrations of the operator into the virtual models. The haptics literature seems to indicate that the pop-through problem has been solved, given the general consensus in the adoption of the constraint renderer as explained in Chapter 2, with later works treating the stability of the rendering itself. The haptic renderer was separated into a projection function and a regulation scheme, with each component analysed separately to determine their limitations.

The projection functions of the minimum distance and constraint renderers were shown to contain discontinuities, which gives rise to the pop-through haptic artifact when encountered by the operator. The exact causes of pop-through were investigated in Section 4.1, showing for the first time how it could occur on curved models under the constraint method. Since the pop-through occurs for both exact and discretized model representations, this represents an issue intrinsic to the constraint renderer's design philosophy. The various regulation schemes were also enumerated in Section 4.2. Regulation is often viewed as a follow-up step to the projection, using the projection point as a desired point and driving the output to that point based on a local linear contact model.

A new haptic rendering method was proposed in Chapter 5 to reduce the singularities in the projection function to the greatest degree possible. The goal was to ensure the continuous rendering of smooth shapes, especially under large penetrations. The projection function was formed via the intermediate level of a curve shortening flow, flowing the model's surface down to a point in its interior. The use of a projection function with curved flow lines necessitated a modification to the regulation scheme. The response force has to be projected onto the surface's normal because the position error vector would skew the perception of the rendered surface.

An evaluation methodology was developed to qualitatively assess the renderers based on comparing the physical phenomena that they exhibit to those that should be exhibited, listed in Chapter 3. This methodology provides a faster, less expensive preliminary evaluation of haptic renderers before embarking on human factors studies. This strategy also helps to generate results that are more easily compared among haptics researchers, since the experimental results from human factors studies often apply to the particular combination of haptic device and control methodology.

A series of tests on smooth and angular models were conducted in Chapter 6 to show how the existing renderers behaved under various test scenarios. While differences were observed between the methods, these differences diminish as the renderable stiffness is increased so that the methods all become equivalent in the limit. It was shown that neither the minimum distance, constraint, nor mapping renderers are capable of generating the appropriate qualitative behaviour in all test scenarios, highlighting the utility of using the identified phenomena in Section 3.2 as a meaningful set of benchmarks. The mapping renderer was shown to perform poorly on convex corners and on curved objects when the HIP crosses the model's medial axis. The pop-through for the constraint renderer, associated with the singularities in the projection function identified earlier, was shown to introduce corners into smooth models when the penetration was sufficiently large. The constraint renderer does perform quite well on angular models, the case for which the method was originally conceived.

Dynamic analyses were also used to evaluate the renderers by considering their performance as bilateral virtual constraints. If they perform poorly as constraints, it can be expected that the parallel form of unilateral haptic renderer would perform poorly as well. The virtual constraints have the advantage of being modelled by continuous differential equations, unlike the haptic systems which experience an abrupt change in behaviour as contact is made or broken. This continuity lends itself well to analysis using the well developed nonlinear control theory. Recasting the common haptic renderers as virtual constraints, Section 6.3 shows that they can only achieve good performance on linear constraints. For curved constraints, they are incapable of making the constraint asymptotically stable and they couple the transversal and tangential control dimensions. The coupling between control dimensions is particularly worrisome since it impacts the tangential dynamics, introducing extraneous surface contact forces.

Set stabilization was then explored in Chapter 7 to address the shortcomings of the current haptic regulation schemes. Set stabilization provides a framework to formulate the goals of the haptic simulator in terms of control objectives and to which nonlinear control techniques can be applied. It was shown that the transversal and tangential dimensions of the problem could be separated to ensure asymptotic tracking of the constraint and dynamic assignment along the constraint. The set stabilization controller, through the use of a virtual output based on the constraint itself, incorporated full knowledge of the haptic device's dynamics and kinematics as well as the constraint's geometry into the controller to achieve output invariance of the constraint. The equivalence between set stabilization and projective controllers was shown in Section 7.3 for linear constraints, as well as their differences on curved constraints. Only the set stabilization controller could maintain the output's position on the circle constraint without impacting the tangential velocity in simulation. The experimental results confirmed that the set stabilization controller was best able to enforce the constraint while having the least impact on the tangential velocity when simulating frictionless sliding.

The extension of set stabilization to closed surfaces in three dimensions was then covered. The use of multiple sets of tangential coordinate functions is required to be able to describe the position of the IHIP across the entire the model's surface. Separate feedback linearization controllers were designed for each of these coordinate charts and a switching law was introduced to transition between them. The result is a smooth control signal that is well defined over all but a small set of singular points in the output space. This extension to 3D allows set stabilization to be applied to practical haptics problems.

8.2 Future Work

The work of this thesis can be continued by improving on the transformation function for the mapping renderer, deriving a numerical implementation of the set stabilization method for arbitrary constraint shapes, compensating for the anisotropic device inertia and improving stability by performing the control design on a sampled data system.

8.2.1 Transformation Function Improvements

The projection function based on curve shortening flows in Chapter 5 were limited to models of genus 0 (i.e. without holes) that have a relatively well proportioned shape. When flowing the model down to a point, the dumbbell underwent a topological change splitting off into two pieces because the velocity function of the flow operated only on local geometric information. To take into consideration that two distant model surfaces may collide when flowing down, global geometric information must be incorporated into the gridding technique. Two possible techniques to overcome this limitation would be to incorporate global information into the flow's velocity function or to consider an alternate method to calculate the transformation between domains.

A natural alternative to curve shortening flows would appear to be an application of nonlinear elasticity to deform the circle into the shape of the model. The problem would consist of a boundary value problem, with the surface of the circle and model being specified as the start and end conditions for the deformation. A constituent law governing the behaviour of the material would then dictate the deformation field on the interior of the model. Ideally, a method would be used that does not require an explicit correspondence between boundary points on the model and the circle as this imposes a parameterization on the surface that may impact the resulting deformation field. The constituent law should resist twisting of the material to ensure that the field will be normal to the surface for a differential penetration.

Models with different genii also need to be considered to extend the class of applicability of the method. This would involve mapping models to canonical shapes with identical genii. The projection functions of models with holes would have more than a single singular point, as was the case for genus 0 models, because their canonical forms are of a more complicated nature. For example, models with one hole would be mapped onto a torus, which has a ring of singular points. The mapping procedure would still minimize the occurrence of singular points in the projection, essentially pruning away the branches of the object's medial axis.

8.2.2 Set Stabilization Extensions

The models considered when evaluating the set stabilization controllers had simple analytical forms that could be incorporated directly into expressions for the controller or for the closed-loop dynamics. Models in general cannot be represented so compactly, so numerical methods must be investigated for the implementation of set stabilization controllers. This would involve separating the controller into vectors and matrices like the inverse tangential coordinate ψ and the Jacobian $\frac{\partial s}{\partial y}$ that would be calculated and multiplied online.

The curve shortening flows from Chapter 5 would provide a means to define the distance to the model s and coordinate along the model φ, combining what have so far been separate areas of research. Just as the mapping renderer's transformation function was developed to minimize the singularities of the projection function in the model space, so would it serve to maximize the domain $\mathcal{E}$ on which the feedback linearization controllers are well defined.

The form of controllers set stabilization controllers used in the experimental results implemented bilateral virtual constraints. The conversion to unilateral constraints requires the consideration of the effect of switching between different stabilizing controllers. The result of switching between stabilizing controllers does not guarantee stability of the overall system [123].

The versatility of the set stabilization framework is well suited to combine the orientation of the end-effector into the control problem. The robot dynamics can be augmented with additional control inputs and states, and the output and virtual output can be augmented with the orientation's representation to produce an all encompassing control strategy. The system would then be able to model more complicated contact scenarios, expanding the capabilities of the haptic simulator.

8.2.3 Device Inertia as Experienced by the Operator

While the set stabilization controllers as presented in Chapter 7 can successfully decouple the transversal and tangential control dimensions, the operator's force has been considered as an external disturbance. This limits the haptic simulator's performance since the operator's force will induce tracking errors in the transversal dimension. The inclusion of force information has been shown to improve the performance of teleoperation in a direct comparison of control topologies, with similar benefits being expected for haptic applications [135].

In addition to the tracking errors, the inertia of the robotic device remains uncompensated, which leads to the virtual tool feeling heavier in some directions than in others. The parallelogram robot, for example, has rotational inertias that differ by more than an order of magnitude, as seen in the d_i values of Table A.1, making side to side movements that rotate the base joint more difficult than up/down motions that move the lighter parallelogram structure. The anisotropic inertia of a haptic device can be minimized during its mechanical design [136], but most commercially available haptic devices possess non-negligible anisotropies in their inertias as seen at their end-effectors.

The device's dynamic model can be modified to include the operator's force F_o as an exogenous input. With F_o appearing in the model, the effective inertia of the virtual tool can be determined by examining the relationship between it and the device's acceleration $\ddot{y}$. The input u in the state space model (2.14) can be replaced with

$$u = u_a + J^\intercal F_o \tag{8.1}$$

to include the operator's force as reflected back onto the joint torques. The state space dynamics then become

$$\dot{x} = f(x) + g(x)u_a + g_o(x)F_o\,, \tag{8.2}$$

where $g_o(x) = g(x)J^\intercal$. The acceleration for the system with dynamics is (8.2) is

$$\ddot{y} = L_f^2 y + L_g L_f y u_a + L_{g_o} L_f y F_o \,. \tag{8.3}$$

Since u is a state feedback controller, it will only contain terms in y and $\dot{y}$ when expressed with respect to the output space. The controller term and the $L_f^2 y$ term can be grouped into a function $b(y, \dot{y})$ to organize the dynamics in (8.3) according to

$$\left(L_{g_o} L_f y\right)^{-1} \ddot{y} = b(y, \dot{y}) + F_o \,. \tag{8.4}$$

The matrix on the left is the device's inertia as appearing at the end-effector

$$M_{device} = \left(L_{g_o} L_f y\right)^{-1} \,. \tag{8.5}$$

The device's anisotropic inertia can be compensated for if the operator's force F_o can be measured or accurately estimated. In this case, the operator's force can be subtracted out of the system and then reintroduced according to the desired virtual tool dynamics. In essence, this creates a hybrid impedance / admittance controller. The operator's force is cancelled by augmenting the set stabilization controller u_{ss} an additional term as

$$u = u_{ss} - J^\intercal F_o \,. \tag{8.6}$$

The operator's force can then be reintroduced into the system through the tangential controller to produce the accelerations the operator would expect based on the virtual tool's model. The desired dynamics on the set can be reformulated as

$$\ddot{y} = a(y, \dot{y}) + M_{desired}^{-1} F_o \,, \tag{8.7}$$

where $M_{desired}$ is the inertia matrix of the virtual tool. The linearization procedure would then result in the closed-loop tangential dynamics

$$M_{desired} \ddot{y} = M_{desired} a(y, \dot{y}) + F_o \,, \tag{8.8}$$

which is capable emulating the desired virtual tool dynamics in (3.1) more fully than the projective or set stabilization controllers. The function a would be designed to include $M_{desired}^{-1}$ to cancel the pre-multiplying $M_{desired}$ in (8.8).

The desired dynamics are achieved by modifying the outer-loop tangential controller as

$$v^{\|} = \left(\left(\frac{d\psi}{d\hat{t}}^{\top} \frac{d\psi}{d\hat{t}} \right)^{-1} \frac{d\psi}{d\hat{t}}^{\top} \right) \left(a\left(\psi(\hat{t}), \frac{d\psi}{d\hat{t}} \frac{d\hat{t}}{dt} \right) + M_{desired}^{-1} F_o - \frac{d^2\psi}{d\hat{t}^2} \frac{d\hat{t}}{dt} \right). \tag{8.9}$$

The current hardware setup of the parallelogram robot does not collect force or acceleration measurements, so the evaluation of the controller composed of the inner-loop controller (8.6) with outer-loop transversal controller (7.14) and tangential controller (8.9) with the desired dynamics (8.8) is left as part of the future work. If the force measurements are of sufficient quality, this would be expected to decrease the tracking errors and increase the system's maximum renderable stiffness.

8.2.4 Stability of the Haptic System

Sampling time is often cited as an important factor in the stability of haptic systems. The experimental results in the literature typically use a sampling rate of 1 kHz, only occasionally surpassing this mark up to 2048 Hz [70]. With the LUT based implementation of the haptic renderer in Chapter 5, the haptic system could be run at much faster rates. Commercially available data acquisition hardware can run at rates of 1 MHz. To take full advantage of this type of hardware, it is essential to shift the burden of computation offline through the use of a LUT. An experiment could be conducted at these faster update rates to demonstrate the practical impact of the sampling time, identifying to what degree it is the temporal discretization that limits the system's stability as opposed to other system characteristics.

Although it is possible to run the controller at faster rates by better optimizing its software implementation and using more advanced hardware, the sampled data system will always introduce some amount of delay. The formulation of the control problem in terms of set stabilization puts the haptics problem into the form of a feedback linearization problem whose implementation been studied in the sampled data case [137, 138]. The stability would be expected to improve when taking into consideration the effects of the sample and hold blocks introduced by the data acquisition hardware, instead of assuming that the design based on the continuous world can be applied directly to the discrete world. Alternatively, the performance of the system could also be considered from an energy point of view in the context of passivity [70]. This would help ensure the stability of the system in its discrete implementation, eliminating undesired effects like the limit cycles that can arise as contact is quickly and repeatedly made and broken.

Appendices

Appendix A

Experimental Platform

The experimental platform in this thesis, pictured in Figure 2.5, has been used in previous research [139, 65, 64, 140]. It underwent some modifications for the work in this thesis, including an encoder upgrade, the reduction of friction through a re-machining of the joints, and the mounting of a force-torque sensor that, unfortunately, suffered many technical setbacks and could not be incorporated into the present work. The system identification experiments were re-run and the identified parameters are listed in Table A.1, which correspond to the terms in the dynamics equation (2.26) and output equation (2.28).

The system's functional block is in Figure A.1, with details on each element provided below.

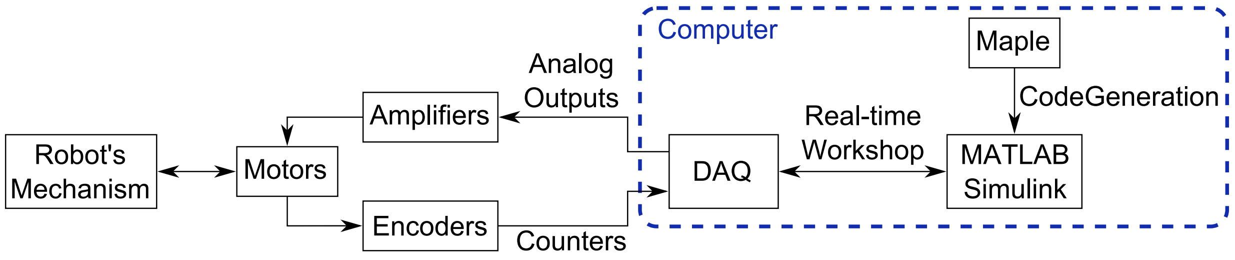


Figure A.1: Functional block diagram of the experimental platform's components.

Table A.1: Identified parameters for the robot model in (2.26) and (2.28)

Parameter	**Value**	**Units**
ℓ_1	0.300	m
ℓ_4	0.346	m
b_1	0.461	kg m/s
b_2	0.105	kg m/s
b_3	0.0336	kg m/s
d_1	0.799	kg m^2
d_2	0.204	kg m^2
d_3	0.01825	kg m^2

A.1 System Components

Computer

The computer's operating system was Microsoft Windows XP Professional 32-bit. It has 3GB of RAM and an Intel Core 2 Q6600 QuadCPU processor running at 2.4GHz. MATLAB R2012a was used to run the experiments in Simulink under its Real-Time Workshop's scheduler.

Data Acquisition Card

A Sensoray Model 626 PCI based data acquisition (DAQ) card was used to read from the encoders and output motor commands to the power amplifier via the analog outputs. Its maximum sampling rate is 1 kHz.

Amplifiers

Three Advanced Motion Controls 50A8 servo amplifiers were used to drive the motors. They accepted a $\pm 10\,V$ signal from the DAQ's analog output, generating a PWM signal switching at 22 kHz to supply current to the motors. The robot's limit switches and emergency stop button were wired into the enable lines on the amplifiers for the sake of safety.

Motors

Three Kollmorgen JR16M4CH ServoDisc DC motors were used, one for each DOF. They are capable of producing a peak torque of 36.8N-cm and a continuous torque of 3.3N-cm. They were driven by the power amplifiers.

Encoders

Three SICK DFS60 (model DFS60A-S4AC65536) encoders are mounted on the end of the motors' shafts. They output a quadrature signal that can be directly input into the DAQ's encoder inputs. They produce 65,536 counts per revolution, which in full quadrature mode generates an angular position resolution of $\approx 2.4 \times 10^{-5}$ rad or $\approx 1.4 \times 10^{-3}$ degrees.

Maple

The dynamic models of the robots and the form of the feedback linearzation controllers were formed in a Maple worksheet. Particular controllers where then generated from these generic expressions. Maple's CodeGeneration package translated the expressions for the controller and forward kinematics into MATLAB code for use in the simulations and experiments.

MATLAB & Simulink

The simulations and experiments were conducted in MATLAB 2012a. Simulink models were developed using the code exported from Maple. The experiments were run under MATLAB's Real-Time Workshop for the Real-Time Windows Target.

References

[1] F. Salsedo, S. Marcheschi, M. Fontana, and M. Bergamasco, "Tactile transducer based on electromechanical solenoids," in *World Haptics Conference (WHC), 2011 IEEE*, june 2011, pp. 581 –586. xii, 9

[2] J. Blake and H. Gurocak, "Haptic glove with mr brakes for virtual reality," *Mechatronics, IEEE/ASME Transactions on*, vol. 14, no. 5, pp. 606 –615, oct. 2009. xii, 10

[3] E. L. Faulring, J. E. Colgate, and M. A. Peshkin, "The cobotic hand controller: Design, control and performance of a novel haptic display," *The International Journal of Robotics Research*, vol. 25, no. 11, pp. 1099–1119, 2006. [Online]. Available: http://ijr.sagepub.com/content/25/11/1099.abstract xii, 11

[4] M. Sato, "Development of string-based force display: Spidar," in *8th International Conference on Virtual Systems and Multimedia*. Citeseer, 2002. xii, 12

[5] L. Cao, W. Ba, and J. Liu, "Computation of the medial axis of planar domains based on saddle point programming," *Computer-Aided Design*, vol. 43, no. 8, pp. 979 – 988, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010448511000625 xiii, 56

[6] B. Hannaford, "A design framework for teleoperators with kinesthetic feedback," *Robotics and Automation, IEEE Transactions on*, vol. 5, no. 4, pp. 426–434, 1989. 1

[7] Merriam-Webster, "Haptic - definition and more," http://www.merriam-webster.com/dictionary/haptic, 2013, accessed: 2013/02/08. [Online]. Available: http://www.merriam-webster.com/dictionary/haptic 1

[8] K. Walker, C. Nielsen, and D. Wang, "The effects of constraint curvature on projective and set-stabilization controllers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, October 2012, pp. 1481 –1486. 2, 109, 126, 128

[9] K. C. Walker and D. W. L. Wang, "The haptic rendering of rigid objects by mapping the model to a simple domain," *Submitted to IEEE Transactions on Haptics*, 2013. 2, 81, 89, 90, 101

[10] K. C. Walker, C. Nielsen, and D. W. L. Wang, "Stabilization of a closed surface for a robotic manipulator," *Accepted to IEEE Transactions on Robotics*, 2012. 2, 126, 129, 148

[11] K. Walker and D. Wang, "Physically-based analytical modelling of deformable haptic environments," in *Haptics Symposium, 2010 IEEE*, 2010, pp. 445 –452. 3, 42

[12] K. C. Walker and D. W. L. Wang, "Analytical modelling of deformable objects for haptics virtual environments," *International Journal of Robotics and Automation*, vol. 27, no. 1, 2012. [Online]. Available: http://dx.doi.org/10.2316/Journal.206.2012.1.206-3535 3, 42

[13] M. Lin and M. Otaduy, *Haptic rendering: foundations, algorithms, and applications*, ser. Ak Peters Series. A.K. Peters, 2008. [Online]. Available: http://books.google.ca/books?id=OMkZAQAAIAAJ 7, 14, 17, 107

[14] M. Otaduy and M. Lin, *High Fidelity Haptic Rendering*, ser. Synthesis lectures in computer graphics and animation. Morgan & Claypool Publishers, 2006. [Online]. Available: http://books.google.ca/books?id=lk0StvDRoEMC 7, 14

[15] A. El Saddik, M. Orozco, M. Eid, and J. Cha, *Haptics Technologies: Bringing Touch to Multimedia*, ser. Springer Series on Touch and Haptic Systems. Springer, 2011. [Online]. Available: http://books.google.ca/books?id=0d9i9-8y268C 7, 9, 10, 39

[16] T. Kern, *Engineering Haptic Devices*. Springer Science & Business Media, 2009. [Online]. Available: http://books.google.ca/books?id=70kJ4RFn_ZMC 7

[17] J. Hollerbach and D. Johnson, "Virtual environment rendering," *Human and Machine Haptics*, 2000. 7, 56, 58

[18] K. Salisbury, F. Conti, and F. Barbagli, "Haptic rendering: Introductory concepts," *Computer Graphics and Applications, IEEE*, vol. 24, no. 2, pp. 24–32, 2004. 7

[19] C. Basdogan, S. Laycock, A. Day, V. Patoglu, and R. Gillespie, *Haptic Rendering*. A.K. Peters, 2007, ch. 3-Dof Haptic Rendering, pp. 311–333. 7

[20] S. Laycock and A. Day, "A Survey of Haptic Rendering Techniques," *Computer Graphics Forum*, vol. 26, no. 1, pp. 50–65, 2007. [Online]. Available: http://www.blackwell-synergy.com/doi/abs/10.1111/j.1467-8659.2007.00945.x 7

[21] L. Marchal-Crespo and D. Reinkensmeyer, "Review of control strategies for robotic movement training after neurologic injury," *Journal of neuroengineering and rehabilitation*, vol. 6, no. 1, p. 20, 2009. 7

[22] R. Avila and L. Sobierajski, "A haptic interaction method for volume visualization," in *Visualization'96. Proceedings.* IEEE, 1996, pp. 197–204. 7, 56

[23] J. C. Roberts and S. Paneels, "Where are we with haptic visualization?" in *EuroHaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2007. Second Joint*, march 2007, pp. 316 –323. 7

[24] T. Coles, D. Meglan, and N. John, "The role of haptics in medical training simulators: A survey of the state of the art," *Haptics, IEEE Transactions on*, vol. 4, no. 1, pp. 51 –66, jan.-feb. 2011. 8

[25] R. Adams and B. Hannaford, "Stable haptic interaction with virtual environments," *IEEE Transactions on robotics and Automation*, vol. 15, no. 3, pp. 465–474, 1999. 8

[26] D. Lawrence, "Stability and transparency in bilateral teleoperation," *Robotics and Automation, IEEE Transactions on*, vol. 9, no. 5, pp. 624 –637, oct 1993. 8

[27] M. Otaduy and M. Lin, “A modular haptic rendering algorithm for stable and transparent 6-dof manipulation,” *Robotics, IEEE Transactions on*, vol. 22, no. 4, pp. 751 –762, aug. 2006. 8

[28] A. Turing, “Computing machinery and intelligence,” *Mind*, vol. 59, no. 236, pp. 433–460, 1950. 8

[29] M. Benali-Khoudja, M. Hafez, J. Alexandre, and A. Kheddar, “Tactile interfaces: a state-of-the-art survey,” in *Int. Symposium on Robotics*, vol. 31. Citeseer, 2004. 9

[30] J. Martin and J. Savall, “Mechanisms for haptic torque feedback,” in *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*, march 2005, pp. 611 – 614. 10

[31] J. Moore, C.A., M. Peshkin, and J. Colgate, “Cobot implementation of virtual paths and 3d virtual surfaces,” *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 2, pp. 347 – 351, apr 2003. 11

[32] M. Inc., “Haptics and robotics - moog,” http://www.moog.com/products/haptics-robotics/, 2013, accessed: 2013/01/21. [Online]. Available: http://www.moog.com/products/haptics-robotics/ 11

[33] J. Colgate and J. Brown, “Factors affecting the z-width of a haptic display,” in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, may 1994, pp. 3205 –3210 vol.4. 11

[34] Geomagic, “Haptic devices - sensable,” http://www.sensable.com/products-haptic-devices.htm, 2013, accessed: 2013/01/21. [Online]. Available: http://www.sensable.com/products-haptic-devices.htm 12, 92

[35] F. Dimension, “Force dimension - products,” http://www.forcedimension.com/products, 2013, accessed: 2013/01/21. [Online]. Available: http://www.forcedimension.com/products 12

[36] H. S. A., "Haption sa - products," http://www.haption.com/site/index.php/en/products-menu-en, 2013, accessed: 2013/01/21. [Online]. Available: http://www.haption.com/site/index.php/en/products-menu-en 12

[37] Q. Inc., "Specialty challenges," http://www.quanser.com/english/html/products/fs_product_challenge.asp?lang_code=english&pcat_code=exp-spe&prod_code=S27-5dofHaptic, 2013, accessed: 2013/01/21. [Online]. Available: http://www.quanser.com/english/html/products/fs_product_challenge.asp?lang_code=english&pcat_code=exp-spe&prod_code=S27-5dofHaptic 12

[38] I. Barrett Tehcnology, "Barrett tehcnology, inc." http://barrett.com/robot/index.htm, 2013, accessed: 2013/01/21. [Online]. Available: http://barrett.com/robot/index.htm 12

[39] M. Spong and M. Vidyasagar, *Robot dynamics and control.* Wiley, 1989. [Online]. Available: http://books.google.ca/books?id=6SoQRAAACAAJ 13, 19, 20

[40] A. Liu, F. Tendick, K. Cleary, and C. Kaufmann, "A survey of surgical simulation: applications, technology, and education," *Presence: Teleoperators & Virtual Environments*, vol. 12, no. 6, pp. 599–614, 2003. 13

[41] S. De, Y.-J. Lim, M. Manivannan, and M. A. Srinivasan, "Physically realistic virtual surgery using the point-associated finite field (paff) approach," *Presence: Teleoperators & Virtual Environments*, vol. 15, no. 3, pp. 294–308, 2006. [Online]. Available: http://www.mitpressjournals.org/doi/abs/10.1162/pres.15.3.294 13

[42] J. Barbic and D. James, "Six-dof haptic rendering of contact between geometrically complex reduced deformable models," *Haptics, IEEE Transactions on*, vol. 1, no. 1, pp. 39 –52, jan.-june 2008. 14

[43] O. Astley and V. Hayward, "Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2, May 1998, pp. 989 –994 vol.2. 14

[44] C. Zilles and J. Salisbury, "A constraint-based god-object method for haptic display," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 146–151, Aug 1995. 16, 42, 58, 63, 101, 127

[45] C. Ho, C. Basdogan, and M. Srinivasan, "Efficient point-based rendering techniques for haptic display of virtual objects," *Presence*, vol. 8, no. 5, pp. 477–491, 1999. 16, 17, 63, 107

[46] D. C. Ruspini, K. Kolarov, and O. Khatib, "The Haptic Display of Complex Graphical Environments," *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 345–352, 1997. 16, 53, 58, 63, 66, 128

[47] T. Thompson, I. David, E. Johnson, and E. Cohen, "Direct haptic rendering of sculptured models," in *in Proc. 1997 Symposium on Interactive 3D Graphics*, 1997. 16, 58, 64, 128

[48] J. Bloomenthal and C. Bajaj, *Introduction to Implicit Surfaces*, ser. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann Publishers, Incorporated, 1997. [Online]. Available: http://books.google.ca/books?id=T3SSqIVnS4YC 17

[49] J.-K. Lee and Y. J. Kim, "Haptic Rendering of Point Set Surfaces," *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2007. 17, 56

[50] J. El-Sana and A. Varshney, "Continuously-adaptive haptic rendering," in *Virtual Environments*, vol. 2000, 2000, p. 135. 18

[51] E. Ruffaldi, D. Morris, T. Edmunds, F. Barbagli, and D. K.Pai, "Standardized Evaluation of Haptic Rendering Systems," *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems 2006*, pp. 225–232, 2006. 18, 41

[52] S. Niku, *Introduction to Robotics*. John Wiley & Sons, 2010. [Online]. Available: http://books.google.ca/books?id=2V4aGvlGt7IC 19

[53] B. Siciliano and L. Villani, *Robot force control*. Kluwer Academic Publishers, 1999. 19, 104

[54] M. Spong and S. Hutchinson, *Robot Modeling and Control*. Wiley, 2005. [Online]. Available: http://books.google.ca/books?id=wGapQAAACAAJ 21

[55] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC Press, 1994. 21

[56] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of Intelligent and Robotic Systems*, vol. 3, pp. 201–212, 1990. [Online]. Available: http://dx.doi.org/10.1007/BF00126069 21

[57] C. Carignan and K. Cleary, "Closed-loop force control for haptic simulation of virtual environments," *Haptics-e*, vol. 1, no. 2, pp. 1–14, 2000. 22

[58] M. Ueberle and M. Buss, "Control of kinesthetic haptic interfaces," in *Proc. IEEE/RSJ Int. Conf. on Intellig. Rob. and Syst., Workshop on Touch and Haptics*, 2004. 22

[59] M. Peshkin, J. Colgate, W. Wannasuphoprasit, C. Moore, R. Gillespie, and P. Akella, "Cobot architecture," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 4, pp. 377 –390, aug 2001. 24

[60] G. Liu, "On velocity estimation using position measurements," in *American Control Conference, 2002. Proceedings of the 2002*, 2002, pp. 1115–1120. 24

[61] A. Hladio, C. Nielsen, and D. W. L. Wang, "Path Following Controller Design for a Class of Mechanical Systems," in *18th IFAC World Congress*. IEEE, 2011, pp. 89–95. 25, 135, 136

[62] A. Isidori, *Nonlinear control systems*, ser. Communications and control engineering series. Springer, 1995, no. 1. [Online]. Available: http://books.google.ca/books?id=fPGzHK_pto4C 26, 112, 113, 114

[63] A. Hladio, C. Nielsen, and D. Wang, "Path following for mechanical systems: Experiments and examples," in *American Control Conference (ACC), 2011*, July 2011, pp. 551–556. 26, 27, 131

[64] M. Ching, "A five-bar-linkage force reflecting interface for a virtual reality system via implicit force control," Master's thesis, University of Waterloo, 1996. 30, 31, 167

[65] D. R. Madill, "Modelling and control of a haptic interface, a mechatronics approach," Master's thesis, University of Waterloo, 1999. 30, 31, 144, 167

[66] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035 – 2057, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109806002871 35

[67] J. Colgate, M. Stanley, and J. Brown, "Issues in the Haptic Display of Tool Use," *Proceedings of the ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 140–144, 1994. 36, 111

[68] M. Sirouspour, S. DiMaio, S. Salcudean, P. Abolmaesumi, and C. Jones, "Haptic interface control-design issues and experiments with a planar device," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, 2000, pp. 789 –794 vol.1. 36

[69] S. Moghimi, S. Sirouspour, and P. Malysz, "Haptic-enabled collaborative training with generalized force and position mappings," in *Haptic interfaces for virtual environment and teleoperator systems, 2008. haptics 2008. symposium on*, march 2008, pp. 287 –294. 36

[70] A. Abdossalami and S. Sirouspour, "Adaptive control for improved transparency in haptic simulations," *Haptics, IEEE Transactions on*, vol. 2, no. 1, pp. 2 –14, jan.-march 2009. 37, 164, 165

[71] V. Hayward and O. Astley, "Performance measures for haptic interfaces," in *1996 Robotics Research The 7th International Symposium*. Springer Verlag, 1996, pp. 195–207. 39, 40

[72] M. Moreyra and B. Hannaford, “A practical measure of dynamic response of haptic devices,” in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1, may 1998, pp. 369 –374 vol.1. 39, 40

[73] M. Khan, S. Sulaiman, M. Said, and M. Tahir, “Exploring the quantitative and qualitative measures for haptic systems,” in *Information Technology (ITSim), 2010 International Symposium in*, vol. 1, june 2010, pp. 31 –36. 39, 40

[74] J. Colgate, M. Stanley, and J. Brown, “Issues in the haptic display of tool use,” in *1995 IEEE/RSJ International Conference on Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings*, vol. 3, 1995. 39, 40, 110

[75] R. Höver, M. D. Luca, and M. Harders, “User-based evaluation of data-driven haptic rendering,” *ACM Trans. Appl. Percept.*, vol. 8, no. 1, pp. 7:1–7:23, Nov. 2010. [Online]. Available: http://doi.acm.org/10.1145/1857893.1857900 39, 40

[76] D. Lawrence, L. Pao, A. Dougherty, M. Salada, and Y. Pavlou, “Rate-hardness: a new performance metric for haptic interfaces,” *Robotics and Automation, IEEE Transactions on*, vol. 16, no. 4, pp. 357 –371, aug 2000. 39

[77] A. Crossan, “The design and evaluation of a haptic veterinary palpation training simulator,” Ph.D. dissertation, University of Glasgow, Glasgow, Scotland, 2003. 40

[78] S. Lederman and L. Jones, “Tactile and haptic illusions,” *Haptics, IEEE Transactions on*, vol. 4, no. 4, pp. 273–294, 2011. 41

[79] S. J. Lederman and R. L. Klatzky, “Haptic perception: A tutorial,” *Attention, Perception, & Psychophysics*, vol. 71, no. 7, pp. 1439–1459, 2009. 41

[80] A. Kirkpatrick and S. Douglas, “Application-based evaluation of haptic interfaces,” in *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on.* IEEE, 2002, pp. 32–39. 41

[81] A. Pressley, *Elementary Differential Geometry*, ser. Springer Undergraduate Mathematics. Springer, 2010. [Online]. Available: http://books.google.ca/books?id=9nT1fOwATf0C 46, 72, 81, 146

[82] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of Physics Extended, 8th Ed.* Wiley India Pvt. Limited, 2008. [Online]. Available: http://books.google.ca/books?id=RVCE4EUjDCgC 49, 50

[83] L. Kim, A. Kyrikou, M. Desbrun, and G. Sukhatme, "An implicit-based haptic rendering technique," in *In Proceeedings of the IEEE/RSJ International Conference on Intelligent Robots*, vol. 3, 2002, pp. 2943–2948. 51, 56, 63

[84] V. Patoglu and R. Gillespie, "A Closest Point Algorithm for Parametric Surfaces with Global Uniform Asymptotic Stability," in *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005*, 2005, pp. 348–355. 56, 57

[85] K. Salisbury and C. Tarr, "Haptic Rendering of Surfaces Defined by Implicit Functions," in *Proceedings of the ASME 6th Annual Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, November 1997, pp. 61–68. 56

[86] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Voxel-Based 6-DOF Haptic Rendering Improvements," *Haptics-e*, vol. 3, no. 7, 2006. 56

[87] E. Ruffaldi, D. Morris, F. Barbagli, K. Salisbury, and M. Bergamasco, "Voxel-Based Haptic Rendering Using Implicit Sphere Trees," *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems 2008*, pp. 319–325, 2008. 56

[88] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles, "Haptic rendering: programming touch interaction with virtual objects," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, ser. I3D '95. New York, NY, USA: ACM, 1995, pp. 123–130. 56, 65, 66

[89] H. Blum, "A transformation for extracting new descriptors of shape," *Models for the perception of speech and visual form*, vol. 19, no. 5, pp. 362–380, 1967. 56

[90] K. Hirota and M. Hirose, “Development of surface display,” in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, sep 1993, pp. 256 –262. 58

[91] Y. Adachi, T. Kumano, and K. Ogino, “Intermediate representation for stiff virtual objects,” in *Proceedings of the Virtual Reality Annual International Symposium.* Published by the IEEE Computer Society, March 1995, pp. 203–210. 58

[92] C.-h. Ho, C. Basdogan, and A. A. Srinivasan, “Haptic rendering: Point- and ray-based interactions,” in *Proceedings of the Second PHANToM Users Group Workshop*, 1997. 58

[93] W. Mark, S. Randolph, M. Finch, J. Van Verth, I. Taylor, and M. Russell, “Adding force feedback to graphics systems: Issues and solutions,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques.* ACM, 1996, pp. 447–452. 58

[94] T. Massie and J. Salisbury, “The phantom haptic interface: A device for probing virtual objects,” in *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, vol. 55, 1994, pp. 295–300. 58

[95] S. P. Walker and J. K. Salisbury, “Large haptic topographic maps: marsview and the proxy graph algorithm,” in *Proceedings of the 2003 symposium on Interactive 3D graphics*, ser. I3D ’03. New York, NY, USA: ACM, 2003, pp. 83–92. [Online]. Available: http://doi.acm.org/10.1145/641480.641499 58

[96] Z. Pezzementi, A. Okamura, and G. Hager, “Dynamic guidance with pseudoadmittance virtual fixtures,” in *Robotics and Automation, 2007 IEEE International Conference on*, april 2007, pp. 1761 –1767. 63

[97] H. Morgenbesser and M. Srinivasan, “Force shading for haptic shape perception,” in *Proceedings of the ASME Dynamic Systems and Control Division*, vol. 58. American Society of Mechanical Engineers, 1996, pp. 407–412. 65, 96

[98] J. Fritz and K. Barner, "Stochastic models for haptic texture," in *Proceedings of SPIEs International Symposium on Intelligent Systems and Advanced Manufacturing–Telemanipulator and Telepresence Technologies III*. Citeseer, 1996, pp. 34–44. 65, 108

[99] H. Culbertson, J. Romano, P. Castillo, M. Mintz, and K. Kuchenbecker, "Refined methods for creating realistic haptic virtual textures from tool-mediated contact acceleration data," in *Haptics Symposium (HAPTICS), 2012 IEEE*, march 2012, pp. 385 –391. 66

[100] N. Melder and W. S. Harwin, "Force Shading and Bump Mapping using the Friction Cone Algorithm," *Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2005. 66

[101] D. Constantinescu, S. Salcudean, and E. Croft, "Haptic rendering of rigid contacts using impulsive and penalty forces," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 309 – 323, june 2005. 68

[102] K. Kuchenbecker, J. Fiene, and G. Niemeyer, "Improving contact realism through event-based haptic feedback," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 2, pp. 219 –230, march-april 2006. 68, 98

[103] J. Beutel and M. Sonka, *Handbook of Medical Imaging: Medical image processing and analysis*, ser. Press Monographs. Spie Press, 2000. [Online]. Available: http://books.google.ca/books?id=NQSaj6kJFVkC 71

[104] M. Miller, "Computational anatomy: shape, growth, and atrophy comparison via diffeomorphisms," *NeuroImage*, vol. 23, pp. S19–S33, 2004. 71

[105] S. Joshi and M. Miller, "Landmark matching via large deformation diffeomorphisms," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1357–1370, 2000. 71

[106] J. Sethian, *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, ser. Cambridge Monographs on Applied and Computational Mathematics. Cambridge Univ Pr, 1999, no. 3. 72, 77, 79

[107] S. Osher and J. Sethian, “Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations,” *Journal of computational physics*, vol. 79, no. 1, pp. 12–49, 1988. 74

[108] M. Grayson, “The heat equation shrinks embedded plane curves to round points,” *J. Diff. Geom*, vol. 26, no. 2, pp. 285–314, 1987. 74

[109] M. E. Gage and R. S. Hamilton, “The Heat Euqation Shrinking Convex Plane Curves,” *Journal of Differential Geometry*, vol. 23, no. 1, pp. 69–96, 1986. 74

[110] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces.* Springer Verlag, 2003, vol. 153. 74, 75, 77

[111] C. Sigg, R. Peikert, and M. Gross, “Signed distance transform using graphics hardware,” in *Visualization, 2003. VIS 2003. IEEE.* IEEE, 2003, pp. 83–90. 75

[112] J. Sethian, “Curvature flow and entropy conditions applied to grid generation,” *Journal of Computational Physics*, vol. 115, no. 2, pp. 440 – 454, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999184712095 79, 80, 83

[113] S. J. Lederman and R. L. Klatzky, “Hand movements: A window into haptic object recognition,” *Cognitive Psychology*, vol. 19, no. 3, pp. 342 – 368, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0010028587900089 104

[114] D. Wang, S. Liu, X. Zhang, J. Xiao, J. Hou, and Y. Zhang, “Six degree-of-freedom haptic simulation of periodontal pathological changes,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, oct. 2012, pp. 39 –45. 106

[115] M. Pharr and R. Fernando, *GPU Gems 2: Programming Techniques For High-Performance Graphics And General-Purpose Computation.* Addison-Wesley Professional, 2005. 109

[116] J. Ginsberg, *Advanced engineering dynamics.* Cambridge University Press, 1998. 110

[117] T. Yoshikawa, "Dynamic hybrid position/force control of robot manipulators–description of hand constraints and calculation of joint driving force," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 5, pp. 386 –392, october 1987. 127

[118] J. J. Abbott and A. M. Okamura, "Pseudo-admittance bilateral telemanipulation with guidance virtual fixtures," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 865–884, 2007. [Online]. Available: http://ijr.sagepub.com/content/26/8/865.abstract 127

[119] H. Arai, T. Takubo, Y. Hayashibara, and K. Tanie, "Human-robot cooperative manipulation using a virtual nonholonomic constraint," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 4, 2000, pp. 4063 –4069 vol.4. 127

[120] J. Abbott, G. Hager, and A. Okamura, "Steady-hand teleoperation with virtual fixtures," in *Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on*, oct.-2 nov. 2003, pp. 145 – 151. 127

[121] A. Shiriaev, J. Perram, and C. Canudas-de Wit, "Constructive tool for orbital stabilization of underactuated nonlinear systems: Virtual constraints approach," *Automatic Control, IEEE Transactions on*, vol. 50, no. 8, pp. 1164 – 1176, aug. 2005. 128

[122] L. Consolini and M. Maggiore, "On the swing-up of the pendubot using virtual holonomic constrains," in *Proceedings of the IFAC World Congress*, 2011. 128

[123] D. Liberzon, *Switching in systems and control*. Springer, 2003. 128, 161

[124] G. Hager, "Human-machine cooperative manipulation with vision-based motion constraints," in *Visual Servoing via Advanced Numerical Methods*, ser. Lecture Notes in Control and Information Sciences, G. Chesi and K. Hashimoto, Eds. Springer Berlin / Heidelberg, 2010, vol. 401, pp. 55–70. [Online]. Available: http://dx.doi.org/10.1007/978-1-84996-089-2_4 128

[125] J. Selig, “Curvature in force-position control,” in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2, may 1998, pp. 1761 –1766 vol.2. 128

[126] U. Mettin, P. La Hera, D. Morales, A. Shiriaev, L. Freidovich, and S. Westerberg, “Path-constrained trajectory planning and time-independent motion control: Application to a forestry crane,” in *14th International Conference on Advanced Robotics (ICAR), Proceedings*, 2009. 129

[127] M. Scheint, M. Sobotka, and M. Buss, “Virtual holonomic constraint approach for planar bipedal walking robots extended to double support,” in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*. IEEE, 2009, pp. 8180–8185. 129

[128] S. Pchelkin, A. Shiriaev, L. Freidovich, U. Mettin, S. Gusev, and W. Kwon, “Natural sit-down and chair-rise motions for a humanoid robot,” in *Decision and Control (CDC), 2010 49th IEEE Conference on*, dec. 2010, pp. 1136 –1141. 129

[129] L. Consolini and M. Maggiore, “Control of a bicycle using virtual holonomic constraints,” in *Decision and Control (CDC), 2010 49th IEEE Conference on*, dec. 2010, pp. 5204 – 5209. 129

[130] C. Nielsen, L. Consolini, M. Maggiore, and M. Tosques, “Path following for the pvtol: A set stabilization approach,” in *47th IEEE Conference on Decision and Control, 2008. CDC 2008*, 2008, pp. 584–589. 129

[131] C. Nielsen, C. Fulford, and M. Maggiore, “Path following using transverse feedback linearization: Application to a maglev positioning system,” *Automatica*, 2010. 129

[132] A. Hladio, C. Nielsen, and D. Wang, “Path following for a class of mechanical systems,” *IEEE Transactions on Control Systems Technology*, to appear. 129

[133] C. Nielsen, “Set stabilization using transverse feedback linearization,” Ph.D. dissertation, University of Toronto, 2009. 129, 136, 146

[134] J. M. Lee, *Introduction to Smooth Manifolds*. New York: Springer, 2002. 130, 146

[135] M. Tavakoli, A. Aziminejad, R. Patel, and M. Moallem, "Enhanced transparency in haptics-based master-slave systems," in *American Control Conference, 2007. ACC '07*, july 2007, pp. 1455 –1460. 162

[136] H. Asada, "Dynamic analysis and design of robot manipulators using inertia ellipsoids," in *Robotics and Automation. Proceedings. 1984 IEEE International Conference on*, vol. 1, mar 1984, pp. 94 – 102. 162

[137] A. Arapostathis, B. Jakubczyk, H.-G. Lee, S. Marcus, and E. Sontag, "The effect of sampling on linear equivalence and feedback linearization," *Systems & Control Letters*, vol. 13, no. 5, pp. 373 – 381, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167691189901035 164

[138] J. Grizzle, "Feedback linearization of discrete-time systems," in *Analysis and Optimization of Systems: Proceedings of the Seventh International Conference on Analysis and Optimization of Systems, Antibes, June 25-27, 1986*. Springer, 1986, p. 273. 164

[139] J. M. Daly, "Output Feedback Bilateral Teleoperation with Force Estimation in the Presence of Time Delays," Ph.D. dissertation, University of Waterloo, 2010. 167

[140] A. Hladio, "Path following for mechanical systems applied to robotic manipulators," Master's thesis, University of Waterloo, 2010. 167