HAPTIC RENDERING OF CONTINUOUS PARAMETRIC MODELS

by

MELDA ULUSOY

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Spring 2010

HAPTIC RENDERING OF
CONTINUOUS PARAMETRIC MODELS

APPROVED BY

Assist. Prof. Dr. Volkan PATOĞLU .....
(Thesis Supervisor)

Assoc. Prof. Dr. Kemalettin ERBATUR .....

Assist. Prof. Dr. Esra ERDEM .....

Assist. Prof. Dr. Ahmet ONAT .....

Assist. Prof. Dr. Gözde ÜNAL .....

DATE OF APPROVAL: .....

To my family...

Acknowledgments

First of all I would like to thank my thesis advisor Assist. Prof. Dr. Volkan Patoğlu for his abundant help, guidance and assistance in numerous ways. I would also like to express my appreciation to Assoc. Prof. Dr. Kemalettin Erbatur, Assist. Prof. Dr. Ahmet Onat, Assist. Prof. Dr. Esra Erdem and Assist. Prof. Dr. Gözde Ünal for spending time on my thesis and for their guidance.

I am grateful to all my friends at Sabancı University Mechatronics Lab. Special thanks to Ayşe Neşe Tüfekçiler, Ahmetcan Erdoğan, Ozan Tokatlı, Elif Hocaoğlu, Aykut Cihan Satıcı and İsmail Hakan Ertaş for their love and enjoyable conversations during the long hours in the lab.

Finally my most special thanks go to my best partner and friend, my husband, Alphan Ulusoy. This work would not have happened without his love and support.

This thesis was financially supported by scholarships from Sabancı University and the Scientific and Technological Research Council of Turkey, TÜBİTAK.

HAPTIC RENDERING OF CONTINUOUS PARAMETRIC MODELS

Abstract

Haptic rendering is the process of computing restoring forces that are required to generate a perception of touch between a user and a virtual environment. The realism of haptic rendering depends mainly on haptic rendering algorithms and the modeling of virtual objects in a virtual environment. Friction and texture rendering also play an important role in increasing the realism of the experience between a user and a virtual environment. The state of the art haptic and friction rendering algorithms in the literature are developed for polygonal models. These approaches can not benefit from the advantages of continuous parametric surfaces such as compact representation, higher order continuity and exact computation of surface normals.

In this thesis, a feedback-stabilized closest point tracking based haptic rendering algorithm is extended by introducing a direct friction rendering method for continuous parametric surfaces. Unlike the existing approaches, the proposed friction rendering method is direct and does not rely on the algorithms introduced for polyhedral surfaces. This algorithm implements the stiction model of friction for haptic rendering of parametric surfaces. It can directly operate on parametric models and can handle surfaces with high curvature. Furthermore, the algorithm allows transitions from sticking to sliding and sliding to sticking, as well as surface to surface transitions, without introducing discontinuous force artifacts. The algorithm also allows for tuning of the friction coefficient during the mode transitions to enable rendering of the Stribeck effect. Thanks to its feedback-stabilized core, it is robust

against drift and numerical noise. The algorithm is computationally efficient (with respect to time and space); its applicability and effectiveness to simulate friction are verified through simulations and real-time implementations. In particular, the friction rendering algorithm is tested using pre-determined trajectories that demonstrate successful rendering of static friction at a corner, the mode changes from static-to-dynamic and dynamic-to-static friction.

Özet

Haptik gerçekleme kullanıcının sanal ortamla etkileşimindeki dokunma hissinin kullanıcıya aktarılabilmesi için gereken kuvvetlerin hesaplanması işlemidir. Haptik gerçeklemenin gerçeğe yakn algılanabilmesi yüksek oranda haptik gerçekleme algoritmalarına ve sanal ortamda yaratılan sanal nesnelerin modellenme yöntemine bağlıdır. Sürtünme ve doku gerçeklemesi de kullanıcıyla sanal ortam arasındaki deneyimin gerçeğe yakınlığını arttırmada önemli rol oynar. Literatürdeki haptik ve sürtünme gerçekleme algoritmalarının çoğu çok yüzeyli modeller için geliştirilmiştir. Bu algoritmalar sürekli parametrik yüzeylerin, kısa gösterim, yüksek derecede süreklilik ve yüzey normallerinin hatasız hesaplanması gibi avantajlarından faydalanamazlar.

Bu tezde sürekli parametrik modeller üzerinde doğrudan çalışabilen bir sürtünme gerçekleme algoritması önerilerek geri besleme kararlılaştırmalı en yakın nokta takibine dayalı haptik gerçekleme algoritması geliştirilmiştir. Varolan diğer yaklaşımlardan farklı olarak önerilen sürtünme gerçekleme metodu çok düzlemli yüzeyler için geliştirilen algoritmalara dayanmamaktadır. Bu algoritma sürekli parametrik modellerin kullanıldığı haptik gerçeklemede kayma yuvarlanma sürtünmesini gerçeklemektedir. Yüksek eğimli yüzeyler de dahil olmak üzere parametrik modeller üzerinde doğrudan çalışabilir. Bunların dışında algoritma statik sürtünmeden dinamik sürtünmeye ve dinamik sürtünmeden statik sürtünmeye geçişe kuvvet sürekliliğini garanti

altında tutarak izin vermektedir. Ayrıca sürtünme katsayının ayarlanmasına izin vererek Stribeck efektini gerçeklemektedir. Geri beslemeli kararlılaştırma alt yapısı sayesinde sapmaya ve nümerik gürültüye karşı gürbüzdür. Önerilen algoritma işlemsel olarak verimlidir, sürtünmeyi uygulayabilmesi ve yeterliliği benzetimlerle ve gerçek zamanlı gerçekleme ile gösterilmiştir. Statikten dinamiğe, dinamikten statik sürtünmeye geçişleri ve köşede statik sürtünmeyi sağlayacak önceden belirlenmiş referanslar sürtünme algoritması üstünde test edilmiştir.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents literature review on haptic rendering techniques, collision detection methods used in haptic rendering, friction models and friction rendering approaches.

1.1 Haptic Rendering

Recent advances in virtual reality (VR) have rendered realistic interactions with virtual environments possible through various senses of humans such as hearing, sight and touch. Haptic, (Greek for *touch*), interfaces are typical in VR applications due to their capability to transmit physical interactions in a virtual environment to the user through the sense of touch. In the past decade haptics has been broadly used in novel applications such as surgical training, virtual prototyping, molecular docking, telerobotics and teleoperation. These novel applications and enormous interest in the field of haptics point out the significance of haptics in realistic perception of virtual environments by the user.

Haptic Rendering is the process of computing forces arising from the in-

teraction between a user and a virtual environment. These forces are fed back to the user by means of a haptic interface so that a mechanical interaction with computationally mediated environments is possible. When compared to visual and audio rendering, haptic rendering has demanding computational requirements. While visual rendering of virtual environments can be carried out at 30-60 Hz, to maintain stability while representing smooth and realistic contact forces, haptic rendering requires haptic update rates of 1kHz or higher[2]. Otherwise, the haptic system can become unstable and force discontinuities may occur.

Haptic rendering algorithms have been developed to render force feedback correctly to the human operator. The structure of a haptic rendering algorithm mainly consists of collision detection and force-response. Collision detection is needed to detect collisions between the human operator and virtual objects. If a collision is detected, force-response algorithm takes care of computing the interaction force between human operators and virtual objects.

The components of a haptic rendering system can be classified as a haptic interface and a computer-simulated virtual environment. Haptic interface and virtual environment are linked by means of a haptic rendering algorithm. The top portion of Figure 1.1.1 depicts the real physical interaction between a human operator and a haptic interface. The human operator interacts with the end-effector of the haptic interface through the finger-tip. The bottom portion of Figure 1.1.1 represents the computer-simulated virtual environment of the same haptic rendering system in which the point E corresponding to the end-effector of the haptic interface is connected to a proxy P through a *virtual coupler*. A and B are virtual objects that the proxy P can interact

with.

Virtual coupling is introduced in [3] and extended in [4]. It consists of a virtual spring and a virtual damper connected in parallel, and links the haptic display with the virtual environment. The use of a virtual coupler guarantees the stability of the haptic display by eliminating the energy introduced to the system due to numerical calculation. The main idea behind the virtual coupling is that the virtual coupler extracts more energy from the system than the energy introduced by the haptic rendering algorithm and this makes the coupled rendering algorithm and haptic coupler system passive. Common experience suggests that humans remain passive when interacting with passive systems such as dampers, masses [5]. Therefore, the human operator is considered as a passive impedance. Since the haptic display itself is passive, the passivity of overall system can then be guaranteed if the haptic rendering algorithm and the virtual coupler can be made passive. Finally, under mild observability assumption passivity implies stability.

The problem in maintaining the passivity of a haptic display arises due to the sampled-data effects of the haptic display. A necessary and sufficient condition for the passivity of a haptic rendering system is

$$b > \frac{KT}{2} + |B| \tag{1.1}$$

where b represents the physical damping present in the mechanism, T is the sampling rate, K the virtual stiffness, and B the virtual damping coefficient [6]. The energy leak due to the errors introduced by the sampled-data system should be dissipated by the physical damping present in the mechanism or by the artificial damping introduced by the controller.

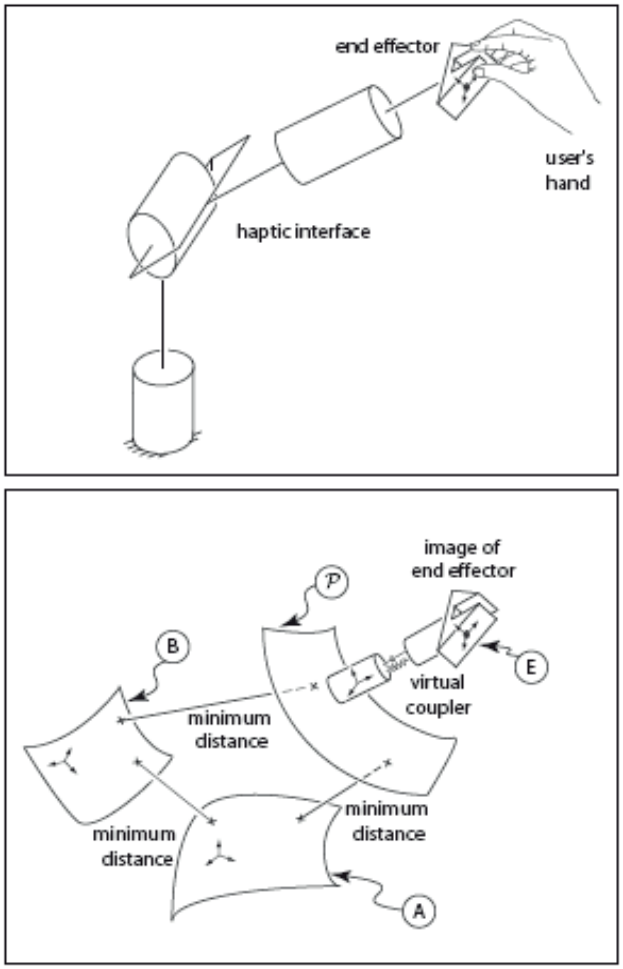

Figure 1.1.1: Representation of a haptic rendering system. The top figure depicts the real physical interaction between the human operator and the haptic interface whereas the figure below corresponds the computer-simulated virtual environment of the same haptic rendering system [1].

1.2 Haptic Rendering Techniques

Haptic rendering techniques are highly dependent of the underlying representation of the geometric models used to form the virtual environment. The approaches based on volumetric representations define vector fields inside the virtual objects that correspond to the restoring forces used for rendering [7]. The approaches that depend on surface representations are commonly developed for a single-point interaction, where the user interacts with the virtual environment through the point end-effector of a haptic device. In these approaches, the restoring force is calculated proportional to the amount of penetration of the haptic interaction point (HIP), the virtual image of the end effector point, into the surface of the virtual object. To calculate the relevant penetration depth and corresponding direction of the restoring force, a point on the surface that is (locally) closest to the HIP, called the closest point (CP), needs to be calculated. Hence, the closest point algorithms used to calculate the CP constitute a crucial element of the haptic rendering.

Surface representations are commonly defined using either polyhedral or parametric models. For polyhedral models, haptic rendering algorithms utilize the state of the art algorithms [8–10] for computing the distance between convex polyhedra to detect collisions and to locate the interaction surface. During haptic rendering, only the (local) closest point to HIP on the relevant interaction surface is of interest. A god-object (GO) is defined as a convenient means of locating the relevant CP [11]. GO is a point that is constrained to move on the interaction surface. As HIP moves, GO tracks it on the interaction surface. When the interaction surface changes, GO transitions to the new interaction surface by tracking HIP in a continuous manner. In addition to a closest point tracking algorithm, the polyhedral models necessitate use

of further algorithms to render refined, realistic virtual worlds. Since sharp corners inherent polyhedral models cause force discontinuities during edge crossings, smoothing techniques, such as force shading, need to be utilized to render the feel of smooth surfaces [11, 12]. In addition, since the numerical tolerances can cause GO to fall through the cracks during interaction surface transitions, finite-sized GOs are suggested for use with polyhedral surface representations [13].

Parametric surface models are inherently smooth; therefore, such representations are natural candidates for haptic rendering of smooth and highly curved objects. Moreover, parametric models possess compact representation and offer efficient means to calculate surface normals at the point of interaction [14]. Most of the haptic rendering algorithms proposed in the literature address the parametric representations using indirect approaches. The most common approach is to use (adaptively refined) meshes to convert the problem into a polyhedral one. Other indirect approaches use intermediate tangent representations to convert the problem into a series of locally polyhedral problems [15, 16]. Indirect approaches do not take advantage of inherent smoothness of parametric surfaces. Intermediate representations fail to approximate surfaces with high curvature, while polyhedral approximations to complex models can grow large in the number of polygons.

Closest point algorithms that can locate CP on curved interaction surfaces lie at the core of direct methods for haptic rendering of parametric models. These algorithms are similar to GO methods in that, once properly initialized they update the location CP on the interaction surface to track the movements of HIP. However, compared to the case with polyhedral models, the update rules are more involved for tracking on curved surfaces. One

direct approach to closest point tracking is due to Thompson *et al.* and uses a first-order approximation and knot insertion to locate CP on a parametric surface [17, 18] . Another approach is proposed in [19, 20], in which a globally uniformly asymptotically stable feedback controller is designed to update the location of CP to continually track HIP on the interaction surface. The algorithm is robust to initialization errors on the interaction surface. Moreover, the algorithm has been extended to handle surfaces formed by tiling several surface patches together [21].

1.3 Collision Detection Algorithms

Collision detection is used to determine the contact points of geometric objects and has been mostly used in robotics, computational geometry and computer graphics. Existing collision detection algorithms can be classified as broad and narrow phase algorithms. Broad-phase collision detection algorithms work on identifying colliding objects whereas narrow-phase approaches try to detect colliding primitives. Most of the narrow-phase collision detection algorithms presented in the literature that are mostly based on the algorithms proposed by Lin and Canny [9] and Gilbert [8], are developed for polygonal models of continuously defined objects. In [8], a convex optimization algorithm computes the separation distance between two convex polyhedra by using a Minkowski-sum iteration. Lin and Canny [9] proposed the *Voronoi marching* algorithm for computing separation distance by tracking the closest features between convex polyhedra. The algorithm makes use of Voronoi regions of the polyhedral object in order to find the closest features between two polyhedra. The collision detection algorithm based on the Lin-

Canny algorithm is improved in [22] by introducing V-Clip (Voronoi-Clip) system and in [23] by proposing the SWIFT algorithm that works based on bounding volume hierarchies.

Collision detection algorithms that can directly work on parametric models, are less in number when compared to the algorithms developed for polygonal models. Such an algorithm is presented in [24] by Lin and Manocha that works on curved models composed of spline or algebraic surfaces. The algorithms developed in [25] and [26] detect collisions based on solving a constrained minimization problem using iterative Newton methods. The work in [27] demonstrates a direct parametric tracing method for sculptured models. The proposed tracking algorithm tracks the closest point on the surface after initializing it with the closest point and works based on Newton iterations. This algorithm is enhanced to work on moving surfaces in [28].

1.4 Friction Rendering

Friction is one of the fundamental aspects while physically interacting with an environment. Grasp and manipulation of objects are only possible through the existence of friction force. Furthermore, friction aids in the perception of the shape and the texture of objects. Due to these significant aspects of friction, friction rendering is an indispensable module of haptic rendering systems. Human mechanoreceptors are very sensitive to even small changes in the friction force such as the difference between the static and dynamic friction due to the static and dynamic coefficient of friction [29]. Therefore, friction force should be accurately computed and rendered via a haptic interface so that the human operator can perceive the virtual environment

realistically. The challenge by rendering friction lies in the nonlinear characteristic of friction which occurs at zero velocity and causes a discontinuity between static and dynamic friction. In the Sections 1.4.1 and 1.4.2, friction models and improved techniques for friction rendering are broadly discussed.

1.4.1 Friction Models

One of the classical friction models is *Coulomb Friction*, named after Charles-Augustin de Coulomb, computes the friction force F_C proportional to the normal load F_N. This friction model is governed by the equation, $F_C = \mu F_N$ where μ represents the coefficient of friction. The friction force at zero velocity is not determined, it can take any value between F_C and $-F_C$. Figure 1.4.1(a) shows the friction force vs. velocity graph of the Coulomb friction model.

Figure 1.4.1(b) represents the Coulomb friction in the presence of viscous friction. Stiction is depicted in Figure 1.4.1(c) and defined as the friction at rest. In the stiction model of friction, static friction resists against the external forces up to a value and prevents the object from moving. It is a multi-valued function that can take on any value between the two extremes $-FS$ and FS. The static friction force is described as a function of external force F_e and is formulated as follows

$$F = \begin{cases} F_e & \text{if } v = 0 \text{ and } |F_e| < F_S \\ F_S \ \text{sgn}(F_e) & \text{if } v = 0 \text{ and } |F_e| \geq F_S \end{cases} \tag{1.2}$$

Another classical friction model is shown in Figure 1.4.1(d) where the friction force decreases continuously. This is called Stribeck effect [30] and it is represented as

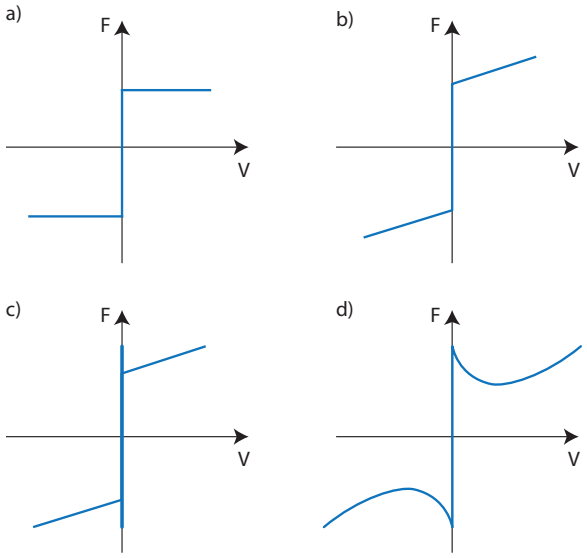

Figure 1.4.1: Friction Models. (a) Coulomb friction model (b) Coulomb friction in presence of viscous friction (c) Stiction model (d) Stribeck effect

$$F = \begin{cases} F(v) & \text{if } v \neq 0 \\ F_e & \text{if } v = 0 \text{ and } |F_e| < F_S \\ F_S \text{ sgn}(F_e) & \text{otherwise} \end{cases} \tag{1.3}$$

where $F(v)$ is an arbitrary function.

The Karnopp friction model is developed by Karnopp [31] to overcome problems with zero velocity detection in simulations. According to this model, the zero velocity is determined using an interval, $|v| < DV$. As long as $|v| < DV$, the friction force is a saturated version of external force as seen in Figure 1.4.2. Otherwise, it is an arbitrary function of velocity. The disadvantage of the model is that it is strongly dependent on the choice of the value DV [32].

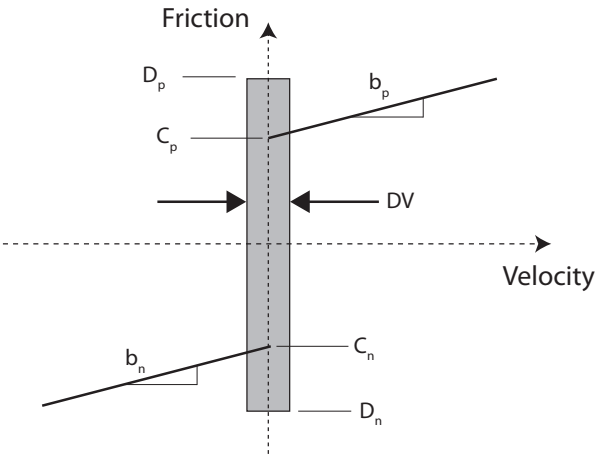

Figure 1.4.2: Karnopp friction model

In addition to the discussed friction models above, dynamic models of friction are explained in the following. Formulations of these models were found empirically.

The Dahl model introduced in [33] was developed for simulating control systems with friction and it was used for adaptive friction compensation. The Dahls friction model is developed based on the stress-strain curve in classical solid mechanics. The following represents Dahl's formulation of friction

$$\frac{dF}{dx} = \sigma(1 - \frac{F}{F_c}\text{sgn}(v))^{\alpha} \tag{1.4}$$

where x is the displacement, F the friction force, F_c the Coulomb friction force, σ the stiffness coefficient and α a parameter determining the shape of the stress-strain curve. This friction model exhibits the property of rate independence which indicates that the model is only position-dependent since it is only a function of the displacement and the sign of velocity [32].

Another dynamic friction model is the Bristle model introduced by Haes-

sig and Friedland [34]. This friction model is developed to capture the behavior of the microscopical contact points between two surfaces. The irregularities of surface are modeled using randomly located bristles. Each contact with bristles is thought of as a bond and as surfaces move relative to each other the strain in the bond increases causing to rise the friction force. The friction force in the Bristle model is formed as the following

$$F = \sum_{i=1}^{N} \sigma(x_i - b_i) \tag{1.5}$$

where N is the number of bristles, σ the stiffness of the bristles, x_i the relative position of the bristles, and b_i the location where the bond was formed.

The significance of this friction model is that it can capture the random nature of friction. But due to its complexity, this model is inefficient when used in simulations [32]. In order to make the bristle model more efficient, Heassig and Friedland also proposed the reset integrator model in [34]. In this model of friction the bond is kept constant by shutting off the increase of the strain at the point of rupture instead of snapping a bristle.

The Lugre model represents another dynamic friction model and is based on the same idea of the bristle model where friction force is calculated using the deflection force of elastic springs [35]. The bristles deflect like springs due to the applied tangential force. The bristle deflection depends on the velocity and is low at low velocities. The deflection decreases with increasing velocity which models the Stribeck effect. The Lugre model has the following form

$$\begin{aligned} \frac{dz}{dt} &= v - \sigma_0 \frac{|v|}{g(v)} z \\ F &= \sigma_0 z + \sigma_1(v)\frac{dz}{dt} + f(v) \end{aligned} \tag{1.6}$$

where σ_0 denotes the stiffness of the bristles, σ_1 the damping and z the average bristle deflection. The function $g(v)$ models the Stribeck effect, and $f(v)$ is the viscous friction.

Further information about other dynamic models such as the friction model by Briman and Sorine and models for lubricated contacts can be found in [36], [37] and [38].

1.4.2 Approaches to Friction Rendering

In the literature, several researchers have adapted static and dynamic friction models to haptic rendering [13, 39–42]. More specifically, in [42], GO method is extended to enable friction simulation based on the Coulomb friction model. The model captures the stick-slip characteristics of friction by defining two distinct modes of operation: sticking and sliding. Similarly, in [43], a friction cone algorithm is proposed that can accommodate the stick-slip friction model. In order to manage the position of GO, the algorithm forms a friction cone initiating at HIP and extending up to the object's surface. The circle of friction is formed as the intersection of the friction cone with the polyhedral surface. During the interaction with the virtual object, the algorithm initializes in the sticking mode, in which GO is set to and kept still at the contact location until the GO moves out of the friction circle, that is, the tangential component of the penalty force exceeds the static friction force threshold. When the GO moves out of the friction circle, a mode transition is triggered and the sliding mode initializes by repositioning the

GO at the boundary of the friction circle. In the sliding mode, the GO stays at the boundary of the friction circle and tracks the HIP as the user moves. The algorithm can also handle curved parametric surfaces indirectly, through construction of tangent planes.

An alternative friction rendering method proposed in [40] introduces snags located on the objects surface in order to emulate stiction. Sticking begins when HIP falls into a snag and continues until the user can apply a force sufficient to leave the snag. When HIP encounters a snag, it is pushed towards the center of the snag by a force tangent to the surface. When HIP is not in a snag, it moves across the surface opposed by a friction force that is proportional to the normal force. In addition to friction rendering, this technique can simulate simple surface textures by varying snag distribution on the surface. A similar approach to friction rendering is based on the bristle model [34]. In this approach, virtual bristles randomly located on the sliding surface imitate the friction by bonding to each other and breaking away [44]. Unfortunately, both of these approaches are computationally expensive for real-time implementation.

In [41] the stick-slip friction is applied to haptic rendering based on a modified version of Karnopp's friction model [31]. This approach is advantageous as it provides a remedy for the erroneous estimation of velocity at low speeds. Velocity at low speeds governs the switching behavior from sliding mode to stuck mode. In another study [45], a computational friction model is proposed that implements a variation of Dahl's friction model [33]. This continuous model of friction is time-free and exhibits four regimes: sticking, creeping, oscillating, and sliding. This model also accounts for the pre-sliding displacement in the sticking regime. Thanks to its distance based implemen-

tation, the computational friction model is robust to noise and does not drift under small bias forces unlike the original Dahl's friction model, which is prone to drift.

1.5 Structure of the Thesis

The rest of this thesis is organized as follows. Chapter 2 addresses the haptic rendering approach of parametric models and discusses the details of the feedback-stabilized closest point tracking algorithm. Simulation and real-time implementation results of a haptic rendering system based on this algorithm are presented in Chapter 2. Chapter 3 introduces the proposed friction rendering algorithm and presents simulation and real-time implementation results of the algorithm. In the last chapter, the contributions of this thesis are summarized and the future work is discussed. In this chapter, a haptic deformation modeling technique that makes use of D-NURBS geometry is provided. The second part of the future work is the human subject experiments designed to measure the effects of friction on perception which is also explained in Chapter 4.

1.6 Contributions

This thesis implements a haptic rendering algorithm based on a feedback-stabilized tracking algorithm [1] and extends this algorithm by introducing a novel friction rendering approach. The feedback-stabilized tracking and friction rendering algorithms are verified through simulations and implemented in real-time using Simulink-environment. Both algorithms possess the fol-

lowing properties:

- computational efficiency due to the relative simplicity of the algorithms
- immunity to start up errors
- robustness to disturbances
- being appropriate for haptic rendering due to fast and easy computation
- capability to treat parametric models directly and to work on polygonal models
- capability of handling surface patches and objects formed by tiling-together surface patches
- being suitable for real-time implementation

The friction rendering algorithm has the following advantages:

- it directly operates on parametric surfaces,
- it handles smooth and highly curved shapes,
- it guarantees force continuity during sharp edge crossings of virtual objects,
- it eliminates the need for extra smoothing algorithms.

Chapter 2

Haptic Rendering

This chapter introduces haptic rendering of continuous parametric surfaces. The formulations for modeling of parametric surfaces are presented and theorem of a feedback-stabilization based closest point tracking algorithm is stated. Simulation and real-time implementation results of the algorithm are represented. The haptic rendering algorithm is compared to Newton-Rhapson based haptic rendering and simulation results are shown.

2.1 Haptic Rendering of Parametric Surfaces

The haptic rendering algorithm presented in this thesis is based on a closest point (CP) tracking algorithm which is broadly discussed in Section 2.1.2. This tracking algorithm can operate on continuous parametric surfaces using a direct approach. As mentioned in Section 1.2, current haptic rendering approaches make use of parametric representations using indirect methods to calculate force response of a haptic rendering system. However, these indirect approaches necessitate a computationally expensive conversion from continuous parametric models to a large number of polygonals. Another disadvantage of indirect approaches is that they fail to approximate complex

and highly curved surfaces. As a remedy to these limitations, we developed an algorithm that requires fewer computations and can properly handle complex surfaces as it can directly operate on continuous parametric models.

For the modeling of parametric surfaces, NURBS (Non-Uniform Rational B-Spline) models are used which are the *de facto* standard throughout the CAD/CAM/CAE industry. NURBS are advantageous in terms of higher order continuity, compact representation and exact computation of surface normals. The definitions of a NURBS curve and surface and important properties of NURBS models are given in the next section. The remaining sections of this chapter discuss a comparison of our haptic rendering algorithm with a Newton-Rhapson based approach and boundary handling of parametric models.

2.1.1 Modeling of Parametric Curves and Surfaces

NURBS modeling is a method of representation and design of complex shapes that is mostly used in design industry. NURBS offer one common mathematical formulation for both analytical and free form shapes. They are invariant under affine and perspective transformations. Fast evaluation of NURBS can be carried out by numerically stable and accurate algorithms. Various complex shapes can be modeled using NURBS geometry. The definitions of a B-spline and NURBS curve are given in the following as explained in greater detail in [46].

Given a set of $n+1$ control points P_0, P_1, ..., P_n, a Bezier curve is defined as

$$C(u) = \sum_{i=0}^{n} P_i B_{i,n}(u) \tag{2.1}$$

where $B_{i,n}(u)$ is a Bernstein polynomial and $u \in [0, 1]$. A B-spline is a

generalization of the Bezier curve and is defined using a knot vector with a nondecreasing sequence $U = u_0, u_1, .., u_m$ with $u_i \in [0, 1]$ and control points $P_0, ..., P_n$. The curve representation of a B-spline curve is of the following form:

$$C(u) = \sum_{i=0}^{n} P_i N_{i,p}(u) \tag{2.2}$$

where $p = m - n - 1$ and the i^{th} B-spline basis function of p-degree (order p+1), denoted by $N_{i,p}(u)$, is defined as

$$N_{i,0}(u) = \begin{cases} 1 \text{ if } u_i \leq u < u_{i+1} \\ 0 \text{ otherwise} \end{cases} \tag{2.3}$$

$$N_{i,p} = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

After introducing the basis functions of a B-spline, a NURBS curve of p^{th}-degree is defined by

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \quad a \leq u \leq b \tag{2.4}$$

where the P_i are the control points, the w_i are the weights and $N_{i,p}(u)$ are the p^{th}-degree B-spline basis functions defined on the nonperiodic knot vector

$$U = \left\{ \underbrace{a, .., a}_{p+1}, u_{p+1}, .., u_{m-p-1}, \underbrace{b, .., b}_{p+1} \right\}$$

The limits of the parameter u are assumed to have the values of $a = 0$ and $b = 1$ and $w_i > 0$ for all i.

Similarly a NURBS surface of degree p in the u direction and of degree q in the v direction is given by

$$S(u,v) = \frac{\sum_{i=0}^{n} \sum j = 0^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j} P_{i,j}}{\sum_{i=0}^{n} \sum j = 0^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j}} \tag{2.5}$$

where $P_{i,j}$ form the bidirectional control net, $w_{i,j}$ are the weights, and $N_{i,p}(u)$ and $N_{j,q}(v)$ are the nonrational B-spline basis functions defined on the knot vectors

$$U = \left\{ \underbrace{0,..,0}_{p+1}, u_{p+1}, .., u_{r-p-1}, \underbrace{1,..,1}_{p+1} \right\}$$

$$V = \left\{ \underbrace{0,..,0}_{q+1}, v_{q+1}, .., u_{s-q-1}, \underbrace{1,..,1}_{q+1} \right\}$$

where $r = n + p + 1$ and $s = m + q + 1$

The shape of a NURBS curve is represented by its control points P_i and a point on the curve is computed by taking a weighted sum of these control points. NURBS models have the feature of local support which allows a single control point to influence only those intervals where it is active. This property indicates that a part of a NURBS model can be modified without having to change the other parts of the geometry. More control on a NURBS curve can be gained by adding more control points and changing the weights which provides a better approximation to the curve. The knot vector of a NURBS curve determines where and how the control points affect the NURBS curve.

2.1.2 Feedback-Stabilized Closest Point Tracking Algorithm for Convex Parametric Surfaces

In penalty-based haptic rendering algorithms, the restoring forces fed back to the user are calculated proportional to the amount of penetration of HIP

into the surface. Hence, the ability to locate the relevant closest point on the interaction surface at interactive rates is a crucial element of such haptic rendering algorithms. In this section, we review the closest point tracking algorithm introduced in [19, 21], which operates on parametric surface models in a direct manner.

The core idea and implementation of the closest algorithm is demonstrated on a smooth curve depicted in Figure 2.1.1. Let u be a parameter and $\overrightarrow{f}(u)$ represent the parametric curve. At any parameter value u, the curve $\overrightarrow{f}(u)$ maps to a point in space. Let $\overrightarrow{\hat{f}_u}(u)$ denote the unit tangent vector at u. Then, the unit surface normal $\overrightarrow{\hat{n}}(u)$ at u can be calculated by taking the cross product of the unit tangent vector with the unit vector $\overrightarrow{\hat{k}}$, which points out of the plane.

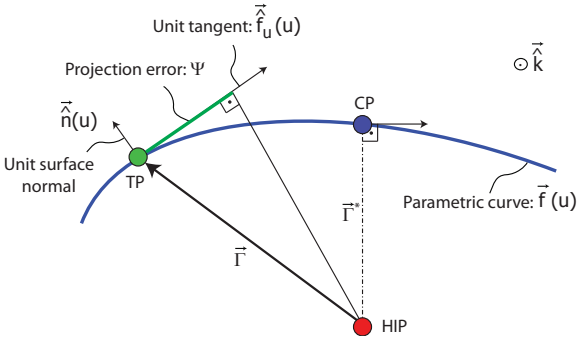

Figure 2.1.1: Feedback-stabilized closest point tracking algorithm

The algorithm is founded on the fact that the line connecting HIP to CP needs to be perpendicular to the surface tangent at CP. This is achieved utilizing a feedback controller that checks for this condition at each instant of time and continually updates the location of a test point (TP) until it converges to CP. In particular, given a TP on the parametric curve, the algorithm forms the vector $\overrightarrow{\Gamma}$ initiating at HIP and extending to TP. Then,

the *projection error* Ψ is calculated by projecting the $\overrightarrow{\Gamma}$ vector on the unit tangent vector evaluated at TP. The controller updates the location of TP by determining its speed of movement along the curve, that is, by setting $\dot{u}_{TP} = -K\,\Psi$, where K is the positive controller gain.

In [21], the algorithm has been extended to surface to surface tracking and its uniformly asymptotical stability is proven for proper selection of the gain K. Uniform asymptotical stability of the algorithm implies robustness against numerical noise and more importantly against initialization errors on the interactive surface. Under some mild assumptions on convexity, the global convergence of the algorithm can also be guaranteed, rendering any initialization on the parametric surface as an acceptable one. Requiring a few function evaluations, simple vector operations and an integrator, the algorithm is computationally efficient and easy to implement.

Let the magnitude of the velocity of HIP ${}^{N}\overrightarrow{v}_{HIP}$ be bounded by ${}^{N}\overline{v}_{HIP}$. Also let the workspace be bounded, ensuring that the magnitude of $\overrightarrow{\Gamma}$ is bounded by a positive constant $\overline{\Gamma}$.

Theorem 2.1.1 *If the image of the mapping* $\overrightarrow{f}(u) : [0,1] \rightarrow \Re^2$ *defines a convex parametric curve* C*,* HIP *and* C *are in continuous motion with respect to one another; given a positive convergence parameter* ϵ *and a controller gain that satisfies* $K > \frac{2\,\overline{\Gamma}\,{}^{N}\overline{v}_{HIP}}{\epsilon^2}$*, then the controller* $\dot{u} = -K\Psi$ *renders the minimum distance point* CP^* *uniformly practically asymptotically stable over the curve* C*.*

Sketch of the Proof The proof is based on a control Lyapunov function which is defined as

$$V = \frac{1}{2}\left(\overrightarrow{\Gamma}^2 - \overrightarrow{\Gamma}^{*2}\right) \tag{2.6}$$

where $\overrightarrow{\Gamma} = \overrightarrow{f}(u) - \overrightarrow{x}_{HIP}$ and Γ^* is the Γ evaluated at the closest point. The control Lyapunov function V is continuous, positive definite and decrescent (see [1]). To prove uniform practical asymptotic stability of the algorithm, the negative definiteness of the time derivative of the control Lyapunov function is to be shown. The time derivative of V is given by

$$\begin{aligned}\dot{V} &= \overrightarrow{\Gamma} \cdot \frac{^N d}{dt}\overrightarrow{\Gamma} - \overrightarrow{\Gamma}^* \cdot \frac{^N d}{dt}\overrightarrow{\Gamma}^* \\ &= \Gamma \cdot (\overrightarrow{f}_u(u)\,\dot{u} - {}^N\overrightarrow{v}_{HIP}) - \Gamma^* \cdot (\overrightarrow{f}^*_u(u)\dot{u}^* - {}^N\overrightarrow{v}_{HIP}) \\ &= (\Gamma \cdot \overrightarrow{f}_u(u))\,\dot{u} - (\Gamma - \Gamma^*) \cdot {}^N\overrightarrow{v}_{HIP}\end{aligned} \tag{2.7}$$

where N is the fixed world reference frame. Next, define the projection error as $\Psi = \overrightarrow{\Gamma} \cdot \overrightarrow{f}_u(u)$. Now if the control law $\dot{u}$ is selected as $\dot{u} = -K\,\Psi$, and substituted into equation 2.7, then

$$\dot{V} = -K\,\Psi^2 - \left(\overrightarrow{\Gamma} - \overrightarrow{\Gamma}^*\right) \cdot {}^N\overrightarrow{v}_{HIP} \tag{2.8}$$

Given a positive number ε sufficiently larger than zero, if $\Psi > \varepsilon$, then the term $-K\Psi^2$ dominates $|(\overrightarrow{\Gamma} - \overrightarrow{\Gamma}^*) \cdot {}^N v_{HIP}|$ and renders $\dot{V}$ negative if $K > \frac{2\,\overline{\Gamma}\,{}^N\overline{v}_{HIP}}{\epsilon^2}$ as explained in the following.

$$|(\overrightarrow{\Gamma} - \overrightarrow{\Gamma}^*) \cdot {}^N v_{HIP}| < |\overrightarrow{\Gamma} - \overrightarrow{\Gamma}^*|\,|{}^N v_{HIP}| \tag{2.9}$$

$$|\overrightarrow{\Gamma}^*| < |\overrightarrow{\Gamma}| \Rightarrow |\overrightarrow{\Gamma} - \overrightarrow{\Gamma}^*| < 2\,\overrightarrow{\Gamma} < 2\overline{\Gamma} \tag{2.10}$$

$$|{}^N\overrightarrow{v}_{HIP}| < {}^N\overline{v}_{HIP} \tag{2.11}$$

If we substitute the upper boundaries of ${}^N\overrightarrow{v}_{HIP}$ and $(\overrightarrow{\Gamma} - \overrightarrow{\Gamma}^*)$ shown in inequalities 2.10 and 2.11, into the right hand side of the inequality (2.9), we can show that

$$|(\overrightarrow{\Gamma} - \overrightarrow{\Gamma}^*) \cdot {}^N v_{HIP}| < 2\,\overline{\Gamma}\,{}^N\overline{v}_{HIP} \tag{2.12}$$

$$-K\Psi^2 + |(\overrightarrow{\Gamma} - \overrightarrow{\Gamma}^*) \cdot {}^N\overrightarrow{v}_{HIP}| < 0 \tag{2.13}$$

$$\tag{2.14}$$

can be assured if the following inequality holds

$$K > \frac{2\,\overline{\Gamma}^{\,N}\overline{v}_{HIP}}{\Gamma^2} \tag{2.15}$$

And since $\Psi > \varepsilon$,

$$K > \frac{2\,\overline{\Gamma}^{\,N}\overline{v}_{HIP}}{\varepsilon^2} \tag{2.16}$$

Rendering $\dot{V}$ negative definite for a given ε assures uniform practical stability. However, $\dot{V}$ can be shown to be negative definite by bounding it with a negative definite function of the state ζ as done in [1], which assures uniform practical asymptotic stability. The uniform asymptotic stability of the algorithm can be shown by analyzing the case $\Psi < \varepsilon$, and adding a feed forward compensation term when in the ε ball [1].

2.1.3 Comparison of the Haptic Rendering Algorithm to a Newton-Rhapson based Approach

Figure 2.1.1 presents typical trajectories during closest point tracking of the feedback-stabilized algorithm [21] and one of the commonly employed methods based on intermediate representations [47]. The trajectories shown in Figures 2.1.1(a) and (b) are achieved by letting HIP approach to the parametric curves by following a straight line. The velocity of HIP is kept constant during its motion along the sloped line. In Figure 2.1.1, the colored straight lines between the HIPs trajectory and the curves represent the minimum distances between HIPs current position and the curve. The pink lines in Figure 2.1.1(a) which are normal to the curve indicate that feedback-stabilized algorithm guarantee convergence to CP independent of the amount of curvature involved while the blue lines that are not perpendicular to the curve in Figure 2.1.1(b) show that methods based on discretizations/intermediate representations may not converge around high curvature regions of the curve. As a

consequence, the feedback-stabilized algorithm strongly contributes to realistic haptic interaction even with the highly curved objects by eliminating discontinuous force artifacts.

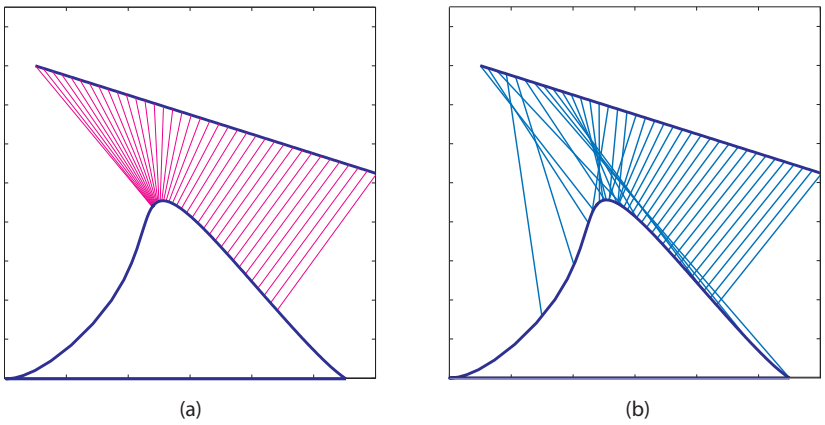

Figure 2.1.2: Typical trajectories during closest point tracking of (a) the feedback-stabilized algorithm (b) the methods based on intermediate representation

2.1.4 Boundary Handling using Voronoi Diagrams

Operating on the projection errors, the controller can update location of CP to continually track HIP on an interaction surface. However, often parametric surfaces are formed by tiling several surface patches together. In such a case, the determination of the relevant surface patch that represents the interaction surface is required. To handle such cases, the control based closest point tracking algorithm has been extended to handle tiling several surface patches [21]. To determine the relevant surface patch, a switching controller is used, where the mode switches are triggered and the proper state initializations are handled via a finite state machine. The finite state machine

governing the discrete behavior of the controller is formed offline based on the Voronoi regions of the object's features and their connectivity[1]. The finite state machine performs a feature-based tracking similar to the algorithms derived for polyhedral collision detection [9] and [10].

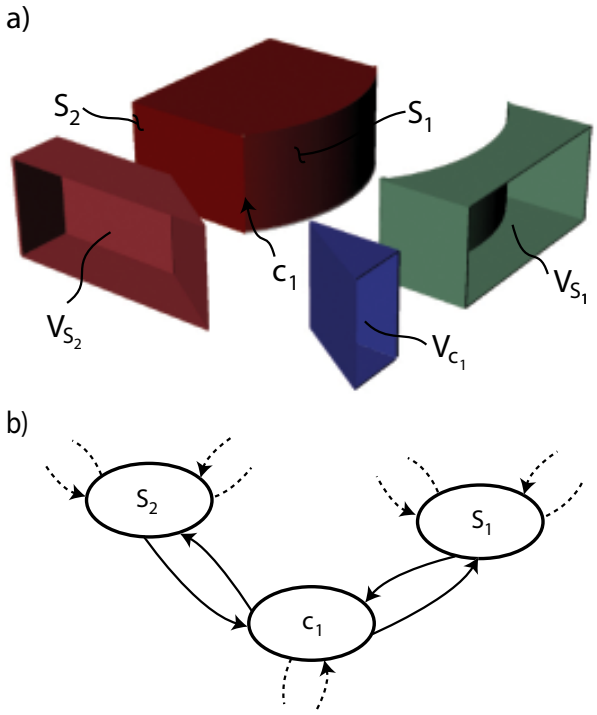

Figure 2.1.3: Boundary handling using Voronoi regions [1]

2.1.5 Simulation Results of Haptic Rendering

The visualization of virtual objects used in the simulation and real-time implementation are carried out using the NURBS SIntef Spline Library (SISL),

[1]Even though the determination of the Voronoi regions of a complex object can be computationally expensive, the real-time performance of the algorithm is not affected since this computation is performed offline.

developed and maintained by the geometry modeling group at SINTEF ICT, Department of Applied Mathematics [48]. The SISL code is appropriate to be used in real-time implementations of haptic rendering algorithms since it is written in C and supports curve/surface creation and manipulation using NURBS geometry while maintaining a high degree of accuracy in numerical calculations.

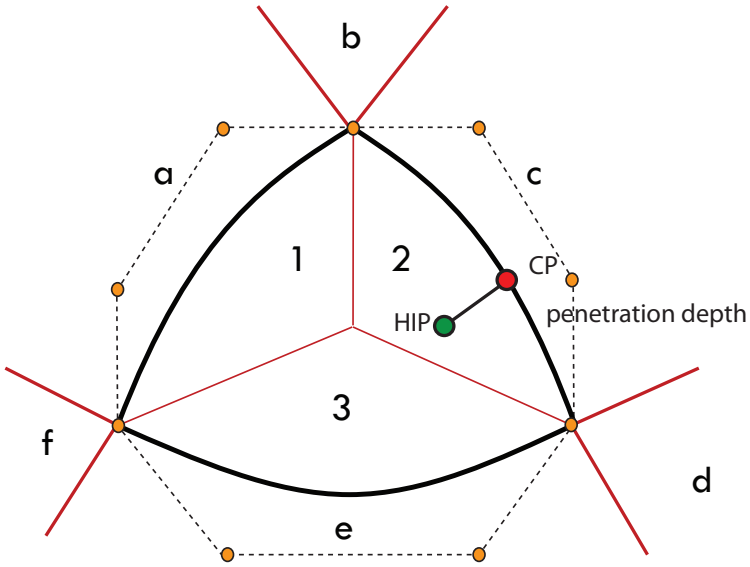

Figure 2.1.4: Internal and external Voronoi regions of a virtual object

Figure 2.1.4 depicts a virtual object that is created by joining three NURBS curves. Each NURBS curve is created using four control points and is represented with orange dots. On the corners of the virtual object, two control points coincide that belong to two different adjacent curves. The virtual object is divided into three internal Voronoi regions whereas the outer space of the environment are tiled into six external Voronoi regions that are denoted by $a - f$. Before a collision with the virtual object, external Voronoi

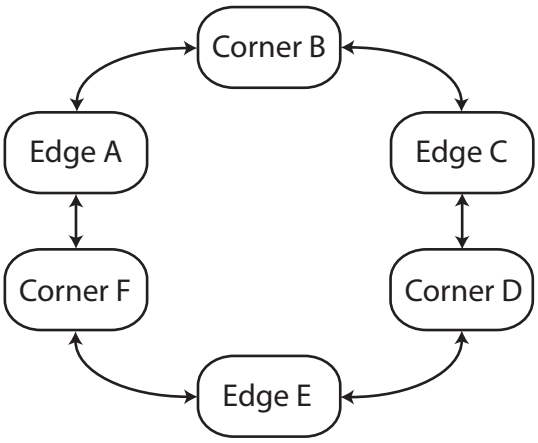

Figure 2.1.5: State flow of Voronoi regions of the virtual object seen in 2.1.4. The corresponding Voronoi regions are shown with same letters as in Figure 2.1.4

regions help to determine the relevant curve to initialize the CP so that HIP can be tracked faster and more efficiently by CP. The same idea holds for the internal Voronoi regions that are used after a collision with the virtual object. The boundaries of the external Voronoi regions shown in Figure 2.1.4 are determined by using the normals of the curves at the end points. Figure 2.1.5 shows the state machine that is used for the determination of active Voronoi regions of the virtual object seen in Figure 2.1.4.

The haptic rendering system shown in Figure 2.1.4 demonstrates the positions of HIP and CP after a collision of the virtual object. The minimum distance between HIP and CP is found using the feedback-stabilized tracking algorithm. As mentioned in Section 1.2, the restoring force after the collision with the virtual object is calculated by multiplicating the penetration depth with the stiffness of the virtual object as following

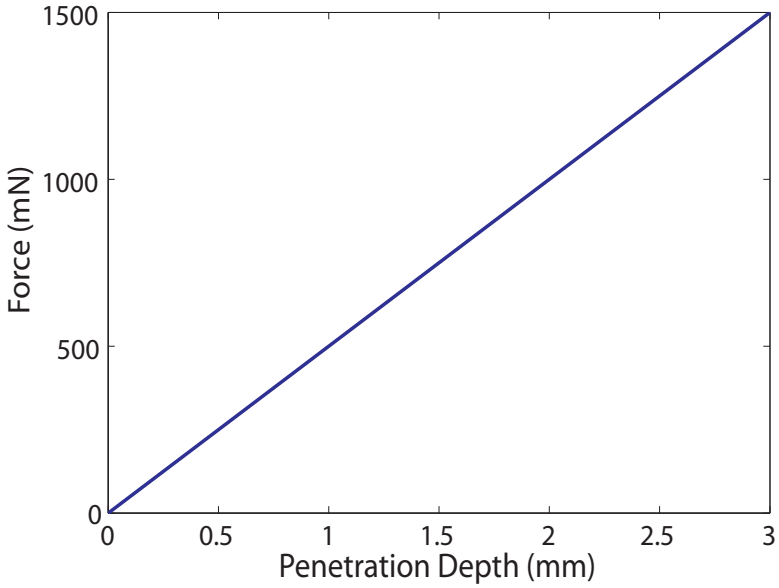

Figure 2.1.6: Force response graph of a haptic rendering system

$$F = k\Delta x \tag{2.17}$$

where k is the stiffness of the virtual object and Δx the penetration depth.

The force response arising from the interaction between HIP and the virtual object is shown in Figure 2.1.6. As HIP penetrates deeper into the virtual object, the restoring force increases proportional to the penetration depth.

2.1.6 Real-Time Implementation of Haptic Rendering

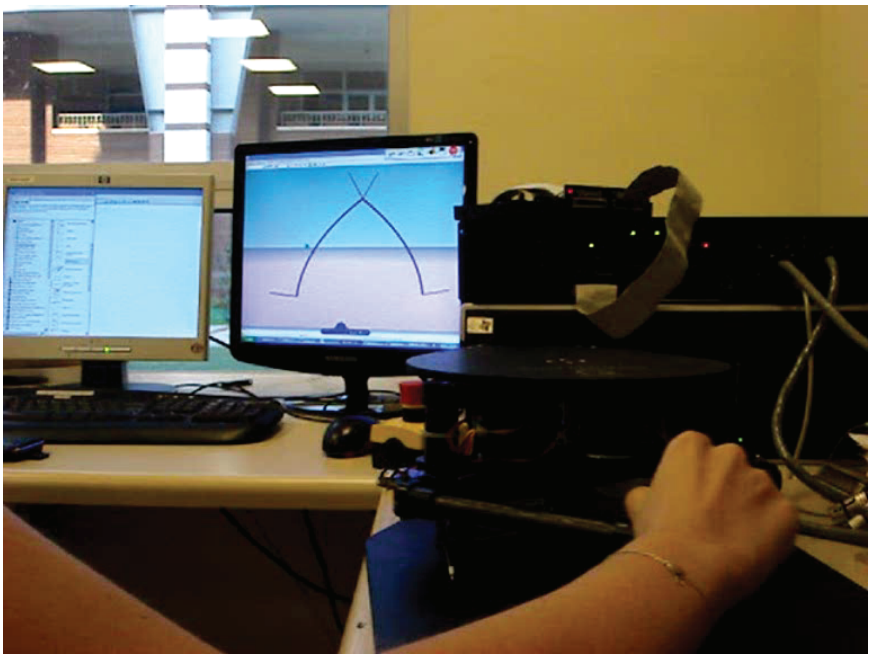

Figure 2.1.7: Measured force in real-time

Figure 2.1.7 depicts the haptic rendering system used in the real-time implementations where the user interacts with the virtual object through the probe of a 2-DoF Pantograph. The haptic update rate is set to be 1kHz

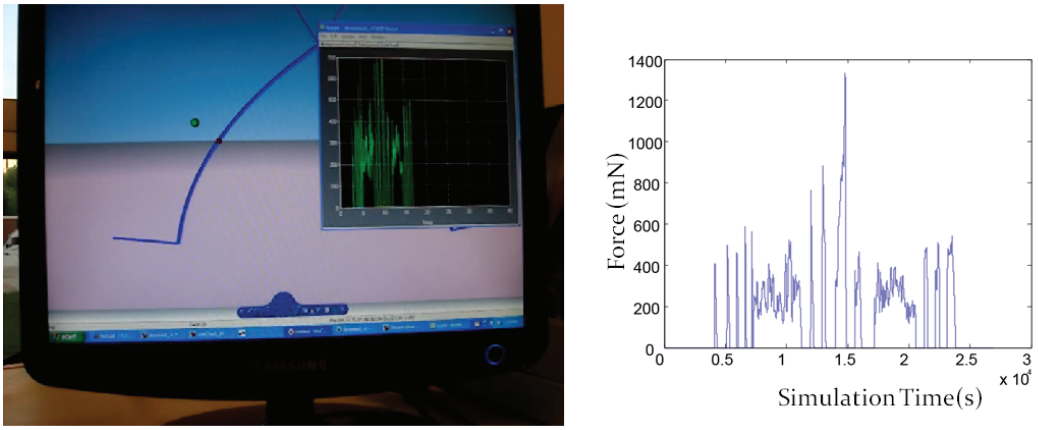

Figure 2.1.8: The left hand side of the figure represents the virtual object that the user interacts with and the computed forces are shown in the right hand side of the figure.

and the stiffness of the virtual object is selected as 100 N/m. Throughout the implementation the user interacts with the virtual object intermittently. The zero value of the force graph represented in Figure 2.1.8 depicts the instants when the user does not penetrate into the virtual object. The increasing values of the graph represent the user's motion while penetrating deeper into the virtual object and the decreasing intervals indicate that the user ends the interaction with the virtual object.

Chapter 3

Friction Rendering

In this chapter, the details of the proposed friction rendering algorithm are explained and discussed. The working principle of stick-slip friction rendering algorithm is illustrated with sketches. Simulations of friction rendering in 2- and 3-D and real-time implementation results verifying the mode changes from static to dynamic friction are represented.

3.1 Friction Rendering Algorithm

The friction rendering algorithm introduced in this thesis provides a direct method to handle parametric surface models. Unlike the existing friction rendering algorithms in the literature, it does not rely on the algorithms introduced for polyhedral surfaces. The algorithm is inherently stable (uniform asymptotic stability) and can handle surfaces with high curvature. Furthermore, this algorithm allows transitions from sticking to sliding and sliding to sticking, as well as surface to surface transitions, without introducing discontinuous force artifacts, which are easily perceivable by and distractive to users. The algorithm is computationally efficient, simple to implement, and possesses an intuitive physical interpretation. The algorithm enables easily

tuning of the change in friction coefficient during the mode transitions to enable rendering of the Stribeck effect. Moreover, thanks to its feedback-stabilized core, the algorithm is robust against drift and numerical noise.

3.1.1 Stick Slip Friction and Stribeck Effect

The implementation of stick-slip friction is based on a variation of the stiction model seen in Figure 3.1.1(a) that makes use of static and dynamic coefficients of friction to compute the friction force. The stiction model is varied such that the reduction from the static coefficient of friction to the dynamic coefficient of friction is modulated during the transition from sticking to sliding to simulate the velocity dependent Stribeck effect as depicted in 3.1.1(b). In particular, the effect is simulated by interpolating between the values of the static and dynamic coefficients of friction based on the time rate of change of GO position during the mode transition.

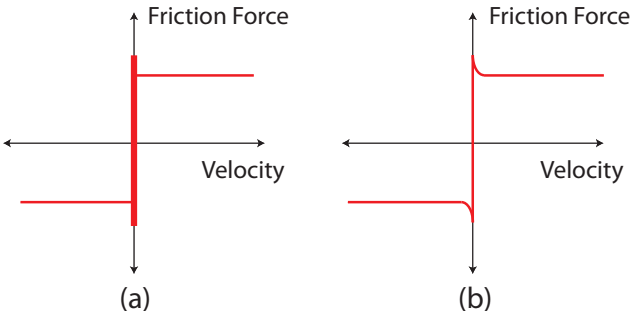

Figure 3.1.1: a) Stiction model of friction, b) Stiction model of friction with Stribeck effect

The friction rendering algorithm consists of three distinct modes of operation: Stuck, slip, and transition from stuck to slip. The switchings between these modes with proper initializations are governed by a finite state machine

as shown in Figure 3.1.2. The mode transitions take place as follows. The algorithm is initialized in the stuck mode and stays in this mode until the breakaway force threshold is exceeded. If CP stops or changes the direction of its movement during the stuck mode, a self transition is triggered that resets the location of GO to CP. Once the breakaway friction force threshold is exceeded, the algorithm goes into the mode that handles the transition from stuck to slip. This mode is active until GO converges to CP. As soon as GO and CP become collocated, another mode switching occurs and the algorithm enters the slip mode. During the slip mode if CP stops or changes the direction of movement, then the algorithm is set back to the stuck mode. Otherwise, a transition out of the stuck mode is triggered only when the applied normal force exceeds the dynamic friction force threshold dictated by the stiction model of friction. In this case, the algorithm enters the stuck mode and GO is reset to a location that ensures force continuity.

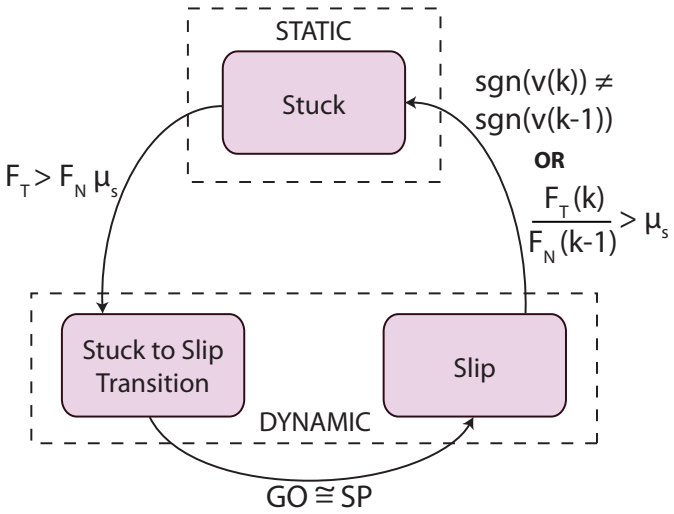

Figure 3.1.2: Finite state machine of the friction rendering algorithm

Figures 3.1.3(a)–(e) present schematic representations of several critical stages of the algorithm, demonstrating the conditions governing state transitions and the formulas used to calculate the force $\overrightarrow{F}_{forcefb}$ to be fed back to the user. At any instant of time, this force is applied at a point called the point of application (PA). At PA, the force can be decomposed into two vector components: the friction component $\overrightarrow{F}_T$ and the normal component $\overrightarrow{F}_N$. Hence, the total force applied to the user at PA is given as the vector sum of these two forces: $\overrightarrow{F}_{forcefb} = \overrightarrow{F}_T + \overrightarrow{F}_N$.

In Figure 3.1.3(a), the algorithm initializes in the stuck state and HIP is located in the virtual object after the collision. In this mode, GO is set to CP and kept still while HIP continues its motion, until the breakaway friction force threshold is exceeded, that is, the magnitude of tangential force exceeds the magnitude of the normal force times static coefficient of friction. This is the case shown in Figure 3.1.3(b).

Once the tangential force calculated in the stuck mode exceeds the breakaway friction force threshold, the transition from stuck to slip becomes active. In this transition mode, the feedback-stabilized closest point tracking algorithm is initialized with GO location. As depicted in Figure 3.1.3(c), the feedback controller ensures that GO quickly converges to CP. While the GO is converging towards the CP, the friction coefficient is continuously modulated with respect to the projection error. In particular, the friction coefficient is adjusted such that it is equal to the static friction coefficient at the first time step and to the dynamic friction coefficient when GO converges to CP. Note that the projection error is inversely proportional to tangential speed of GO; therefore, the modulation of the friction coefficient renders velocity dependent Stribeck effect.

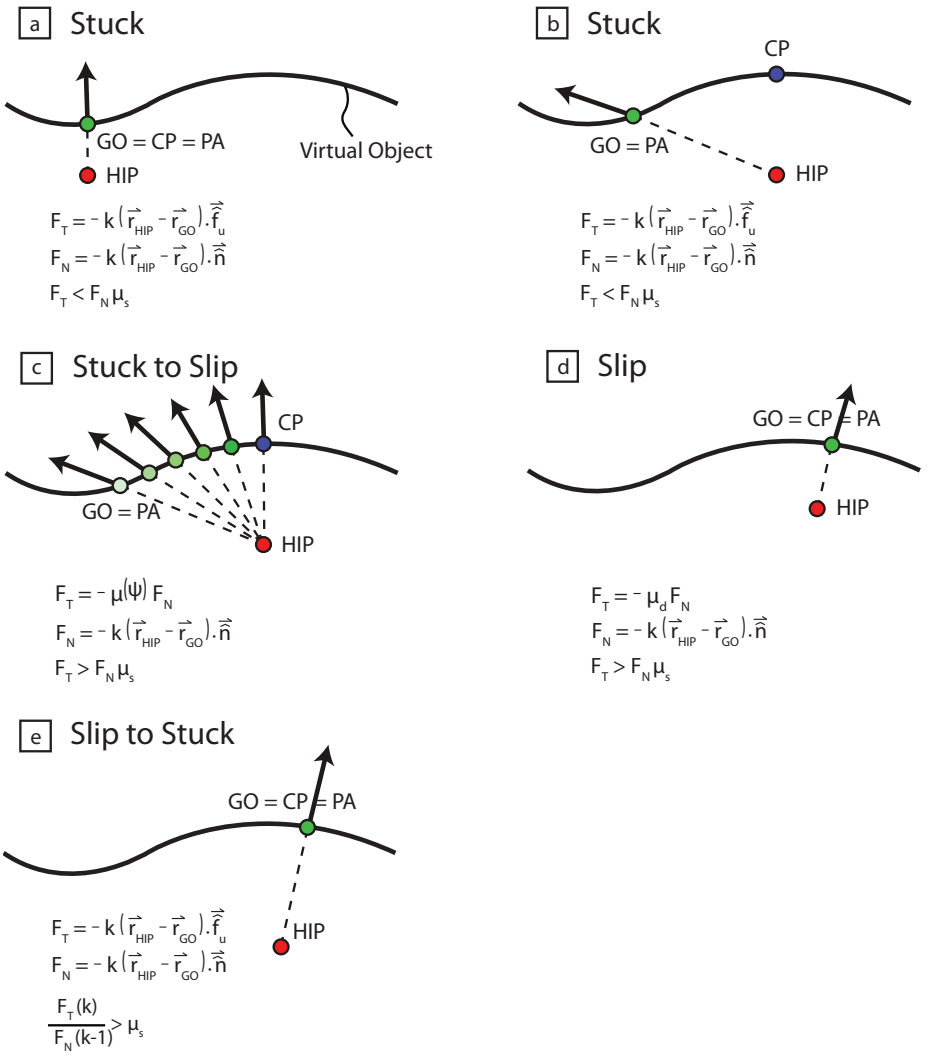

Figure 3.1.3: Schematic representations demonstrating critical stages of the friction rendering algorithm

Slip mode initiates when GO converges to CP as in Figure 3.1.3(d). In the slip mode, since the friction force is directly proportional to the normal force, GO is set as CP and tracks the motions of HIP on the surface. A transition out of this mode to the stuck mode can be triggered if CP stops or changes direction of its motion. Another transition is possible if a sudden, sufficiently large increase is observed in the normal force. The check on normal force is performed by comparing the coefficient of static friction to the ratio of the tangential force at the previous state to the normal force at the current state.

In Figure 3.1.3(e), the instant when the slip to stuck transition takes place is depicted. This transition is triggered when the ratio of the tangential force at the previous state to the normal force at the current state exceeds the coefficient of static friction. When such a mode switch is triggered, GO and CP are collocated causing the tangential force being set to zero which would be incorrect. Therefore, after this mode switch the tangential force is calculated such that the friction force at previous state is added to the value of the tangential force at the current state. Thus, the tangential force is kept continuous after this mode switch and prevented from being set to 0 from where it will begin to grow as HIP moves in the tangential direction. Following the transition from slip to stuck, except the value added to the tangential force, GO behaves as depicted in Figure 3.1.3(c). The Pseudo code of the friction rendering algorithm is presented in Algorithm 1.

Algorithm 1 Friction Rendering Algorithm.

Require: A connectivity graph ($\mathbb{V}$,$\mathbb{E}$) of Voronoi Regions, a set of surfaces $\mathcal{S}$, static μ_s and dynamic μ_d coefficients of friction, surface stiffness K, convergence parameter ϵ.

Ensure: Active feature $\mathcal{F}[k]$, curvilinear coordinates of PA[k], the magnitudes of friction force $F_f[k]$ and normal force $F_N[k]$.

$\mathbf{r}_{HIP}[1], \mathbf{v}_{HIP}[1] \leftarrow$ get position and velocity of HIP from the haptic interface;

$(\mathcal{V}[1], \mathcal{F}[1]) \leftarrow$ determine which Voronoi region HIP is in and the associated surface feature;

$\mathbf{r}_{GO}[0] \leftarrow$ initialize GO on the active feature $\mathcal{F}[1]$;

state = STATIC;

Voronoi_Switch_Flag = True;

Stuck_Transition_Flag = True;

Force_Continuity_Flag = False;

$k = 1$;

$\mathbf{r}_{GO}[k]$, *collision* $\leftarrow$ track the closest point with respect to $\mathcal{F}[k]$, $\mathbf{r}_{GO}[k-1]$, and $\mathbf{r}_{HIP}[k]$

while True **do**

  if $\neg collision$ **then**

    state = STATIC;

    Stuck_Transition_Flag = True;

    Force_Continuity_Flag = False;

    $F_N[k] = 0$;

    $F_f[k] = 0$;

    $\mathbf{u}_{PA} = Null$

  else

    if state = STATIC & Stuck_Transition_Flag **then**

$\mathbf{r}_{stuck} = \mathbf{r}_{GO}[k]$;

Stuck_Transition_Flag = False

end if

if state = STATIC **then**

$\mathbf{r}_{GO}[k] = \mathbf{r}_{stuck}$

end if

$F[k] = K \ \|(\mathbf{r}_{GO}[k] - \mathbf{r}_{HIP}[k])\|$;

$\mathbf{u}_{PA}[k] \ \leftarrow$ determine curvilinear coordinates of $\mathbf{r}_{GO}[k]$;

$F_T[k] \ \leftarrow$ project $F[k]$ onto unit tangent at PA;

$F_N[k] \ \leftarrow$ project $F[k]$ onto unit normal at PA

case state **of**

STATIC:

Voronoi_Switch_Flag = False

if $F_T[k] < \mu_s \ F_N[k]$ **then**

if Force_Continuity_Flag = True **then**

$F_f[k] = F_{offset} + F_T[k]$

else

$F_f[k] = F_T[k]$

end if

else

State = DYNAMIC;

$F_{offset} = F_f[k]$;

Voronoi_Switch_Flag = True;

Stuck_Mode = False

end if

DYNAMIC:

if $sign(\mathbf{v}_{HIP}[k]) \neq sign(\mathbf{v}_{HIP}[k-1])$ **then**

state = STATIC;

Stuck_Transition_Flag = True;

Force_Continuity_Flag = False

else if $\mu_d \; F_N[k] > \mu_s \; F_N[k-1]$

state = STATIC;

Stuck_Transition_Flag = True;

Force_Continuity_Flag = True;

$F_{offset} = F_f[k]$

else

$\Psi \leftarrow$ calculate the projection error with respect to $\mathcal{F}[k]$, $\mathbf{r}_{GO}[k]$, and $\mathbf{r}_{HIP}[k]$

if $\Psi > \epsilon$ **then**

$\mu[k] \leftarrow$ modulate friction coefficient based on Ψ to imitate Stribeck effect

else

$\mu[k] = \mu_d$

end if

if Force_Continuity_Flag = True **then**

$F_f[k] = F_{offset} + \mu[k] \; F_N[k]$

else

$F_f[k] = \mu[k] \; F_N[k]$

end if

end if

end case

k = k+1;

$\mathbf{x}_{HIP}[k], \mathbf{v}_{HIP}[k] \leftarrow$ get position and velocity of HIP from the haptic interface;

if $\mathbf{x}_{HIP}[k-1] \in \mathcal{V}[k-1]$ and $\mathbf{x}_{HIP}[k] \notin \mathcal{V}[k-1]$ & Voronoi_Switch_Flag **then**

$(\mathcal{V}[k],\ \mathcal{F}[k])$ $\leftarrow$ track the relevant Voronoi region and the associated surface feature

else

$(\mathcal{V}[k],\ \mathcal{F}[k]) = (\mathcal{V}[k-1],\ \mathcal{F}[k-1])$

end if

$\mathbf{r}_{GO}[k]$, *collision* $\leftarrow$ track the closest point with respect to $\mathcal{F}[k]$, $\mathbf{r}_{GO}[k-1]$, and $\mathbf{r}_{HIP}[k]$

end if

end while

3.1.2 Static Friction at a Corner

The friction rendering algorithm can operate on objects that consist of one or more parametric surfaces. In case of multiple surfaces, the switchings among the surfaces must be handled carefully. When the friction rendering algorithm is in the slip mode, then the transition from one interaction surface to another is dictated by the Voronoi regions of the surfaces and all the calculations are performed with respect to the interaction surface, in the Voronoi region of which HIP lies. However, these transitions are handled differently when the friction rendering algorithm is in the stuck mode. In particular, if the algorithm is in the stuck mode and HIP moves from one Voronoi region to another, the Voronoi region based switching of interaction surface is deactivated until the breakaway force is overcome. Once the breakaway force threshold is exceeded, GO is released on the updated interaction surface to converge to CP. Deactivation of Voronoi switching is necessary, since, otherwise, the algorithm would transition to the slip mode before the breakaway force threshold is overcome.

Static friction at the corner is of importance when a user holds an object close to its corner. Such a case is illustrated in Figure 3.1.4 with an object constructed using three parametric curves. In the figure, the internal Voronoi regions of the objects are marked as 1–3.

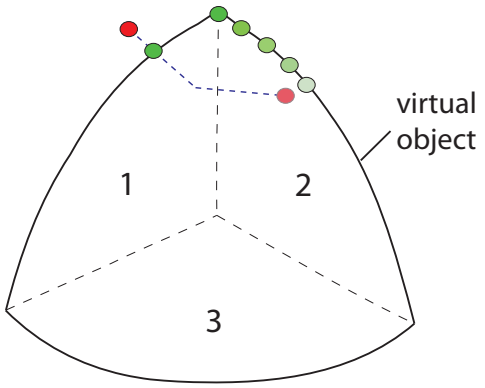

Figure 3.1.4: Static friction at a corner

3.2 Simulation Results

3.2.1 Friction Rendering in 2-D

The virtual object used in the experiments consists of two NURBS curves that merge at a vertex as shown in Figure 3.2.1(a). The straight lines at both ends of the NURBS curves represent the Voronoi regions associated with these curves and the vertex. Since the curves are convex, each straight line is perpendicular to the corresponding end of the curve to form the Voronoi regions.

The virtual object is assumed to have a stiffness of 1000 N/m and the static and dynamic coefficients of friction are set as 0.7 and 0.5 respectively. The left and right curves of the virtual object in Figure 3.2.1(a) are referred to

as the first and the second surfaces respectively. When the simulation starts, HIP is positioned outside the virtual object and within the Voronoi region of the first surface (see Figure 3.2.1(b)). During the simulation, HIP tracks a pre-determined path as depicted in Figure 3.2.1(a). The pre-determined path is set such that HIP penetrates the first surface in Figure 3.2.1(a) until a certain depth is reached, and does so in the direction that is perpendicular to the interaction surface (see Figures 3.2.1(b)–(c)). Then, in Figures 3.2.1(c)–(e) HIP moves parallel to the surface towards the vertex, while keeping its penetration depth constant. In Figure 3.2.1(f), HIP penetrates deeper into the virtual object along the surface normal to demonstrate a transition from sliding to stuck mode. Next, as seen in Figures 3.2.1(g)–(h) HIP moves parallel to the surface until sliding is achieved and then it moves out of the virtual object (see Figure 3.2.1(i)). As seen in Figure 3.2.1(i), HIP collides with the object for the second time at a location close to the corner. It penetrates the object along the surface normal and then approaches to the second curve along the horizontal direction (see Figures 3.2.1(j)–(l)). The second collision of HIP with the object aims to demonstrate static friction at corner whereas the rest of the pre-determined path aims to illustrate the stick-slip behavior of the friction rendering algorithm.

Figure 3.2.2 represents the frictional and cumulative feedback forces calculated during the execution of the pre-determined path in Figure 3.2.1(a). A linear increase in the cumulative force is observed until the marker 1, as HIP penetrates deeper into the virtual object (see Figures 3.2.1(b)–(c)). During the $1 \rightarrow 2$ interval, the algorithm is in the stuck mode and the friction force begins to increase as the user tries to apply a tangential force to overcome the breakaway friction force threshold (Figures 3.2.1(d)).

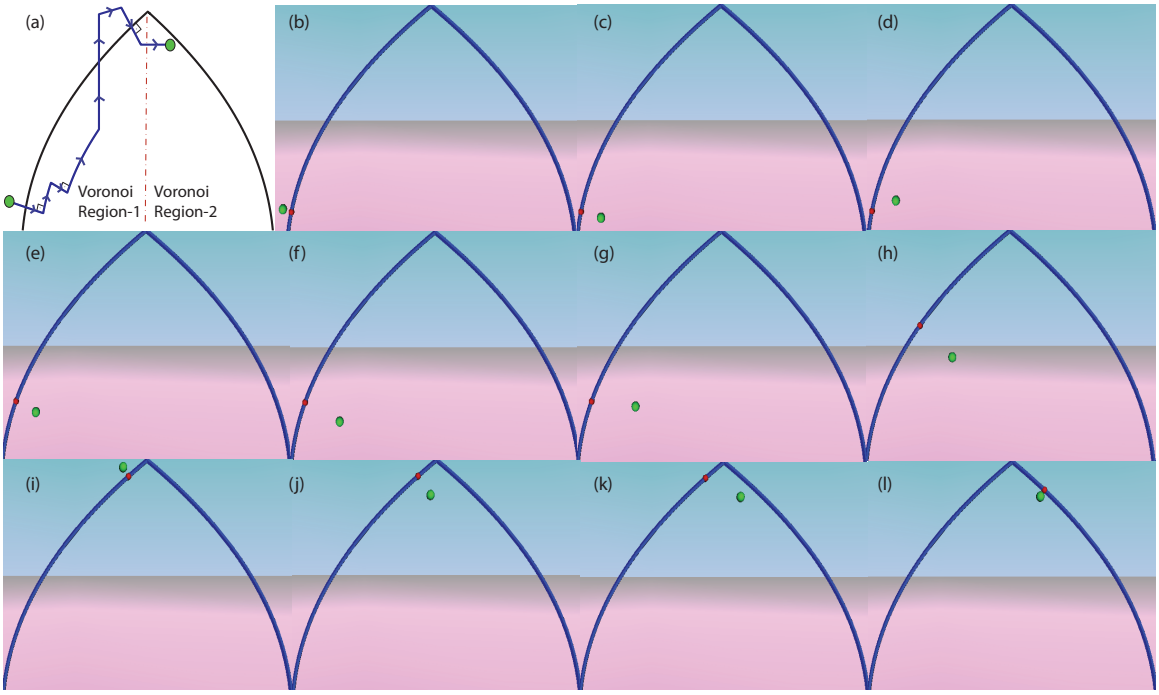

Figure 3.2.1: Snapshots depicting the location of HIP and GO during the simulation, in which HIP tracks a pre-determined path.

At the marker 2, the friction force in the stuck mode rises up to the breakaway friction threshold and a switching from the stuck mode to the slip mode is triggered. This transition takes place during the $2 \rightarrow 3$ interval, where the static coefficient of friction is gradually reduced to the dynamic coefficient of friction (see Figure 3.2.3). Note that the modulation of the friction coefficient is based on the tangential velocity of the GO; hence, this transition provides a velocity dependent transition effect. In this particular simulation, the controller gains are selected to smooth the transition effect by letting the transition take place for a longer duration in time. The continuous change in friction force during this transition can be observed in the intervals $2 \rightarrow 3$, $6 \rightarrow 7$ and $12 \rightarrow 13$ in Figure 3.2.2.

At the marker 3, HIP begins to penetrate into the virtual object (see Figure 3.2.1(e)) where the response force and friction force increase due to the increasing normal force . At 4, the condition for transition from sliding to stuck mode is met but HIP continues its motion along the surface normal as seen in Figure 3.2.1(f). At 4, the continuity of the friction force during the transition from slip to stuck mode is represented. During the $4 \rightarrow 5$ interval HIP continues its penetration and since there is no tangential force to be balanced by a friction force, the friction force is kept still until 5. During the interval $5 \rightarrow 6$ HIP moves parallel to the surface as in the $1 \rightarrow 2$ interval (see Figures 3.2.1(g)–(h)). Hence the friction force increases until it reaches the breakaway friction threshold at the marker 6. The transition from the stuck mode to the slip mode takes place in the $6 \rightarrow 7$ interval and the algorithm stays in the slip mode during the $7 \rightarrow 8$ interval (see Figures 3.2.1(g)–(h)). At 8, HIP begins to move out of the virtual object and it stays out of the virtual object during the $9 \rightarrow 10$ interval (see Figure 3.2.1(i)). In this interval the friction and cumulative forces are calculated as zero since there is no collision with the object. At 10, once again HIP penetrates into the object along the surface normal until 11 as seen in Figure 3.2.1(j). The cumulative force increases due to the increase of normal force and there is no change in the friction force. At 11, HIP continues its motion horizontally towards the second curve (see Figure 3.2.1(k)). Since the breakaway force is not overcome during the interval $11 \rightarrow 12$ and the algorithm is in stuck mode, Voronoi switching is not activated until 12, when the breakaway force threshold is overcome, even though HIP has already crossed the Voronoi switching boundary (see Figure 3.2.1(k)). At 12, the breakaway force threshold is overcome and transition from slip to stuck mode is presented in the

12 → 13 interval (see Figure 3.2.1(l)). The steep decrease in the cumulative and friction forces in this interval is due to the sharp corner, as depicted in the Figure 3.2.1(a).

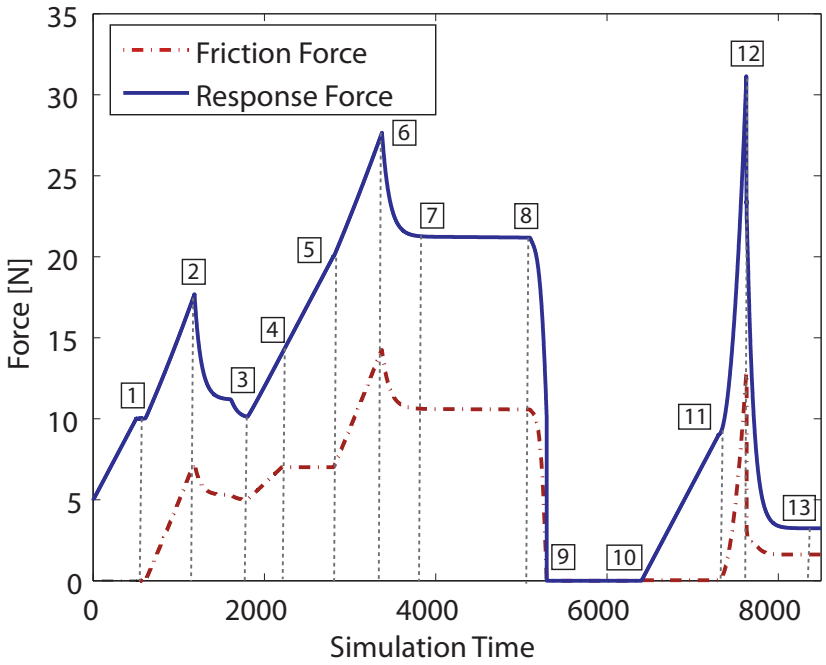

Figure 3.2.2: Plots representing the frictional and normal forces calculated during the simulation

Since the friction force is calculated based on the position of GO, it is meaningful to examine its change with respect to CP. Figure 3.2.4 presents the change in the parameter values $u \in [0\ 1]$ that locate CP and GO with respect to time, during the execution of the pre-determined path as explained above. Until the marker 1, both GO and CP stay collocated while perpendicular penetration takes place. The interval $1 \rightarrow 2$ defines the stuck mode, in which the laws of static friction apply. In this interval, CP moves as user tries to apply a tangential force to overcome the breakaway friction force

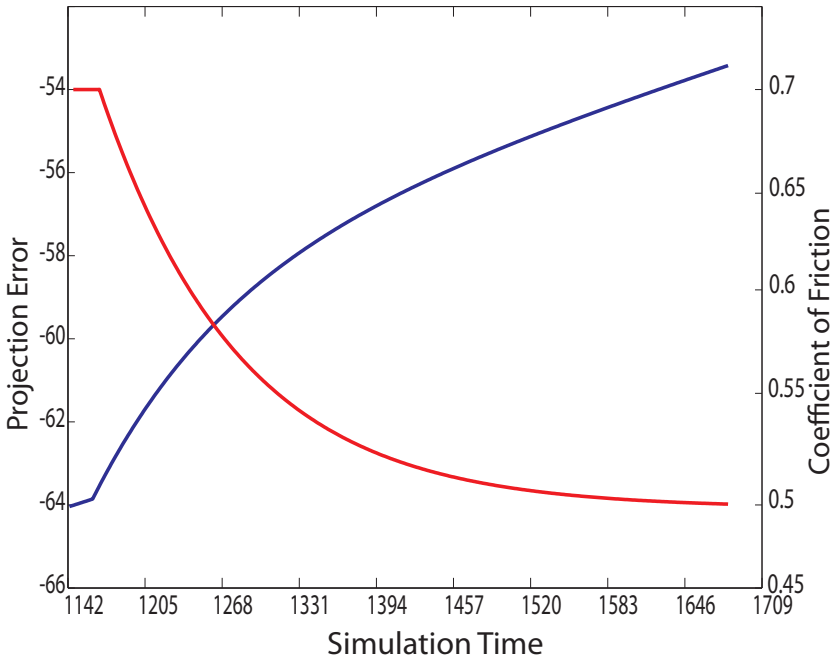

Figure 3.2.3: Simulation of the Stribeck effect through the modulation of the coefficient of friction during transition from stuck to slip mode based on the velocity of GO

threshold, while GO stays fixed. The breakaway friction force threshold is overcome at the marker 2 and GO is released to converge to CP. The transition from the stuck mode to the slip mode takes place during the $2 \rightarrow 3$ interval, while GO is converging towards CP. At marker 3, GO and CP are collocated and the slip mode is triggered. During the slip mode in $3 \rightarrow 4$ interval, GO and CP stay collocated. Although stuck mode is triggered at the marker 4, GO and CP are still collocated which is due to the motion of HIP along the surface normal. During the interval $4 \rightarrow 5$, HIP exhibits the same motion as in the $1 \rightarrow 2$ interval. Hence, GO stays fixed in the interval $5 \rightarrow 6$ whereas it converges towards CP in the $6 \rightarrow 7$ interval. From

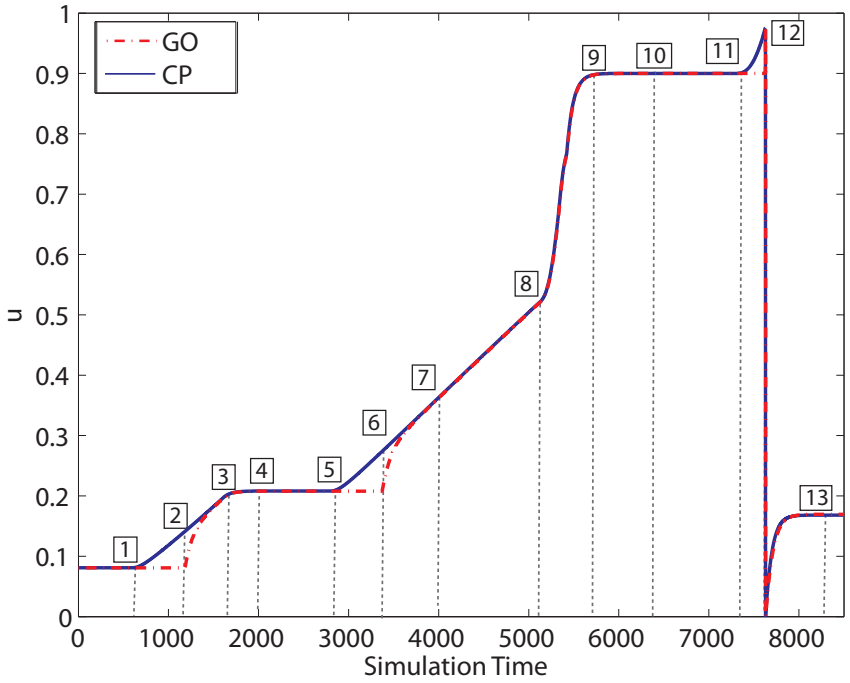

Figure 3.2.4: Changes in the positions of GO and CP while executing the pre-determined path

the marker 7 to 9, the slip mode applies as HIP continues its motion parallel to the surface and then moves out of the virtual object; hence, GO and CP stay collocated throughout this interval. During the interval 9 $\rightarrow$ 10, HIP is outside the virtual object and GO and CP are kept collocated. HIP penetrates into the object along the surface normal during the interval 10 $\rightarrow$ 11 during which stuck mode is active and GO and CP stay collocated in a similar manner as in the 4 $\rightarrow$ 5 interval. At the marker 11, HIP moves towards the second curve in the horizontal direction in order to transition to slip mode. Therefore, during the interval 11 $\rightarrow$ 12, GO stays fixed and CP tracks HIP. At the marker 12, HIP is in the second Voronoi region and a

change from stuck to slip mode is triggered. Both GO and CP are reset to zero since after the marker 12 force calculations are carried out based on the second curve and second curve is parameterized with the parameter values $u \in [0\ 1]$, starting at the end of the first curve. During the interval $12 \rightarrow 13$ dynamic friction applies and GO and CP move collocated.

3.2.2 Friction Rendering in 3-D

The transitions between the sticking and sliding modes of the friction rendering algorithm are also tested on a NURBS surface. The visualization of haptic rendering is carried out using OPENGL and the OPENGL Utility Toolkit (GLUT). The unidirectional communication between Simulink and Integrated Development Environment (IDE) is performed using shared memory.

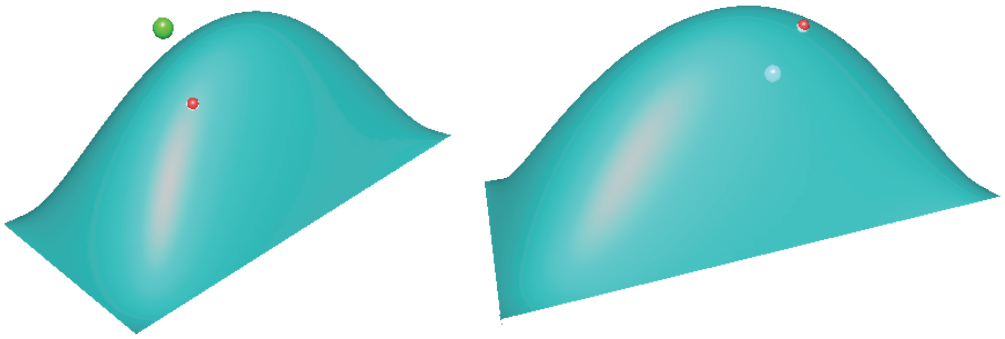

Figure 3.2.5: HIP and GO positions are represented before and after a collision with the NURBS surface

The NURBS surface used in the simulation is represented in Figure 3.2.5 where the surface at the left hand side of the figure depicts the instant before a collision with the surface and the right hand side demonstrates HIP in the

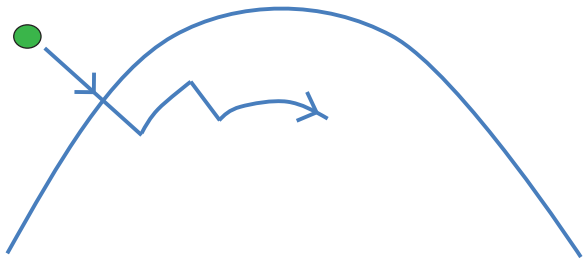

Figure 3.2.6: Trajectory of HIP on the NURBS surface. HIP is allowed to move along the parameter v and its motion along the parameter u is kept still.

surface and GO on the surface after a collision with the NURBS surface. The NURBS surface is parameterized by u and v in two directions. A similar trajectory as in Figure 3.2.1 is tracked by GO where the transitions from stuck to slip and slip to stuck and then again stuck to slip are enforced. The trajectory is represented in Figure 3.2.6 and is generated by keeping the parameter u constant while changing the parameter v. Figure 3.2.7 shows the normal and friction force graphs with respect to time. According to the trajectory, at first instant HIP is located outside the surface where no collision has been occurred yet and forces are calculated as zero. In the interval $1 \rightarrow 2$, the normal force increases as HIP starts to penetrate into the surface. The interval $2 \rightarrow 3$ demonstrates HIP's movement parallel to the surface where the normal force almost stays constant and friction force arises due to the increase of the tangential force. At 3, the breakaway force is overcome and the transition from stuck to slip occurs and friction force is computed by modulating the static and dynamic coefficient of friction. Until 4, HIP continues its motion parallel to the surface. Then it penetrates deeper

into the surface along the surface normal where the normal and friction forces increase due to the increase in the penetration depth. The motion of HIP deeper into the surface normal enforces a change from the slip to stuck mode. At 5, stuck mode is triggered and friction force remains constant due to the zero tangential force and normal force continues to increase. In order to overcome the breakaway force and enter the slipping mode again, HIP moves in the tangential direction parallel to the surface until the rest of the trajectory. The slipping mode is triggered at 7 and dynamic friction applies where the $7 \rightarrow 8$ represents the smooth transition from static to dynamic friction.

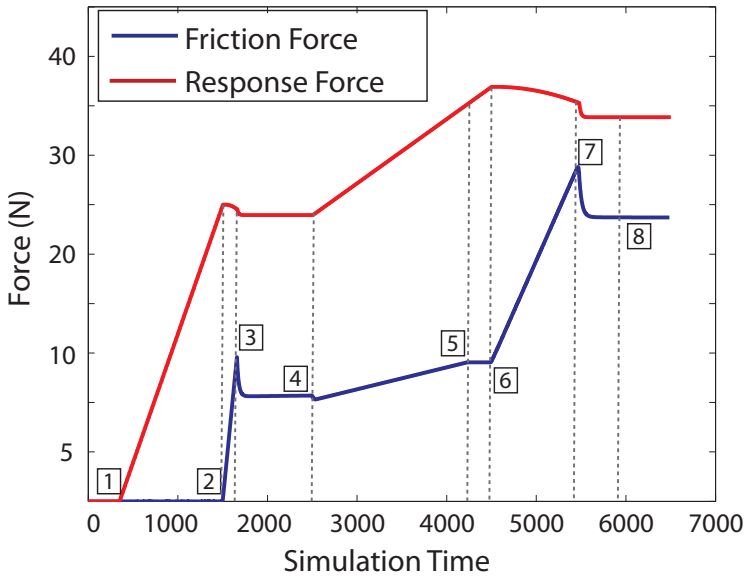

Figure 3.2.7: Plots representing the frictional and normal forces calculated during the friction rendering of a NURBS surface

3.3 Real-time Implementation Results

The controllers are programmed in C and implemented in real-time at 1 kHz utilizing a PC running the RTX real-time operating system. The PC-based control architecture compromises of a workstation simultaneously running RTX real-time operating system and Windows XP SP2. A 2-DoF Pantograph mechanism is used as the haptic interface. The real-time tests are performed with the virtual object depicted in Figure 3.2.1. The stiffness of the object is set to be 1000 N/m.

The algorithm is tested to demonstrate the stick-slip friction behavior and a sample test result is shown in Figure 3.3.1. At the beginning of the test, the user/HIP is outside the virtual object and the friction force is zero. The collision with the virtual object occurs at 6.625 sec.s and the stuck mode is activated due to this collision. From this instant on, the friction force increases as the user applies larger force to overcome the breakaway friction force. The breakaway friction force is exceeded at 6.63 sec.s, and transition from stuck to slip mode initiates. The smoothed transition is completed at 6.725 sec.s, after which the slip mode stays active throughout the simulation.

3.4 Results

Sections 3.2.1 and 3.2.2 demonstrate the results of friction rendering in 2- and 3-D, respectively. Both results are acquired using pre-determined trajectories of HIP. These trajectories are created to validate the mode transitions of the friction rendering algorithm. The results in Section 3.2.1 indicate that the friction rendering algorithm handles transitions from stuck to slip and slip to stuck successfully and without any force discontinuity. A special case

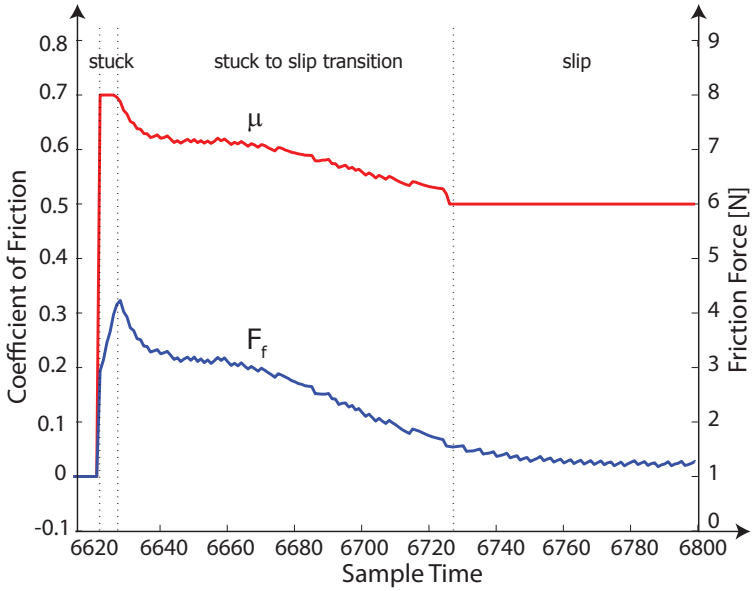

Figure 3.3.1: Real-time implementation of the friction rendering algorithm demonstrating an occurrence of the stuck to slip transition

of static friction is also rendered where the Voronoi-switching is canceled if static friction applies during Voronoi-boundary crossings. Section 3.2.2 represents similar results of friction rendering in 3-D. And results in Section 3.3 verify the mode transitions of friction rendering algorithm in real-time.

Chapter 4

Conclusions and Future Work

4.1 Conclusions

This thesis focuses on the simulation and real-time implementation of haptic and friction rendering of continuous parametric models while enhancing the realism of haptic rendering. The haptic rendering algorithm used to compute the force response resulting from the interaction between a user and a virtual environment, is based on a feedback-stabilized control scheme. The controller for the algorithm is designed and realized using Simulink and implemented in real-time on a PC running the RTX real-time system. A significant property of this haptic rendering algorithm is that it can treat continuous parametric models directly unlike most of the algorithms presented in the literature which require a conversion of continuous models into intermediate representations. A friction rendering algorithm that can also work on continuous parametric models, is presented to improve the haptic rendering approach.

The direct friction rendering method for continuous parametric models implements the stiction model of friction, and can handle transitions from stuck to slip and slip to stuck modes without introducing discontinuous force artifacts. Our algorithm allows for tuning of the friction coefficient during

the mode transitions to simulate Stribeck effect. The approach is robust against drift and numerical noise due to its feedback-stabilized core. More importantly, the friction rendering algorithm is inherently stable and can handle surfaces with high curvature. The approach is physically intuitive and easy to implement. Simulation results are presented and the feasibility of the approach is verified through real-time implementations.

4.2 Future Work

Future work includes human subject experiments and haptic deformation using D-NURBS. Next section introduces the human subject experiments that are performed to measure the effects of friction on shape perception and summarizes the experimental procedures. The definition and preliminary simulation results of D-NURBS are given in Section 4.4.

4.3 Human Subject Experiments

An experiment is designed in order to study the effects of friction on the perception of different parametric shapes by human subjects. The parametric models that are rendered to subjects, are created by joining two curves at a vertex as seen in Figure 4.3.1. There are in total five different possibilities of representation of the virtual object. As shown in Figure 4.3.1 the curves that are used to generate the parametric models, are either convex or concave or straight which are lines. The order of the virtual objects of different shapes that are rendered to subjects are randomized but they are equally represented in the experiment. Haptic rendering of models are carried out with and without friction. During rendering of different parametric models

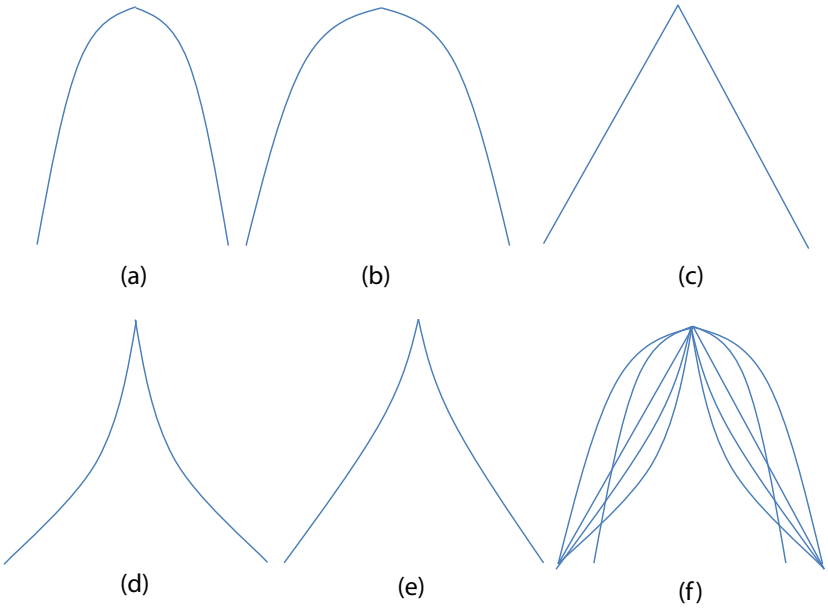

Figure 4.3.1: a) b) Virtual object formed by convex curves c) Virtual object formed by straight lines d) Virtual object generated by concave curves e) The parametric models in a), b), c) and d) are represented on top of each other

subject can not see the rendered models on the monitor. They are expected to feel the rendered geometry via the end effector of a 2-DoF Pantograph without seeing the shape of the virtual object. The experiment consists of several trials and in each trial a single model is rendered to the subject. Following the completion of each trial, the subject is asked to select a model from five different models that are displayed on the monitor. The aim of the experiment is to check whether the subjects will be able to distinguish between geometrically similar shapes with and without friction and how does the existence of friction in haptic rendering contributes to the realism of

haptic rendering.

The human subjects are selected among right-handed people and during the experiment they use their right hand to hold the end effector of the Pantograph. They are restricted to see their hand using a black fabric. After completing each trial subjects make a choice from different shapes but no feedback is given to them regarding their choice. The experiments are being continuing.

4.4 Deformation Modeling using Dynamic NURBS

The realism of haptic interaction with a virtual environment can be improved with the help of friction and texture rendering. Haptic deformation is another approach that can be used to increase this realism. With the help of deformable objects, the user can gain more understanding of the virtual environment he/she is interacting with. When the significance of haptic deformation is taken into account, Dynamic NURBS (D-NURBS) [49], a generalization of the nonuniform rational B-Spline (NURBS) model, can be a useful technique to model deformation in haptic rendering.

D-NURBS represents an approach for physics-based deformation modeling by incorporating mass distributions, energies and other physical quantities into NURBS structure. Due to physics-based modeling of D-NURBS, shape of NURBS models can be dynamically varied and manipulated with respect to time under applied force. More control over dynamically varied NURBS is possible by modifying the parameters related to mass, damping and elasticity. The formulation and equations of motion of D-NURBS are given in the next section.

4.4.1 Formulation of Dynamic NURBS

This section closely follows [49] and provides information regarding D-NURBS geometry.

The definition of a NURBS curve that combines a set of piecewise rational functions with $n+1$ control points and weights is given in the following. Please note that the basis functions are denoted by B which are represented with N in section 2.1.1.

$$C(u) = \frac{\sum_{i=0}^{n} B_{i,k}(u) w_i P_i}{\sum_{i=0}^{n} B_{i,k}(u) w_i} \tag{4.1}$$

where u parameterizes the curve between the values 0 an 1, unless otherwise stated. A NURBS curve has $n+k+1$ knots t_i when the basis functions are of degree $k-1$ that are defined as

$$B_{i,1}(u) = \begin{cases} 1 \text{ if } t_i \leq u < t_{i+1} \\ 0 \text{ otherwise} \end{cases}$$

$$B_{i,k} = \frac{u - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(u) + \frac{t_{i+k} - u}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(u)$$

Now defining a NURBS curve as a function of the parameter u and time t provides the formulation of a D-NURBS curve as follows

$$c(u,t) = \frac{\sum_{i=0}^{n} B_{i,k}(u) w_i p_i}{\sum_{i=0}^{n} B_{i,k}(u) w_i} \tag{4.2}$$

where $p_i(t)$ the control points and weights $w_i(t)$ become functions of time and are generalized coordinates of D-NURBS. For an easy and compact representation of generalized coordinates of D-NURBS the following vectors are defined:

$$p_b(t) = [p_0^T, ..., p_n^T]^T,$$
$$p_w(t) = [w_0, ..., w_n]^T,$$
$$p(t) = [p_0^T w_0 .... p_n^T w_n]^T$$

Following the compact representation of generalized coordinates of a D-NURBS curve, the formulation of a Jacobian matrix that maintains a relationship between the generalized coordinates of the D-NURBS curve and an evaluated points on the curve at an instant, is given in the equations below. For the derivation of equation (4.3), please refer to [49]. The contents of the Jacobian matrix are formed by letting a 3x3 matrix whose diagonal entries are the rational basis functions

$$N_i(u,p) = \frac{\partial c}{\partial p_i} = \frac{w_i B_{i,k}}{\sum_{j=0}^{n} w_j B_{j,k}}$$

and letting the 3x1 vector

$$w_i(u,p) = \frac{\partial c}{\partial w_i} = \frac{\sum_{j=0}^{n} (p_i - p_j) w_j B_{i,k} B_{j,k}}{(\sum_{j=0}^{n} w_j B_{j,k})^2}.$$

For $i = 0, ..., n$ collecting B_i into B and w_i into W yields

$$B(u,p) = [B_0 ... B_n],$$
$$B(u,p) = [w_0 ... w_n],$$
$$J(u,p) = [B_0 w_0 ... \quad B_n w_n].$$

Now the D-NURBS curve can be expressed as the product of the generalized coordinate vector and the Jacobian matrix.

$$c(u,p) = Jp \tag{4.3}$$

The equations of motion of D-NURBS are formed using Lagrangian dynamics. When the control points and imaginary straight lines connecting each

pair of them are considered, a D-NURBS geometry can be interpreted as a planar mechanism. In this regard, the control points and straight lines correspond to joints and links, respectively. Letting f_i the applied force acting on the generalized coordinated p_i, T the kinetic energy, U the potential energy and F the dissipation energy the Lagrangian equations of motion can be developed as

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{p}_i} - \frac{d}{dt}\frac{\partial T}{\partial p_i} + \frac{\partial F}{\partial \dot{p}_i} + \frac{\partial U}{\partial p_i} = fi \tag{4.4}$$

By applying (4.4) equations of motion of D-NURBS are given by

$$M\ddot{p} + D\dot{p} + Kp = f_p - I\dot{p} \tag{4.5}$$

The mass matrix is defined as

$$M(p) = \int \mu J^T J du \tag{4.6}$$

where $\mu(u)$ is the mass density function defined over the parametric domain of the curve. The damping density function is denoted by $\gamma(u)$ and the density matrix is given as following

$$D(p) = \int \gamma J^T J du \tag{4.7}$$

The $\alpha(u)$ and $\beta(u)$ are defined as the elasticity functions which aim to control local tension and rigidity. The stiffnes matrix is given by

$$K(p) = \int \alpha_1 J_u^T J_u + \beta J_{uu}^T J_{uu} du \tag{4.8}$$

Finally, the generalized force vector and the inertia matrix are found by

$$\begin{aligned} f_p(p) &= \int J^T f(u,t) du, \\ I(p) &= \mu J^T \dot{J} du \end{aligned} \tag{4.9}$$

4.4.2 Simulation Results

Figures 4.4.1(a)-(d) depict deformation of a B-spline under applied force. The figures show both deformed and undeformed shapes of the B-spline. The B-spline before deformation is colored in blue and formed using four control points which are illustrated as four blue circles. In each of the figures the force is applied along the straight green line at its point of contact with the curve. In the figures, the exact force application points on the curve are $u = 0.5$, $u = 0$, $u = 0$ and $u = 1$, respectively. The dashed lines represent the deformed B-spline after the applied force and red circles show newly located control points of the curve after deformation.

The coordinates of the control points under force application are calculated by solving the equation of motion for Dynamic NURBS given in Section 4.4.1. The integration of dynamical differential equations is carried out using Gaussian quadrature, which is used to approximate the definite integral of an arbitrary function. The results shown in the figures indicate that the B-spline is deformed towards the direction of the applied force which is physically meaningful. The obtained deformation can be rendered more realistically by adding more control points to the B-spline and defining constraints on chosen control points.

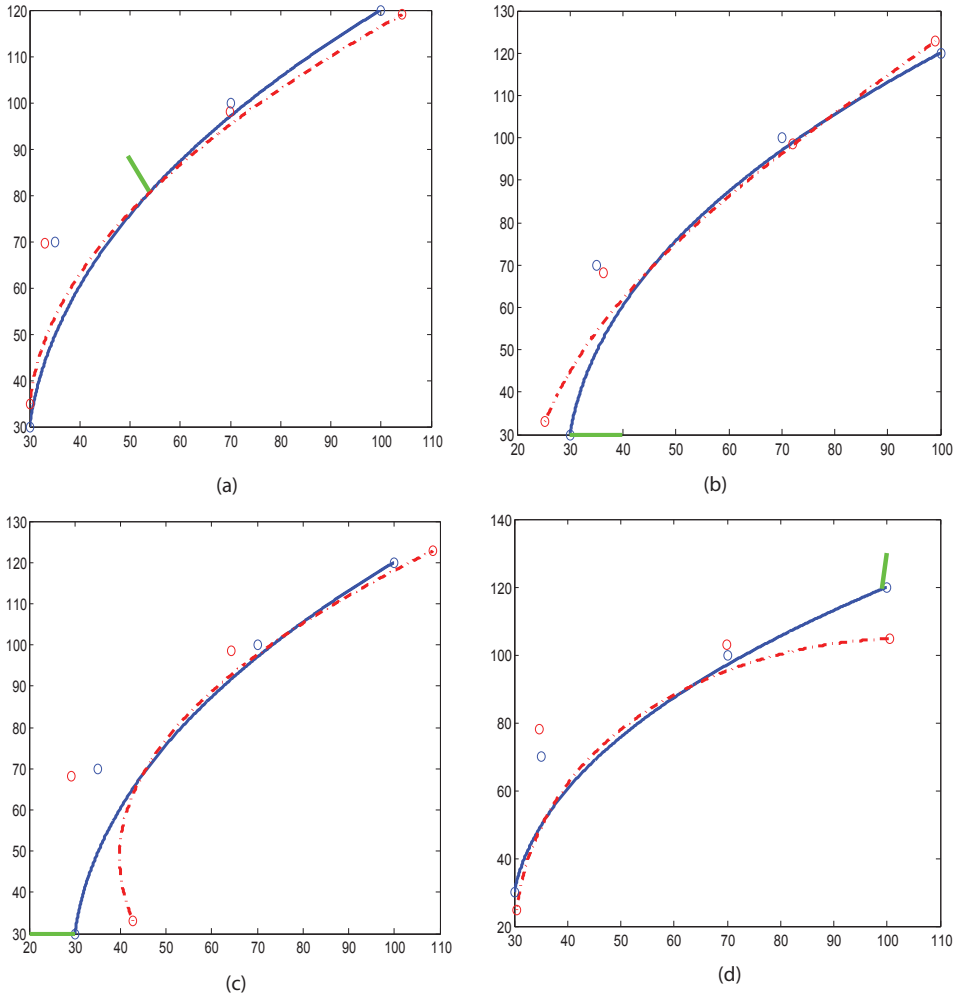

Figure 4.4.1: Deformation of a B-Spline under force application. The dashed B-Splines represent the deformed objects. The B-Splines are parameterized between $u = 0$ and $u = 1$. The force is applied to the B-Splines at $u = 0.5$, $u = 0$, $u = 0$ and $u = 1$, respectively.

Bibliography

[1] Volkan Patoğlu. *Guaranteed Stability for Collision Detection and Simulation of Hybrid Dynamical Systems*. PhD thesis, University of Michigan, 2005.

[2] John M. Hollerbach and David E. Johnson. Virtual environment rendering. In *in Human and Machine Haptics*. MIT Press, 2000.

[3] J.E. Colgate, M.C. Stanley, and J.M. Brown. Issues in the haptic display of tool use. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 140 –145 vol.3, 5-9 1995.

[4] R.J. Adams and B. Hannaford. Stable haptic interaction with virtual environments. *Robotics and Automation, IEEE Transactions on*, 15(3): 465 –474, jun 1999. ISSN 1042-296X.

[5] Ming C. Lin, Miguel Otaduy, Ming C. Lin, and Miguel Otaduy. *Haptic Rendering: Foundations, Algorithms and Applications*. A. K. Peters, Ltd., Natick, MA, USA, 2008. ISBN 1568813325, 9781568813325.

[6] J.E. Colgate and G. Schenkel. Passivity of a class of sampled-data sys-

tems: application to haptic interfaces. In *American Control Conference, 1994*, volume 3, pages 3236 – 3240 vol.3, 29 1994.

[7] T.M. Massie and J.K. Salisbury. The phantom haptic interface: a device for probling virtual objects. *Proc. ASME Dynamic Systems and Control Division*, pages 295–301, 1994.

[8] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. Fast procedure for computing the distance between convex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, 1988.

[9] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1008–1014, 1991.

[10] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3):177–208, 1998.

[11] C.B. Zilles and J.K. Salisbury. A constraint-based god-object method for haptic display. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 146–151 vol.3, Aug 1995.

[12] Srinivasan M.A. Morgenbesser H.B. Force shading for haptic shape perception. *Proceedings of the ASME Dynamics Systems and Control Division*, 58:407–412, 1996.

[13] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib. The haptic display of complex graphical environments. In *Computer Graphics Proceedings, Annual Conference Series, SIGGRAPH 97, Los Angeles, California,*, pages 345–352, Aug 1997.

[14] J. M. Snyder. Interactive tool for placing curved surfaces without interpenetration. In *Proceedings of the ACM SIGGRAPH*, pages 209–218, 1995.

[15] Y. Adachi, T. Kumano, and K. Ogino. Intermediate representation for stiff virtual objects. In *Proceedings Virtual Reality Annual International Symposium*, pages 203–210, 1995.

[16] P. Stewart, Y. Chen, and P. Buttolo. CAD data representations for haptic virtual prototyping. In *Proceedings of ASME Design Engineering Technical Conference*, 1997.

[17] T. V. Thompson II, D. D. Nelson, E. Cohen, and J. Hollerbach. Maneuverable NURBS models within a haptic virtual environment. In *Proceedings of ASME International Mechanical Engineering Congress and Exposition*, volume 61, pages 37–44, 1997.

[18] D. D. Nelson, D. E. Johnson, and E. Cohen. Haptic rendering of surface-to-surface sculpted model interaction. In *Proceedings ASME Dynamic Systems and Control Division*, volume 67, pages 101–108, 1999.

[19] V. Patoglu and R. B. Gillespie. Extremal distance maintenance for parametric curves and surfaces. In *Proc. 2002 IEEE International Conference on Robotics and Automation*, pages 2817–2823, 2002.

[20] V. Patoglu and R. B. Gillespie. Haptic rendering of parametric surfaces using a feedback stabilized extremal distance tracking algorithm. In *Proc. IEEE International Conference on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, volume 3, pages 391 – 399, 2004.

[21] V. Patoglu and R. B. Gillespie. Feedback stabilized minimum distance maintenance for convex parametric surfaces. *IEEE Transactions on Robotics*, 21(5):1009– 1016, 2005.

[22] Brian Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17:177–208, 1998.

[23] S.A. Ehmann and M.C. Lin. Accelerated proximity queries between convex polyhedra by multi-level voronoi marching. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, pages 2101 –2106 vol.3, 2000.

[24] Ming C. Lin and Dinesh Manocha. Efficient contact determination between geometric models. Technical report, International Journal of Computational Geometry and Applications.

[25] Tom Duff. Interval arithmetic recursive subdivision for implicit functions and constructive solid geometry. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 131–138, New York, NY, USA, 1992. ACM. ISBN 0-89791-479-1.

[26] Brian Von Herzen, Alan H. Barr, and Harold R. Zatz. Geometric collisions for time-dependent parametric surfaces. *SIGGRAPH Comput. Graph.*, 24(4):39–48, 1990. ISSN 0097-8930.

[27] Thomas V. Thompson, II, David E. Johnson, and Elaine Cohen. Direct haptic rendering of sculptured models. In *I3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 167–176, New York, NY, USA, 1997. ACM. ISBN 0-89791-884-3.

[28] David E. Johnson and Elaine Cohen. An improved method for haptic tracing of a sculptured surface, 1998.

[29] C. Richard and M.R. Cutkosky. Friction modeling and display in haptic applications involving user performance. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pages 605 – 611 vol.1, 2002.

[30] R. Stribeck. Die wesentlichen eigenschaften der gleit- und rollenlager the key qualities of sliding and roller bearings. *Zeitschrift des Vereines Seutscher Ingenieure*, 46:1342–1348, 1902.

[31] Dean Karnopp. Computer simulation of stick-slip friction in mechanical dynamic systems. *Journal of Dynamic Systems, Measurement, and Control*, 107(1):100–103, 1985.

[32] H. Olsson, K. J. strm, C. Canudas de Wit, M. Gfvert, and P. Lischinsky. Friction models and friction compensation. *Europian Journal of Control*, 4(3):176–195, 1998.

[33] P. R. Dahl. A solid friction model. Number TOR-158(3107-18), 1968.

[34] David A. Haessig and Bernard Friedland. On the modeling and simulation of friction. In *American Control Conference, 1990*, pages 1256–1261, May 1990.

[35] C. Canudas de Wit, H. Olsson, K.J. Astrom, and P. Lischinsky. A new model for control of systems with friction. *Automatic Control, IEEE Transactions on*, 40(3):419 –425, March 1995. ISSN 0018-9286.

[36] P.-A. Bliman and M. Sorine. Friction modelling by hysteresis operators. application to dahl, stiction and stribeck effects. In *Proceedings of the Conference Models of Hysteresis,Trento,Italy*, 1991.

[37] P.-A. Bliman and M. Sorine. A system-theoretic approach of systems with hysteresis. application to friction modelling and compensation. In *Proceedings of the second European Control Conference, Groningen, The Netherlands*, pages 1844–49, 1993.

[38] P.-A. Bliman and M. Sorine. Easy-to-use realistic dry friction models for automatic control. In *Proceedings of 3rd European Control Conference, Rome, Italy*, pages 3788–3794, 1995.

[39] Laehyun Kim, A. Kyrikou, G.S. Sukhatme, and M. Desbrun. An implicit-based haptic rendering technique. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2943–2948 vol.3, 2002.

[40] W.R. Mark, S.C Randolf, M. Finch, J.M. Van Verth, and R.M. Taylor III. Adding force feedback to graphics systems:issues and solutions. *Association for Computing Machinery SIGGRAPH, New York, NY, ETATS-UNIS*, pages 447–452, 1996. ISSN 1069-529X.

[41] S. E. Salcudean and T. D. Vlaar. On the emulation of stiff walls and static friction with a magnetically levitated input/output device. *Journal of Dynamic Systems, Measurement and Control*, 119(1):127–132, March 1997. ISSN 0022-0434.

[42] K. Salisbury, D. Brocki, T. Massiet, N. Swarupf, and C. Zillest. Haptic

rendering: Programming touch interaction with virtual objects. *Symposium on Interactive 3D Graphics, Monterey CA USA*, 1995.

[43] N. Melder and W.S. Harwin. Extending the friction cone algorithm for arbitrary polygon based haptic objects. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings. 12th International Symposium on*, pages 234–241, March 2004.

[44] J. Chen, C. DiMattia, R. M. Taylor II, M. Falvo, P. Thiansathon, and R. Superfine. Sticking to the point: A friction and adhesion model for simulated surfaces. volume 61, pages 167–171, 1997.

[45] Vincent Hayward and Brian Armstrong. A new computational model of friction applied to haptic rendering. In *In Experimental Robotics VI, P. I. Corke and J. Trevelyan (Eds.), Lecture Notes in Control and Information Sciences*, pages 403–412. Springer-Verlag, 2000.

[46] Les Piegl and Wayne Tiller. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997. ISBN 3-540-61545-8.

[47] K. Salisbury and Tarr C. Haptic rendering of surfaces defined by implicit functions. In *Proceedings on ASME 6th Annual Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Dallas, TX*, volume 3, pages 61–68, Nov 1997.

[48] Sisl homepage - sintef. URL `http://www.sintef.no/Informasjons--og-kommunikasjonsteknologi-IKT/Anvendt-matematikk/Fagomrader/Geometri/Prosjekter/The-SISL-Nurbs-Library/SISL-Homepage/`.

[49] Hong Qin and D. Terzopoulos. D-nurbs: a physics-based framework for geometric design. *Visualization and Computer Graphics, IEEE Transactions on*, 2(1):85 –96, mar 1996. ISSN 1077-2626.