Haptic Virtual Fixtures for Robotic Surgery

Sina Nia Kosari

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

University of Washington

2013

Reading Committee:

Howard Jay Chizeck, Chair

Blake Hannaford

Payman Arabshahi

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Haptic Virtual Fixtures for Robotic Surgery

Sina Nia Kosari

Chair of the Supervisory Committee:
Professor Howard Jay Chizeck
Electrical Engineering

This dissertation presents a method to protect objects from touch by the robot during teleoperation. Using the point cloud obtained in real-time by a depth camera, a virtual fixture is placed around protected areas of the workspace to keep the robot at a safe distance. Upon violating the protected area, the operator receives force feedback that opposes motion inside the forbidden region. Due to the fast nature of our haptic rendering algorithm, this method can protect moving objects in a dynamic environment as it is captured in real-time by the depth camera.

The methods are implemented and evaluated on a surgical robot. In addition, adaptive techniques are developed to improve the control performance of the robot a) in contact with soft tissue with nonlinear, unknown and time varying biomechanical properties, b) in the absence of end effector measurements and c) under variable cable tension.

It is anticipated that virtual fixtures will increase the speed and accuracy of procedures and will reduce the operator's fatigue.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

HAPTIC RENDERING: The process of generating computer controlled forces and/or torques for display to an operator in order to create the sensation of touch.

HAPTIC DEVICE: Devices that are capable of applying forces and/or torques to the operator in a controlled fashion.

DEPTH CAMERA: A camera capable of providing depth information for each pixel in the frame.

RGB-D CAMERA: A camera capable of providing color information in addition to depth information for each pixel in the frame.

POINT CLOUD: A set of points in space often generated by depth cameras and 3D scanners.

TELEOPERATION: The act of remotely controlling a robot as opposed to autonomous control.

VIRTAL FIXTURE: Abstract sensory information overlaid on top of reflected sensory feedback from a remote environment [1].

FORBIDDEN REGION: Also known as a no-fly zone, is a region in space where is robot is not allowed to enter.

PROXIMAL WORLD: The location of the operator in the teleoperation system.

REMOTE WORLD: The location of the remote robot and sensors in the teleoperation system.

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to my family for their love and support.

I would like to thank my advisors, Howard Jay Chizeck and Blake Hannaford, for encouraging creativity and for their guidance throughout my professional development. They have set an exceptional example as mentors, researchers and teachers.

I am deeply grateful to the committee members for providing invaluable feedback and helping me keep sight of the big picture while focusing on immediate goals.

Several people have been instrumental in the success of this work and deserve acknowledgement. Fredrik Ryden for his significant role in development and implementation of haptic rendering and virtual fixture technology, Srikrishnan Ramadurai for his role in mathematical modeling of cable driven systems and Hawkeye King and Lee White for their help with the Raven Software. I would also like to thank Dr. Tom Lendvay for providing surgical expertise and feedback to keep the project in line with real-life needs and for devoting his time to this work.

Finally, I would like to thank all the members of the UW BioRobotics Laboratory for their help and for creating an environment to thrive.

DEDICATION

to my family

Chapter 1

INTRODUCTION

Over the past three decades robot assisted minimally invasive surgery has transformed from a concept to the standard of operation for many procedures. Since the FDA approval and public offering of the *da Vinci* ® Surgical System by Intuitive Surgical Inc. (Sunnyvale, CA, USA) in the year 2000, over 1.4 million robot assisted minimally invasive procedures have been performed. The *da Vinci* ® surgical system has been granted FDA approval for urology, gynecology, pediatric, transoral otolaryngology, prostatectomy, cholecystectomy and hysterectomy procedures among others. With over 2700 systems installed worldwide, this surgical system is used to perform four out of five radical prostatectomies in the United States.

Clinical benefits of robot assisted minimally invasive surgery over open surgery include shorter hospital stays [2, 3, 4, 5], less blood loss [2, 3, 4, 6, 7, 8], less need for a blood transfusion [3, 4, 6, 8], lower risk of complications [4, 8], lower risk of wound infections [8], less pain [6, 9], faster recovery [7, 10] and smaller incisions and scarring.

While the benefits of robot assisted minimally invasive surgery over open surgery are clear, there is no consensus on its advantages over laparoscopic surgery. Proponents of the robot assisted method argue that it enables far more dexterous manoeuvres with its wristed instruments compared to laparoscopic instruments. In addition, the robot reduces hand tremor and enables precise hand movements due to its motion scaling feature. On the other hand, no haptic feedback is provided to the surgeon in robotic surgery and the technology is associated with much higher costs compared to laparoscopic surgery [11, 12, 13]. A recent study of $264,758$ women who underwent hysterectomy for benign gynecologic disorders at 441 hospitals in the United States shows that robot assisted procedures increased from 0.5% in 2007 to 9.5% in 2010 [14]. According to this study, the complication rate, transfusion requirements and the rate of discharge to a nursing facility were similar for both robot assisted and laparoscopic hysterectomies. However, robot assisted surgery increased the cost by $2189 per case compared to laparoscopic hysterectomy. This study and several others (e.g.

see [15, 16, 17]) have raised doubts over the clinical benefits of robot assisted surgery in comparison to laparoscopic surgery.

In today's robot assisted minimally invasive surgery, the robot end effector behaves as an extension of the surgeon's hand. With the exception of tremor dampening and motion scaling, the robotic instruments follow the motions of the surgeon. However, the large technology base in robotics research creates a great potential that still needs to be exploited for minimally invasive surgery. Advancements in mechanical design and manufacturing, 3D imaging and registration, motion planning, sensor fusion and dynamics and control have made an impact on many robotic applications and could potentially give robot assisted minimally invasive surgery the edge over laparoscopic surgery. One successful application of such technologies in surgery was in hip replacement surgery [18]. Given the CT images the robot can cut the shape of the prosthetic replacement joint into the femur with superior precision than was possible with manual instruments [19]. Knee surgery, spine surgery and neurosurgery are among areas that have benefited from the advancements in imaging and robotics technology. However, several obstacles stand in the way of applying these technologies to minimally invasive general and thoracic surgery. These challenges include interaction with highly deformable soft tissue and motion of the organs due to respiration and heartbeat, creating a dynamically changing environment. Due to these limitations minimally invasive general and thoracic procedures have been limited to manual teleoperation in which the robot follows the surgeons hand motion without any autonomy.

In addition, one of the main drawbacks of the commercially available robot assisted surgical system is the lack of force feedback to the surgeon. Although the surgical console is equipped with actuators to provide kinaesthetic feedback, in the commercially available design this feedback is mostly perceivable in contact with rigid objects e.g. tool on tool collisions, rather than contact with soft tissue [20]. This is mainly due to the issues that arise in transparency and stability of the operation by closing the kinaesthetic feedback loop. In addition, this feedback in provided after contact is made and therefore may not be used to prevent contact. Due to these shortcoming, the surgeon must close the control loop by relying on his/her experience in extracting visual cues from the stereoscopic images.

The literature on bilateral teleoperation is rich in methods for providing haptic feedback to the operator (see [21] for a survey of these methods). However, true haptic feedback requires force

sensors for accurate measurement of contact forces [22]. Although several attempts have been made to design force sensors for laparoscopic instruments, e.g. [23, 24, 25], severe size and sterilizability constraints have prevented the widespread use of these sensors. Furthermore, the addition of sensors can significantly increase the cost of laparoscopic instruments.

Surgery often involves manipulation in the vicinity of sensitive or delicate structures such as nerves and arteries. A limiting factor in the success of these operations is the need for precise motions and interactions in a dynamically changing environment created by tissue deformations and motions. For example, the surgeon may need to ablate tissue while protecting adjacent nerves or vasculature and avoiding contact with moving anatomical structures and other surgical tools. The problem is further amplified in the absence of haptic feedback to the operator. If contact with such structures is not visually detected, collisions can cause serious damage to the patient.

Haptic virtual fixtures are computer-generated constraints that limit the motion of the robot. Any violation of these constraints will be opposed by a resisting force. Safety and performance of the operation can be enhanced by placing the constraint around sensitive objects, thus constructing a "forbidden region" in the task space. If the operator tries to drive the robot into the constrained space, a resisting force to the hand controls or the robot will be rendered to resist or prevent the violation of the constraint.

This dissertation focuses on methods for creating forbidden region haptic virtual fixtures. Due to the dynamic nature of the environment, such constraints must be updated in real-time and cannot be generated solely from preoperative images. Recent advances in RGB-D imaging have made it economical to obtain rich information from the environment in the form of a depth map (also known as a point cloud). Our methods focus on creating the virtual fixtures from this information. Even though today's commercially available RGB-D cameras do not meet the requirements for use in surgery, the trend of this technology in the recent years suggests that a depth endoscope is not far-fetched. In addition, our goal is to provide haptic feedback at 1 kHz, as recommended in the literature[26], in order to render rigid contact in a stable fashion when the operator violates a constraint.

The main contributions of this dissertation include:

- Developing a novel real-time haptic rendering method from streaming point clouds.

- Developing real-time forbidden region haptic virtual fixtures from streaming point clouds.

- Designing an optimal control and estimation method for controlling a cable-driven mechanism in contact with soft tissue using Unscented Kalman Filter (UKF) and Differential Dynamic Programming (DDP).

- Developing a method for estimating cable tension and compensating for its effects on control performance.

The rest of this document is organized as follows. Chapter 2 proposes a novel method for haptic rendering of streaming point clouds. This method enables the operator to feel the shapes of objects in real-time as captured by an RGB-D camera. These methods are extended for creation of virtual fixtures and are implemented using a surgical robot in Chapter 3. The problem of control of cable-driven robots in contact with soft tissue and under variable tension is addressed in Chapters 4 and 5 respectively. Shortcomings of the proposed methods and directions for future work are described in Chapter 6.

Chapter 2

HAPTIC RENDERING FROM STREAMING POINT CLOUDS [1]

Recent advancements in RGB-D cameras have resulted in low cost and low latency cameras with good resolution. An example of this is the Xbox Kinect. This camera gives a depth value for each pixel and using these values one can build a set of points in space, referred to as a *point cloud*. A point cloud derived from Kinect appears as a set of time varying points in space, updated in real-time at 30 Hz.

The first step in achieving haptic virtual fixtures from a point cloud is developing algorithms for real-time haptic rendering from arbitrary points in space. This chapter describes a novel haptic rendering method from streaming point clouds.

While there exists a rich body of literature on haptic rendering for virtual objects with defined surfaces, little work has been done on haptic rendering of virtual environments defined by real-time point clouds. Haptic rendering from points was first developed for virtual environments with complex meshes, with the goal of improving the computational performance [28]. In that work, a point map, also known as a Voxmap PointShell is created by sampling of meshes. As a consequence, each point is associated with additional information regarding its corresponding mesh, such as the surface normal. This information is not available in point clouds captured by RGB-D cameras.

Haptic rendering from static point clouds without normal information has been investigated by several groups. For example, surfaces can be created from points using Moving Least Squares [29] [30] [31]. Haptic rendering from static point clouds without normal information was also implemented in [32]. Their method involves building a collision tree around the points by placing bounding boxes around each point in the scene. Building this tree requires pre-processing. An innovation of our work is that the haptic rendering can be done for dynamically changing point

[1] ©[2011] IEEE. Reprinted, with permission, from [27] Rydén, F., Nia Kosari, S., & Chizeck, H. J. "Proxy method for fast haptic rendering from time varying point clouds." In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on (pp. 2614-2619).

clouds, sufficiently fast for real-time control applications.

Haptic rendering from RGB-D data was first investigated in [33]. Their method requires pre-processing of recorded camera data and therefore is not suitable for real-time rendering of dynamic environments.

One could argue that the simplest solution would be to construct surfaces and meshes of the whole point cloud at every frame, and then use traditional haptic rendering algorithms. This would require a large number of computations. The method developed in this chapter avoids that requirement.

To the best of the authors knowledge, there exists no method prior to the work of our group for real-time haptic rendering when only the location of the points in the time varying point cloud is known. This chapter develops such a method by extending the proxy method for haptic rendering [34]. It builds upon preliminary results that were recently demonstrated in [35].

2.1 Background and Literature Review

2.1.1 Haptic Interaction Point and Proxy

A Haptic Interaction Point (HIP) is to a haptic device what a mouse pointer is to a computer mouse. Ideally the HIP should not be able to penetrate objects. However, this is not possible since we only have control over the force applied to the HIP (and thus the user), and not over its position (which is generated by the user). For this reason, consider instead a virtual massless point that represents the ideal position of the HIP on the surface; i.e. it can not penetrate objects. Such a point was suggested in [36] and was named a *god-object* even though it actually is a point. This massless point is useful for haptic rendering in environments where shapes are well-defined. A common way to render haptic forces is to attach a spring between the HIP and the ideal point on the surface. The spring constant can then be set to an appropriate value.

A similar idea was presented in [34] where the massless point was replaced with a massless sphere with some volume. This sphere is referred to as the *proxy* and serves the same purpose as the god-object as shown in Fig. 2.1.

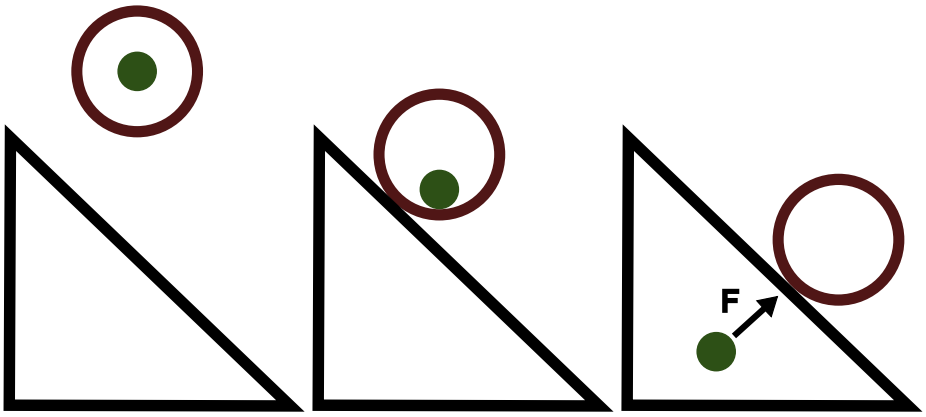

Figure 2.1: Illustration of the proxy method in two dimensions. **Left:** The proxy (big circle) and HIP (small filled circle) in free motion. **Middle:** At the moment of contact, the HIP continues to penetrate the object while the proxy is constrained by the surface. **Right:** The HIP is inside the object and proxy remains on the surface. F indicates the force on the HIP.

2.1.2 RGB-D cameras

Recent advancements in the gaming industry have reduced the cost for RGB-D cameras by more than an order of magnitude while providing acceptable latency and depth resolution. RGB-D cameras capture images similar to a regular video camera and in addition also capture a depth image containing depth values for each pixel. This can be done using different techniques. The most common are active stereo, time-of-flight and projected pattern [37].

The camera used in this chapter is Xbox Kinect. The four important parts of the Kinect for this work are:

1. The infrared (IR) projector that projects a defined dot pattern (see Fig. 2.2).

2. The IR camera that captures the reflections of the projected IR dot pattern. This is then used to create a depth map based upon triangulation.

3. A regular (RGB) camera that captures an image that can be superimposed onto the depth map.

4. Accelerometers that obtain the X-Z-orientation of the camera.

Processing is performed on the captured dot pattern resulting in a 640x480 matrix $M(u,v)$ containing a depth value for each pixel [38]. The depth values range between 0 and 2047 (11 bits) with $1cm$ depth resolution and $3mm$ horizontal/vertical resolution at $2m$ distance. Capturing of data and signal processing is done in real-time at the rate of 30 Hz. [39]

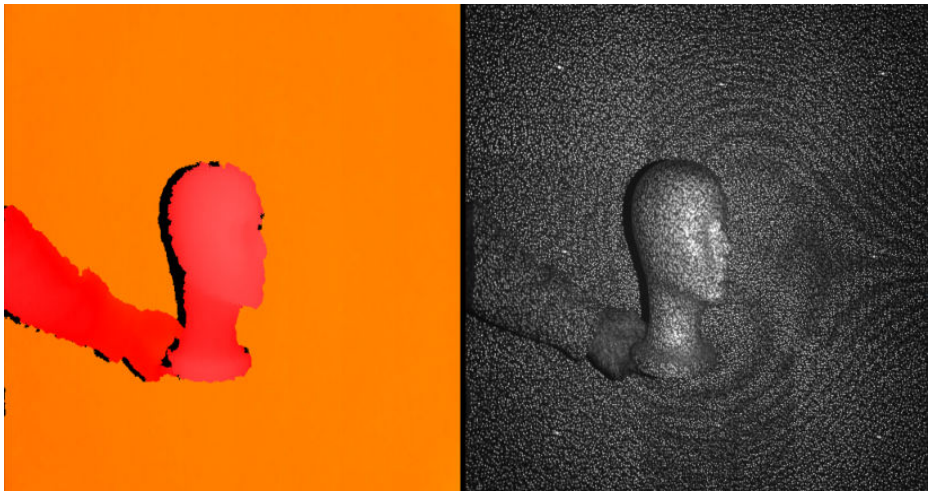

Figure 2.2: **Left:** Color coded depth map generated from Kinect depth data. **Right:** IR dot pattern projected on a Styrofoam head held up against a flat background.

2.1.3 Interpretation of Kinect Depth Data

In order to render haptic forces, depth data needs to be represented in a Cartesian coordinate system. We denote Kinect's coordinate system by K and the Cartesian coordinate system by C. The transformation between the two coordinate systems is done by first rotating, translating and scaling the data with respect to the camera intrinsics. Then the data is rotated with respect to the cameras X-Z-orientation as given by the accelerometers. The two transformations are invertible.

2.2 Haptic Rendering for Point Clouds

The point clouds referred to in this chapter are sets of points, representing physical objects, that are captured by the Kinect RGB-D camera. As mentioned earlier, no other information about the underlying objects is given, such as surface normals or information about what object a specific

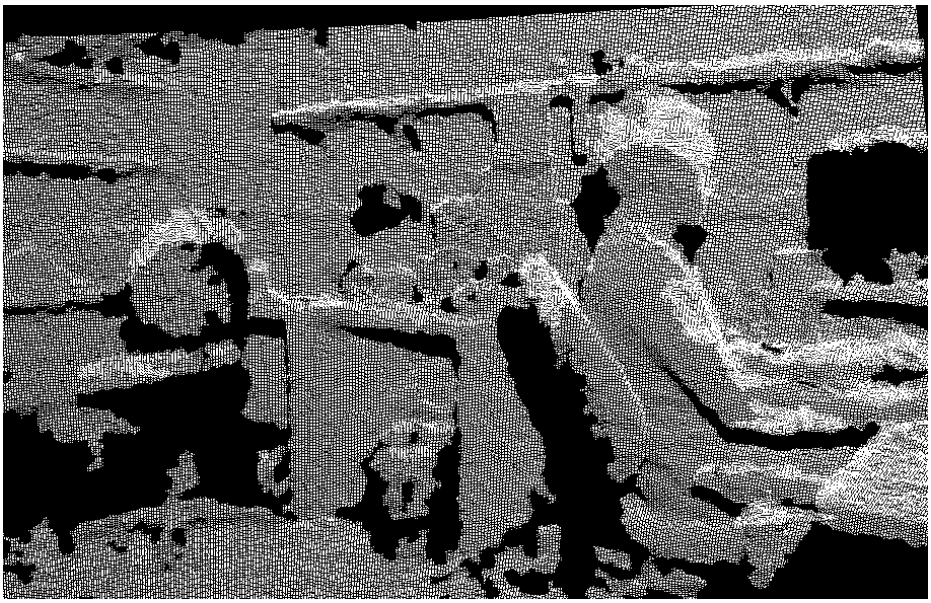

Figure 2.3: Depth data from an office visualized as a point cloud. The black spots correspond to regions without depth information.

point in the cloud corresponds to.

This is different from the established haptic rendering methods for Voxmap Pointshell, where known meshes from a CAD model are sampled and therefore surface normal information is available [28].

Fig. 2.3 shows a Kinect derived point cloud visualized in OpenGL. The black regions represent the areas where depth information could not be obtained.

We develop here a haptic rendering algorithm to extend the notion of proxy to time varying point cloud data derived from an RGB-D camera.

In order to render the haptic forces the following is done:

1. Depth data from the RGB-D camera is captured.

2. The position of the haptic device is read and the HIP is placed accordingly.

3. The proxy is moved towards the HIP with respect to point cloud constraints using the *Proxy Movement Algorithm* which is described below.

4. Forces are rendered based on the distance between the proxy and the HIP.

The camera captures depth data at a rate of 30 Hz. The haptic device sends position and receives forces at a rate of 1000 Hz. The rate of the Proxy Movement Algorithm is variable. The basic properties of the resulting haptic rendering is illustrated in Fig. 2.4.

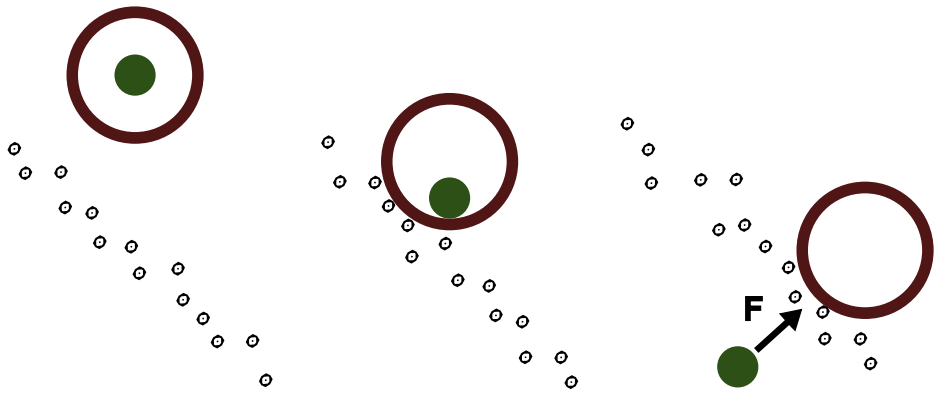

Figure 2.4: Two dimensional illustration of haptic rendering from time varying point clouds (where haptic rendering is based on points rather than surfaces as in Fig. 2.1). **Left:** The proxy (big circle) and HIP (small filled circle) in free motion. **Middle:** At the moment of contact, the HIP continues to move through the points while the proxy does not. **Right:** The HIP is behind the points and the proxy remains on an estimated surface. F indicates the force on the HIP.

2.2.1 The Proxy

The proxy has three possible states: *free motion*, *contact* and *entrenchment* (the proxy digging into the point cloud). Consider the following three radii R_1, R_2, R_3 extending out from the center of the proxy denoted by $\mathbf{X}_{\mathbf{proxy}}$ (see Fig. 2.5). If there are any points within R_1 of $\mathbf{X}_{\mathbf{proxy}}$, the proxy is said to be *entrenched*. If there are any points between R_1 and R_2 (but not within R_1) the proxy is said to be *in contact*. If there are no points within R_2 the proxy is said to be in *free motion*. Points within R_3 will be used to estimate surface normals.

The quantity $R_2 - R_1$ is chosen to be just larger than the anticipated noise level. Large values will degrade contact detection. Radius R_3 is chosen sufficiently large to get good surface normal estimation. If R_3 is chosen too large, the sharpness of the surface normal estimation will be reduced.

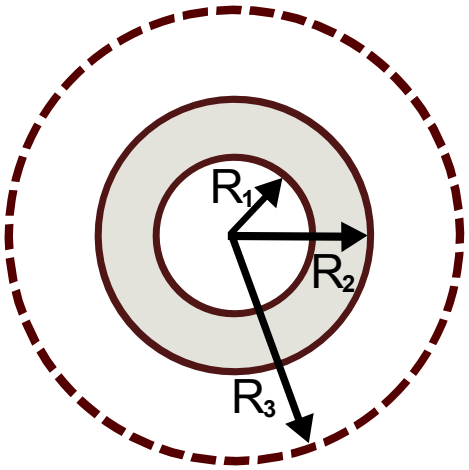

Figure 2.5: 2-dimensional cross-section of the proxy. Proxy regions are defined by $R_1 < R_2 < R_3$.

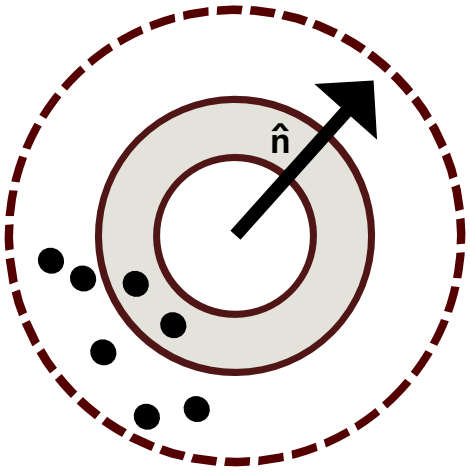

Figure 2.6: Estimation of surface normal.

2.2.2 Selection of Points for Surface Normal Estimation

Having defined regions of points around the proxy, we now consider how to move the proxy in virtual 3-space. The surface represented by the points needs to be locally estimated in order to constrain the proxy movement while in contact, so that only movements perpendicular to (or away from) the estimated surface are allowed. To do the surface normal estimation, one could simply search the entire point cloud. However, this requires excessive computation and slows down the haptic rendering. Instead we will select a subset of points (in Cartesian coordinates) containing only points in the proximity of the proxy.

This point selection is done by performing the following three steps on every Kinect frame (that is, 30 times per second).

1. Create a bounding cube with sides of length L around the proxy (in the Cartesian coordinate system) and transform this cube to the Kinect's coordinate system.

2. The transformed cube can now be projected onto the depth data matrix $M(u,v)$. Then, select $\{\mathbf{P_{sub}}\}_K$ as those points in the $M(u,v)$ within the boundaries of the projected cube.

3. Finally, create the set $\{\mathbf{P_{sub}}\}_C$ by transforming all the points in $\{\mathbf{P_{sub}}\}_K$ to the Cartesian

coordinate system.

This method of selection will reduce the number of points used for surface normal estimation.

2.2.3 Surface Normal Estimation

The haptic rendering process is carried out much more faster ($1kHz$) than the Kinect frame rate. During each cycle ($1ms$) the proxy is moved iteratively in steps of size $\delta > 0$. The number of steps taken in 1 ms depends on the speed of the computer.

Initially the proxy and the HIP coincide and the proxy is in free motion. The surface normal $\hat{\mathbf{n}}$ is estimated at each iteration as follows:

1. Define $\{\mathbf{P_k}\}, k = 1, 2, ..., N$, as the points in the set $\{\mathbf{P_{sub}}\}_C$ within radius R_3 of the proxy.

2.

$$\hat{\mathbf{n}} = \frac{1}{N} \sum_{k=1}^{N} \frac{\mathbf{X_{proxy}} - \mathbf{P_k}}{||\mathbf{X_{proxy}} - \mathbf{P_k}||} \tag{2.1}$$

2.2.4 Proxy Movement Algorithm

The Proxy Movement Algorithm moves the proxy such that it tracks the HIP with respect to the estimated surface defined by $\hat{n}$. The algorithm is implemented using the following steps. Here $\mathbf{v}$ denotes the vector between the proxy and the HIP.

1. If the proxy is in free motion (as described in Section 2.2.1), move the proxy one step of length δ along $\mathbf{v}$ (the direction of the HIP).

2. If the proxy is entrenched, move one step in the direction of $\hat{\mathbf{n}}$ (see Fig. 2.6).

3. If the proxy is in contact or entrenched and the HIP is outside of the estimated surface, move one step along $\mathbf{v}$ (see Fig. 2.7).

4. If the proxy is in contact or entrenched and the HIP is inside the estimated surface, project $\mathbf{v}$ on the plane defined by $\hat{\mathbf{n}}$ ($\mathbf{v_{plane}}$). Then move one step in this direction (see Fig. 2.8).

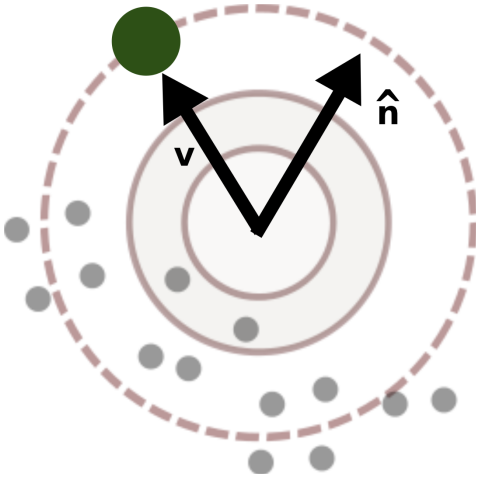

Figure 2.7: The proxy is in contact. Since the HIP is moving out of the surface the proxy will move in the direction of **v**.

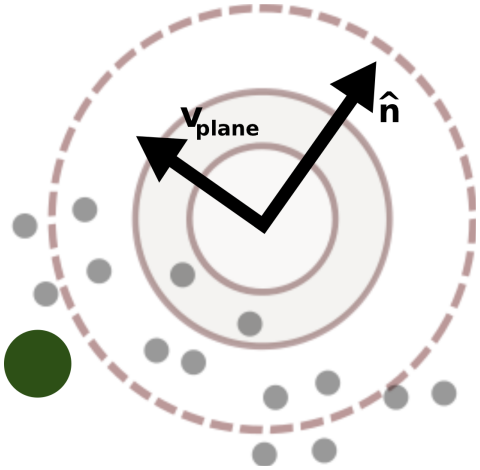

Figure 2.8: The proxy is in contact and the HIP is inside the estimated surface. In this case the proxy will move in the direction of $\mathbf{v_{plane}}$.

5. Iterate steps 1 to 5.

This iteration is continued until the end of current haptic cycle ($1ms$). This gives the proxy position for the next haptic cycle.

2.2.5 Force Rendering

The resulting proxy position is used to render the force to the user. A virtual spring is attached between the HIP and the midpoint between R_1 and R_2 (see Fig 2.9).

The force acts only on the HIP and only when the HIP is outside the midpoint between R_1 and R_2 as given by Eq. 2.2.

$$\mathbf{F_{HIP}} = -\frac{\mathbf{v}}{||\mathbf{v}||} * K_{Spring}(||\mathbf{v}|| - \frac{R_1 + R_2}{2}) \tag{2.2}$$

Fig. 2.9 is theoretically correct but in practice the computations are in discrete time for this marginally stable system. An artifact of the discretization, since the spring has no accompanying damping, is a jittery rendering of force during contact [40]. This can be compensated for by adding a damping term *during contact*, denoted by B, to the force calculation. Essentially this damping ties the HIP to the reference frame.

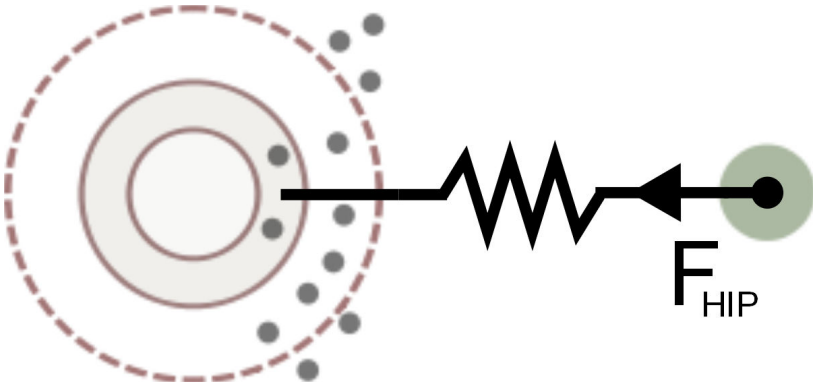

Figure 2.9: Illustration of force acting on the HIP. Note that the left side of the virtual spring is attached to the midpoint between R_1 and R_2.

2.3 Experimental Evaluation

In this section, we demonstrate the performance of the haptic rendering algorithm described above.

2.3.1 The Setup

The haptic rendering is evaluated using an Xbox Kinect depth camera, a Phantom Omni (Sensable Inc.) haptic device and two desktop PC's (see Fig. 2.10).

The *Virtual Environment* consists of software developed to perform the haptic rendering and the graphic visualization that runs on a desktop computer (Intel Core 2 Quad, 4 GB RAM, Nvidia Geforce 7100) with Ubuntu 10.10. The other computer is connected to the Virtual Environment through a network connection.

All the points in the matrix $M(u,v)$ are transformed (as in Section 2.1.3) and visualized in the Virtual Environment. Only points selected in Section 2.2.2 are transformed and used in the haptic rendering. The transformation in the first case is performed on the graphics card while the latter is performed on the CPU.

2.3.2 Evaluation of Haptic Rendering

The proxy in Fig. 2.11 has a radius of $5mm$ and moves in steps of $\delta = 1mm$. Table 2.1 contains a list of parameters used.

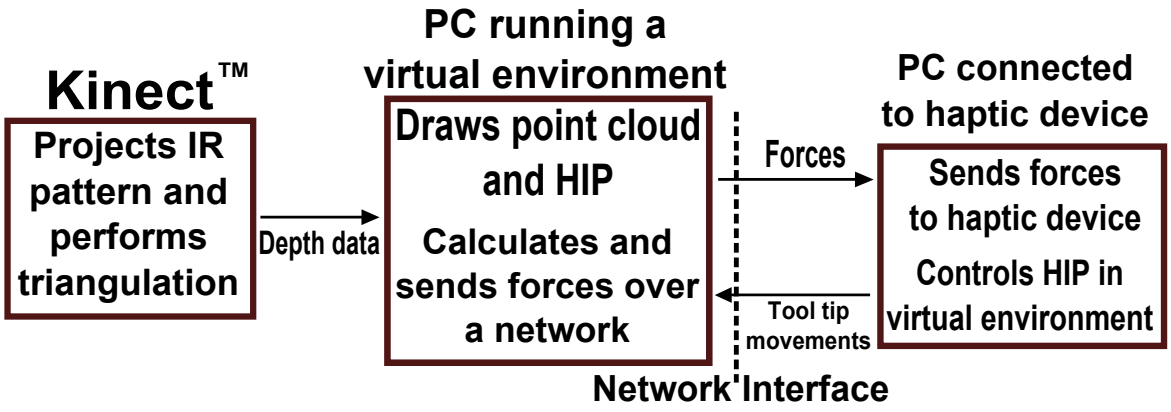

Figure 2.10: Experimental setup.

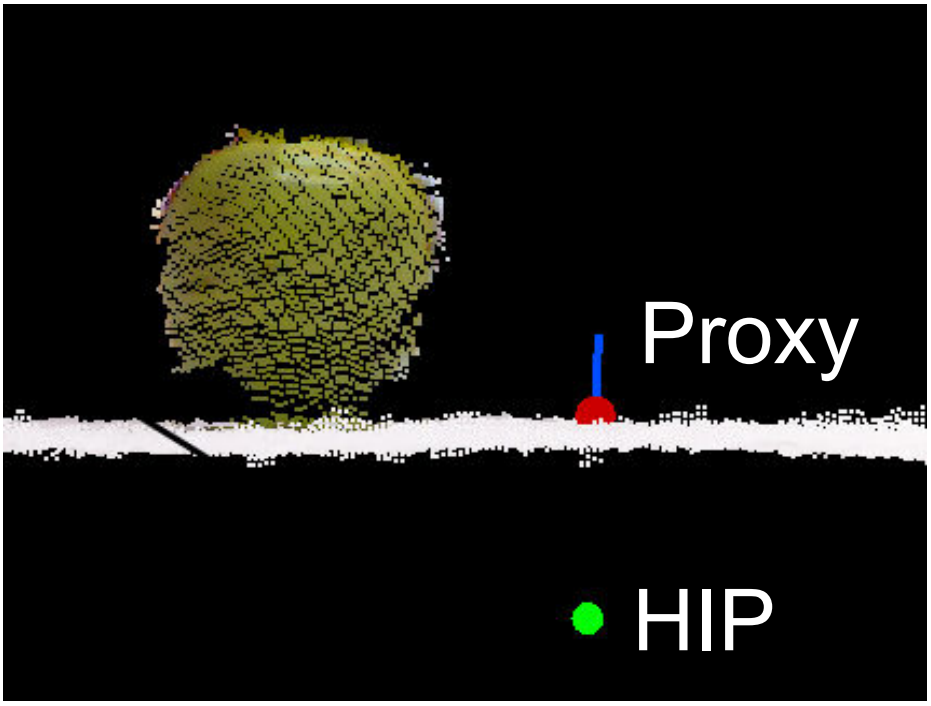

Figure 2.11: The proxy and the HIP next to a point representation of an apple (width $5cm$). The pin sticking out of the proxy indicates the estimated normal $\hat{n}$.

In a typical haptic scene the Proxy Movement Algorithm runs between $10kHz$ and $100kHz$ with an average of $80kHz$. It used roughly 0.5% of the point cloud (about 1500 points).

If all points in the point cloud were used for surface normal estimation (instead of just a subset

as described in Section 2.2.2) the rate of the Proxy Movement Algorithm drops to $200Hz$.

This 400-fold improvement in the Proxy Movement Algorithm allows us to achieve the haptic rendering rate of $1kHz$ recommended in [26].

Table 2.1: Parameters used in Experimental Evaluation

R_1	$5mm$
R_2	$5.1mm$
R_3	$10mm$
Proxy Movement Step Size (δ)	$1mm$
Bounding Cube Side Length ($L = R_2 * 4$)	$20.4mm$
Spring Constant (K_{Spring})	$350N/m$
Damping Constant (B)	$3Ns/m$

2.4 Results

Evaluation of the accuracy and the correctness of a haptic rendering that is obtained from point clouds derived from a camera is not a well-posed problem, since the true forces that represent physical contact can not be defined.

Fig. 2.12 compares the position of the proxy developed in this chapter with the HIP as the user moves the HIP into a virtual surface. The HIP penetrates the surface but the proxy does not. That is, the "pop-through" problem for point cloud haptic rendering does not arise for objects that are not moving. The bottom trace of Fig. 2.12 shows the force applied to the user. The three peaks are caused by the user trying to push through the surface.

The movement of proxy at a very high rate causes noisy force signals. In order to suppress this noise a 26 point FIR lowpass filter is applied to the force signal. Another source of noise is the noise on the depth signal with a standard deviation of $0.82mm$ (at $0.45m$ depth).

The haptic rendering algorithm developed here allows interaction with convex and concave surfaces as well as sharp corners (see Fig. 2.13). Fig. 2.14 shows the y-position of the proxy while it

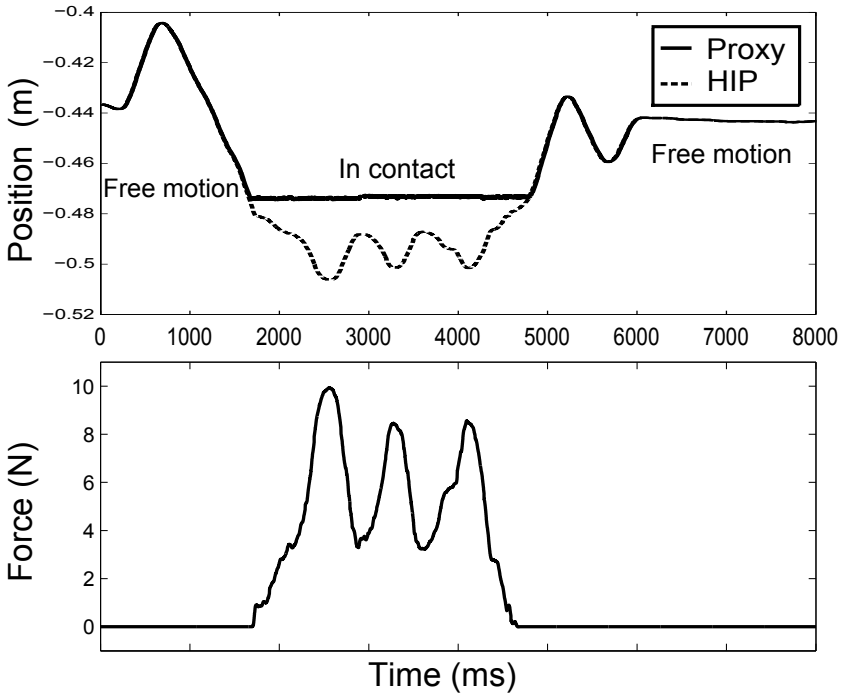

Figure 2.12: The user moves the HIP towards a surface, continues to push into the surface and then moves out of the surface. The bottom trace shows forces applied to the user. Only the y-value of position and force is illustrated.

is moving off a 5*mm* ledge.

This approach to haptic rendering allows interaction with moving point clouds. To illustrate this, the HIP was kept at a fixed point while a flat surface was moved in the positive y-direction (up) in front of the Kinect, forcing the proxy away from the HIP. The distance between the proxy and the HIP was measured continuously. An average velocity of $0.08m/s$ was achieved before the proxy popped-through the object (see Fig. 2.15).

By extending the proxy method, we developed and implemented a novel method for haptic rendering from time varying point clouds where surface normals are unknown. The ability to remotely feel objects in real-time using non-contact devices will be used in the following chapter to create haptic virtual fixtures for teleoperation.

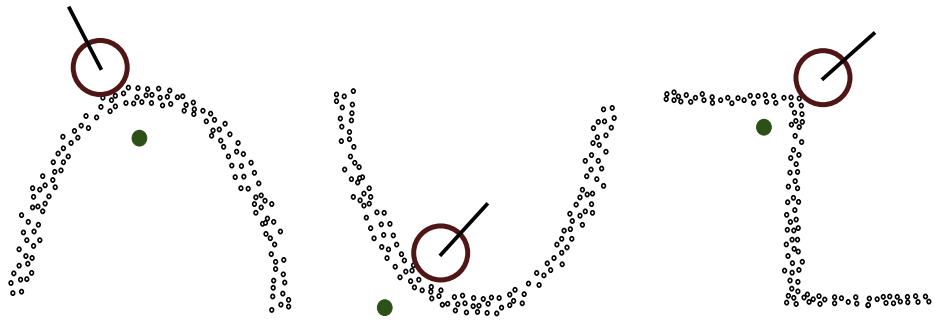

Figure 2.13: **Left:** Proxy on a concave surface. **Middle:** Proxy on a convex surface **Right:** Proxy on a sharp corner

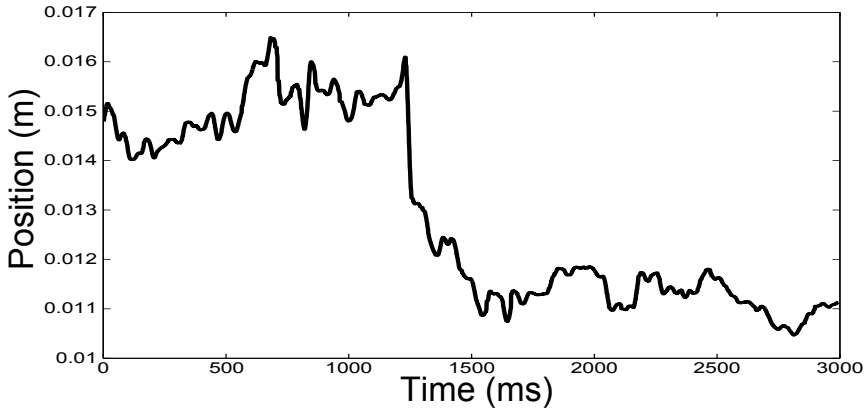

Figure 2.14: The y-position of the proxy as it moves off a 5*mm* ledge.

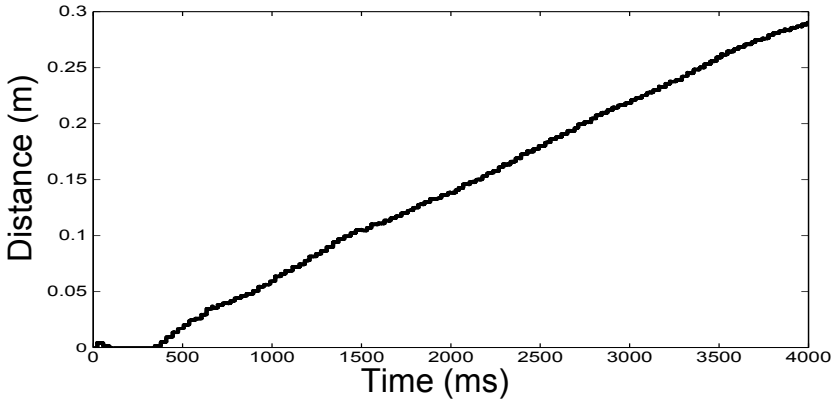

Figure 2.15: The distance between the proxy and the HIP as the flat surface is moved up in front of the camera, forcing the proxy away from the HIP.

Chapter 3

HAPTIC VIRTUAL FIXTURES

Building on the algorithms described in Chapter 2, we propose a method for construction of forbidden region virtual fixtures with haptic feedback from streaming point clouds. The objective is to provide assistance to the operator by preventing the motion of the robot in the forbidden region. The effectiveness of the method will be demonstrated in teleoperation using a surgical robot.

3.1 Background and Literature Review

The term virtual fixture was first introduced in [1] as "abstract sensory information overlaid on top of reflected sensory feedback from a remote environment." According to this general definition, the abstract sensory information can refer to all forms of feedback including haptic, visual, auditory or a combination of these modalities, e.g. haptic-auditory [1], visual-auditory [41] and haptic-visual [42]. Virtual fixtures were shown to increase the performance for robot positioning tasks [43],[1] and path following [44].

Virtual fixtures (also referred to as "active constraints", "surgical macros", "virtual surfaces" and "haptically augmented teleoperation") can further be categorized according to their function. Guidance virtual fixtures constrain the motion of the robot to a desired path or surface. Guidance virtual fixtures were implemented in [45] to direct the robot toward a specific point or a curve. The effect of virtual fixtures on user performance under guidance was investigated in [46]. A haptic-visual guidance virtual fixture was shown to improve the performance of a simulated trans-apical aortic valve replacement in [42]. In [47] a surgical robot was constrained to a path inside a human skull phantom resembling the path followed in an endoscopic sinus surgery. Using hidden Markov Models guidance virtual fixtures for microsurgical applications with automatic recognition of operator activities were developed in [48].

In contrast to guidance virtual fixtures, Forbidden Region Virtual Fixtures (FRVF) prevent the robot from entering certain regions in space without affecting motion elsewhere. In [49], a virtual

wall constrained an instrument's motion to assist with the dissection of the interior mammary artery during a minimally invasive coronary artery bypass (CABG) procedure. In [50] a safe region for cutting the bone during unicondylar knee arthroplasty was defined pre-operatively and commands to cut the bone in the forbidden region were refused by the robot. A virtual fixture for assembly tasks was implemented in [51] allowing the operator to slide massive payloads on the virtual surface with less effort.

Virtual fixtures have been employed for both telemanipulation and cooperative manipulation (e.g. [50],[52],[46]). In telemanipulation the operator controls the motion of the remote robot in the environment with the aid of a master robotic device[1]. There is no mechanical linkage between the operator and the remote robot. In cooperative manipulation, the master station is eliminated and the operator directly manipulates the robot in the environment. An application of this type of manipulation is in ophthalmic micro-surgery where it is desired to keep the surgical tool at a fixed safe distance from the retinal surface. Guidance virtual fixtures were implemented in [53] by constraining the motion of the tool, cooperatively held by the robot and the operator, to a concave surface simulating the retinal surface.

Virtual fixtures are typically specified by the operator. Both guidance and forbidden region virtual fixtures have been constructed in a variety of shapes. Forbidden region virtual walls in the form of planes were used in [1] for a peg-in-the-hole task and in [49] for a blunt dissection task. In [54] the virtual fixtures were generated for a given geometric constraint by forming a quadratic optimization problem with linear constraints.

Recently, there has been an interest in constructing virtual fixtures from physical objects or features. For instance, a virtual fixture obtained from medical images can protect an anatomical structure from unwanted damage by the robot. Such virtual fixtures have been generated using RGB images [45], Computed Tomography (CT) scans [47], [50], X-ray fluoroscopy [55], Magnetic Resonance Imaging (MRI) [42] as well as images from a stereo camera [56].

A 1 DOF forbidden region virtual fixture to protect a moving phantom tissue was implemented in [57]. The tissue phantom was mounted on a linear stage and its location was constantly measured

[1]In the teleoperation literature, the remote robot is sometime referred to as the robot or slave robot. Throughout this chapter the term robot refers to the remote robot unless otherwise specified.

by encoders and used to update the location of the virtual fixture. In practice, measurements of the coordinates of objects in space are not available to track their motions. Computer vision was used in [45] to implement virtual fixtures for point positioning and path following. Vision based virtual fixtures generation is typically performed in two steps: 1) obtaining a 3D model of the fixture and 2) registering the model to the robot and/or the patient. In [47] a virtual fixture of patient's skull was generated from CT scans. The 3D model of the skull was obtained pre-operatively and registered to the patient intra-operatively in real-time. This method is effective for rigid anatomies with non-moving parts such as the skull. However, for structures that are deformable or have moving parts, the 3D model must be reconstructed in real-time to reflect motions and deformations.

A method of generating dynamic forbidden region and guidance virtual fixtures for beating heart procedures was proposed in [58]. Virtual fixtures were constructed pre-operatively from MR/CT images and registered to the patient using intra-operative ultrasound images. To account for the dynamic nature of the heart, several 3D images depicting the heart at different time points in the cardiac cycle were taken and a library of virtual fixtures was constructed. The appropriate virtual fixture was then registered to the patient both spatially and temporally. This method works when the geometry of the heart matches that of the recorded cycle, but it is not readily applicable to deformable objects with arbitrary motions and deformations.

For a general-purpose implementation of moving virtual fixtures, both 3D modeling and registration must be performed in real-time to reflect changes in the physical world. Real-time MRI was proposed in [42] for generating guidance VF for robot-assisted aortic valve replacement. This approach limits the materials that can be present in the environment and requires the operation of an MRI compatible robot. RGB-D cameras and laser scanners that generate point clouds are commercially available at relatively low cost, impose minimal restrictions on the composing material of the objects and can provide streaming (30Hz or above) point cloud representations capturing motions and deformations. In this chapter, we propose a method for real-time generation of forbidden region virtual fixtures with haptic feedback from such streaming point clouds.

The closest work to our approach to virtual fixtures in the literature is [56] where a method for generating virtual fixtures from static point clouds captured by a stereo-vision camera is presented. In that work, the operator selects a subset of the points in the point cloud which undergoes a series of offline preprocessing steps including downsampling, filtering, normal estimation and offsetting

to generate a smooth forbidden region virtual fixture. Due to the offline nature and computational cost of these steps, this method is not applicable to streaming point clouds. Our work eliminates the need for offline preprocessing of the point cloud and therefore can be applied in real-time. To the best of our knowledge this is the first implementation of virtual fixtures from streaming point clouds.

3.2 System Description

The system consists of the master side, the remote side and the software components. We refer to the master side environment as the *proximal world* due to its proximity to the operator. Similarly the remote environment is referred to as *remote world* and its representation in software is referred to as the *virtual world*. These components along with the data flow between them are shown in a block diagram representation in Fig. 3.1.

The operator resides at the master side and interacts with the remote environment with the aid of the master console. The master console has means of capturing operator's commanded positions (X_m) and displaying visual feedback from the environment. The visual feedback can contain overlaid visual cues generated by the software components, also referred to as augmented reality visual feedback. In addition to the visual feedback from the environment, for haptic virtual fixtures the master console must be capable of providing force feedback to the operator (F).

The remote side consists of at least a robot, a depth sensor and a camera. The depth sensor captures a 3D depth map of this environment in the form of a point cloud while the camera captures RGB images for visual feedback to the operator. In some devices, such as the Microsoft Kinect, the depth sensor and the camera are integrated in one package. The remote side generates the greater part of the data required for the implementation of virtual fixtures including the data from the robot and depth sensor. Therefore, computations are typically performed at the remote side and the communication with the master is done over a network as shown in Fig. 3.1. Other arrangements such as performing the computations on the master side or a cloud service are possible but would result greater network bandwidth requirements.

Data from the master side and remote side are processed by the software components. For the point cloud data to be useful it must be registered in the robot frame. This is performed by

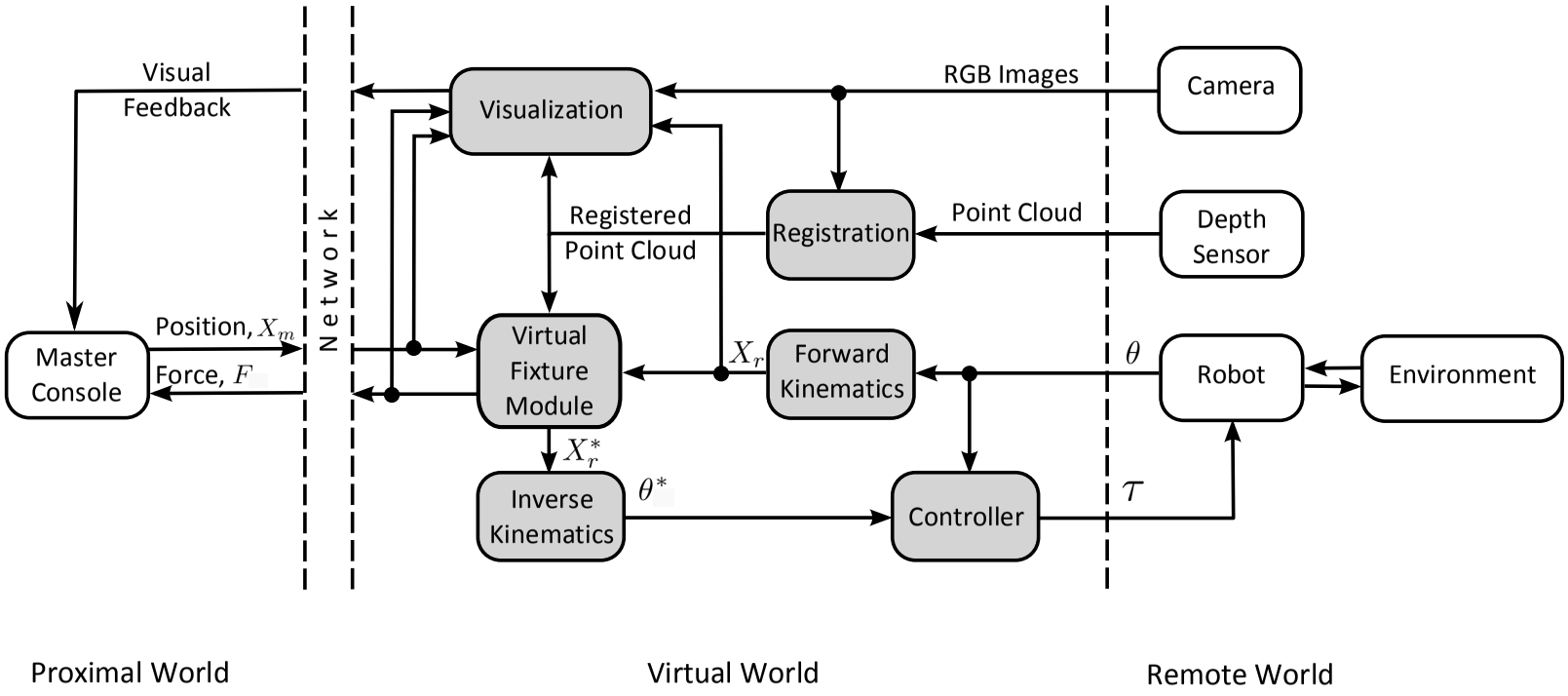

Figure 3.1: Block diagram representation of the system. Shaded blocks represent software components and clear blocks represent hardware elements. The subscripts *m* and *r* denote the master and robot quantities. Commanded (desired) values are indicated by an asterisk, *.

the registration module, which constructs a transformation between the point cloud and the robot. Measurements from the robot are typically in the form of joint angles, i.e. θ. These angles are converted to end effector coordinates, X_r, by the forward kinematics module. The registered point cloud data, the robot end effector coordinates and the master device commands X_m are employed by the virtual fixture module to compute the force feedback to the operator. The other output of this module is the desired position of the robot, X_r^*, which is first converted to desired joint angles, θ^*, according to the inverse kinematics and then translated to motor torque commands by the controller.

The methods proposed in this chapter are implemented on an impedance type haptic device, i.e. devices that measure position and display force. The block diagram in Fig. 3.1 shows one depth sensor and one robot. In order to capture the environment from different angles and construct an improved 3D model, it is desirable to have multiple depth sensors. In addition, having multiple robots can enhance dexterity (e.g. two arms acting as the two arms of a surgeon). The architecture proposed in this chapter is fully object-oriented and can be extended to support multiple depth sensors and robots. The following three sections describe the hardware and software components employed in this work along with the proposed methods and algorithms for the generation of virtual fixtures.

3.3 Remote World

3.3.1 Robot

The Raven II™ surgical system [59], developed in a collaborative effort between the University of Washington (UW) and the University of California at Santa Cruz (UCSC), is an open platform designed to support research in robot assisted surgery. Each arm of the Raven II system consists of three main components: a surgical instrument, a spherical mechanism for positioning the instrument, and a stationary base supporting the spherical mechanism and the actuators. A total of seven DC motors actuate the seven joints of each arm via cable couplings. These joints are: shoulder joint, elbow joint, tool insertion, tool roll, tool wrist, and two grasping joints. Except for the prismatic tool insertion joint, all joints are rotational. Each motor is equipped with an optical rotary encoder for measuring the angle of rotation. Assuming negligible deformations in the cables, these measurements are used in place of the robot's joint angles in forward and inverse kinematics.

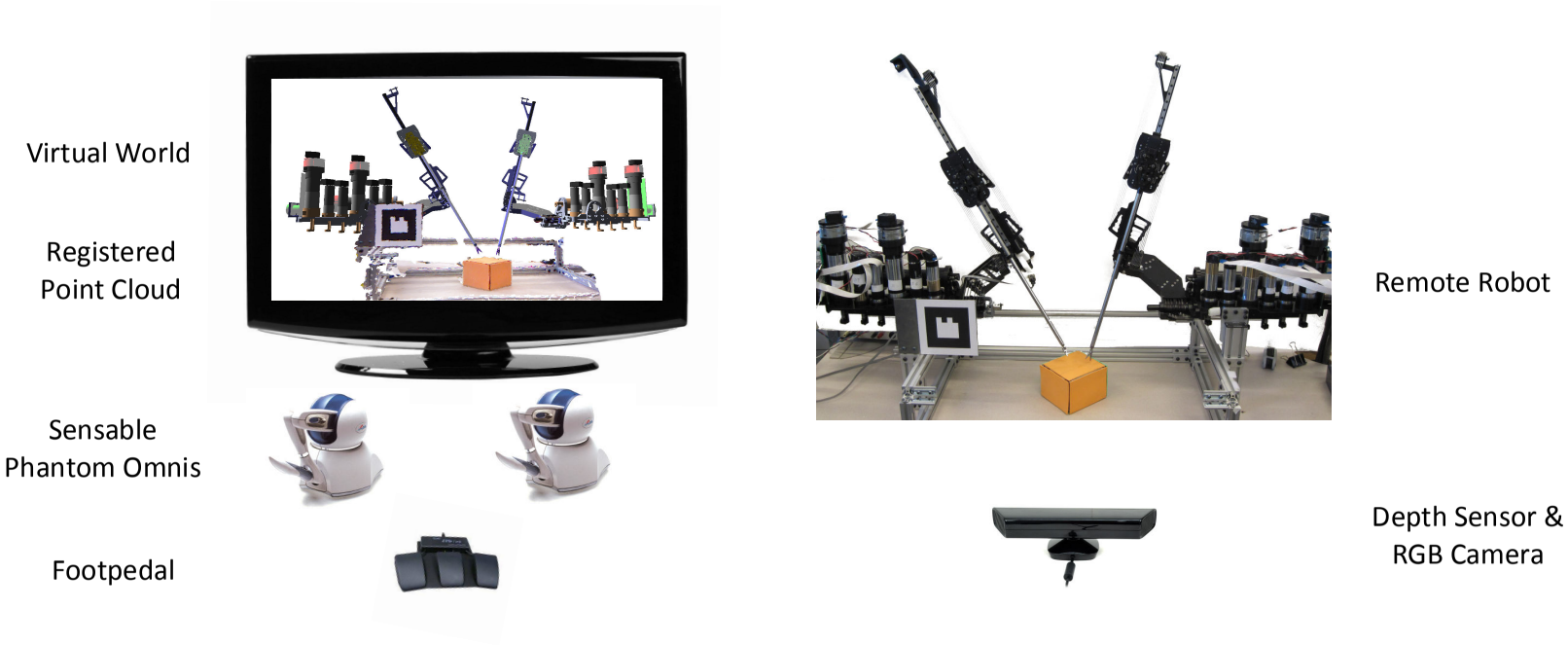

Figure 3.2: Components of the virtual fixture system.

3.3.2 Depth Sensor and Camera

The methods proposed in this chapter are applicable to all point cloud streams from a variety of depth sensors including projected pattern and time of flight sensors, stereo cameras and laser scanners. We demonstrate the proposed methods on streaming point clouds obtained by the PrimeSense Carmine sensor. Carmine operates on the same principle as the Microsoft Kinect but has an improved close range performance. In addition to the depth sensor, Carmine and Kinect are both equipped with a color camera which adds RGB values to each point. The RGB data can be used for registration, segmentation and visualization. However, RGB data are not required for the construction of virtual fixtures.

3.4 Proximal World

3.4.1 Master Console

The master console consists of two Sensable Phantom® Omni haptic devices, a foot-pedal and a display. The Omnis are held by the operator, each capturing six degrees of freedom of hand motion. The Omnis are capable of providing active force feedback on the three translational degrees of freedom. Therefore, only the translational degrees of freedom are used to control the robot and the orientation measurements are discarded. The foot-pedal enables the control of the robot arms by the operator. Upon releasing the pedal the robot enters the idle state. This feature allows for the mapping of the limited workspace of the haptic device to the larger workspace of the robot. Upon reaching the workspace limit, the operator can release the foot-pedal, reposition the master device and continue teleoperating from there. This is also referred to as indexing and is in particular useful when motions are scaled down for more precise positioning.

Communication over a network is subject to delay, jitter and packet loss. Teleoperation requires fast and reliable communication between the master console and the remote robot. The Interoperable Teleoperation Protocol (ITP) [60] is a flexible communication protocol designed specifically to address the challenges of teleoperation over a network. ITP minimizes the overhead by taking advantage of the UDP data structure. In addition, employing motion increments in place of absolute positions and orientations ITP increases the robustness and safety of teleoperation under network delay and packet loss.

A packet rate of $1kHz$ or greater is essential for stable haptic rendering [26]. ITP was originally designed for unilateral teleoperation. In this work the protocol was modified to allow the computed forces from the virtual fixture module to be sent back to the master station. The master and remote stations were connected with a gigabit switch. As a result the communication delay was negligible, and hence was not considered. The effectiveness of virtual fixtures in teleoperation under time delay was also considered in [61].

3.5 Software Components

3.5.1 Framework

Several software modules are involved in generating the virtual fixtures. These modules include the image registration module, camera drivers, forward and inverse kinematics, visualization and robot control. The Robot Operating System (ROS) [62] is the software framework employed in this work. ROS provides a communication layer for the software components and allows the modules to be developed independently. Its peer-to-peer structure allows the computation-expensive tasks to run on a heterogeneous computer cluster, increasing the computational capacity with minimum effort. In addition, the growing library of open source packages supported by ROS facilitates the software integration. These features makes ROS an ideal framework for developing virtual fixture software.

3.5.2 Registration

In generating virtual fixtures from physical objects, accurate positioning of the point cloud with respect to the robot is essential in achieving robustness to the motion of the camera and the environment. To this end, the point cloud must be registered to the robot frame. Ideally, the camera is mounted on a robotic arm which provides its position and orientation with respect to the robot's base. In the absence of such an arm, visual markers can be placed at fixed positions and orientations with respect to the robot's base frame as shown in Fig. 3.2. Software packages are available for tracking markers. ARToolKit [63] is an open source software library for calculating the transformation between the camera and the marker. A ROS wrapper for AR_ToolKit (i.e. AR_Kinect) is used in this work; it enhances marker localization using point cloud data from Kinect.

The ROS tf package is a convenient way of dealing with various transformations in a system.

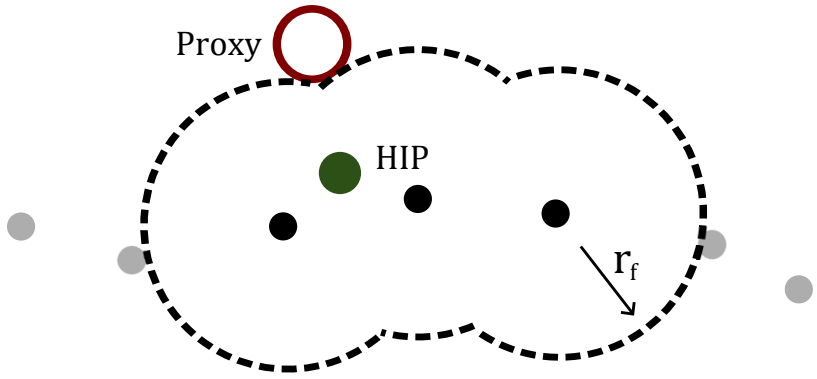

Figure 3.3: A 2D illustration of the haptic rendering method. The black points are selected and assigned a forbidden-region radius r_f. In this case, when the HIP entered the forbidden region from the top, the proxy position corresponds to the local minimum of the distance between the proxy and the HIP.

By constructing a transformation tree which determines the translation and rotation offsets between different coordinate frames, the tf package makes it possible to obtain the transformation between any two frames in the tree at any given time. A low pass filter was implemented in this work to attenuate the noise in the calculated transformation. The combination of AR_Kinect and tf packages provides a robust solution to registration problem and eliminates the need for a fixed camera.

3.5.3 Virtual Fixture Module

The main objective of the virtual fixture module is to keep the robot from making contact with protected regions while allowing it to move freely elsewhere. Haptic virtual fixtures achieve this goal by generating opposing forces in the haptic rendering process. In this work, we implement the haptic rendering method presented in [64]. This is an extension of the haptic rendering method presented in Chapter 2, where the virtual fixtures were defined implicitly as spherical constraints around the points in a point cloud with radius r_f as shown in Fig. 3.3.

The haptic rendering algorithm has two outputs. 1) the interaction forces, and 2) the position of the proxy which represents the ideal point of interaction. Various FRVF architectures are possible depending on the way these outputs are employed and the robot is teleoperated. In [65] four teleoperation architectures (i.e. one unilateral and three bilateral teleoperation architectures) and various impedance-type FRVF architectures were investigated in a one degree-of-freedom experi-

ment. Tracking, safety and submittance (the ability to move the robot to any position outside the forbidden region) were three evaluation metrics employed in the experiments. It was concluded that a single best architecture does not exist and the choice of architecture should be made according to the application, due to the inherent tradeoff between the metrics.

In this work we present three impedance-type FRVF architectures for teleoperation and qualitatively compare the advantages and disadvantages of each architecture. Since most existing surgical and underwater teleoperation systems are unilateral, the focus of this chapter is on evaluating forbidden region virtual fixtures for such systems. In unilateral teleoperation FRVF's the only force on the master is in response to violating the forbidden region, and not due to its coupling to the robot or measured environment forces, as is the case in bilateral teleoperation. Nevertheless, the architectures presented can be easily adapted for bilateral teleoperation.

As mentioned in Chapter 2, the force of the master device in all architectures is proportional to the distance between the proxy and the HIP, i.e. $\mathbf{F} = k(X_{Proxy} - X_{HIP})$. The distinction between the architectures arises from the choice of 1) HIP and 2) the robot position command. In the rest of this section X_m refers to the position of the master device and X_r and X_r^* respectively refer to the actual and commanded position of the remote robot.

Architecture I: Master Controlled Robot - Master Controlled HIP

$$\begin{aligned} X_{HIP} &\leftarrow X_m \\ X_r^* &\leftarrow X_m \end{aligned} \tag{3.1}$$

In this architecture, the HIP is set to the master device as shown in Fig 3.4.a. As a result, the haptic rendering is carried out using the master position i.e. $\mathbf{F} = k_s(X_{Proxy} - X_m)$. As soon as the master violates the forbidden region, an opposing force is calculated and applied to the user. The robot also servos to the master through an independent pathway. In other words, the master is teleoperating the robot while interacting with the virtual environment created by the point cloud. There is no direct connection between the haptic rendering process and the teleoperation process. This makes it possible to disable each process on demand. For instance, in order to gain a better understanding of the environment the user can disable the teleoperation and safely feel the virtual environment

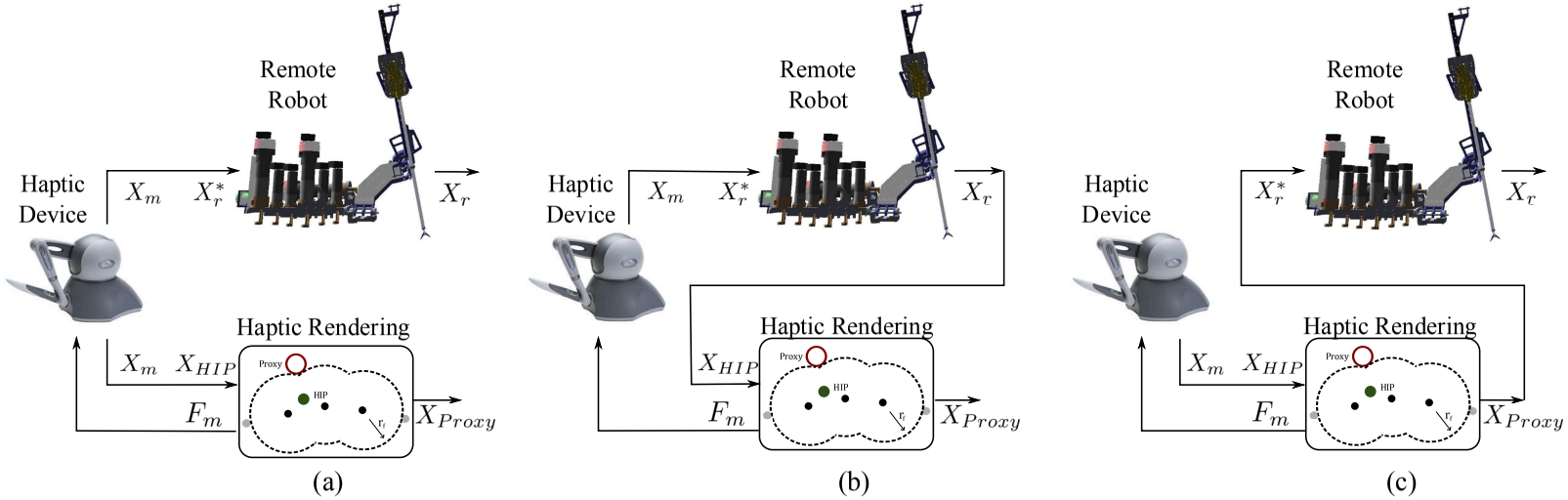

Figure 3.4: Block Diagram of the three virtual fixture architectures. a) Architecture I: Master Controlled Robot - Master Controlled HIP. The force is based on master position. Robot servos to the master. b) Architecture II: Master Controlled Robot - Robot Controlled HIP. The force is based on robot position. Robot servos to the master. c) Architecture III: Proxy Controlled Robot - Master Controlled HIP. The force is based on master position. Robot servos to the proxy.

while the robot remains stationary. It is also possible to engage and disengage the haptic rendering process at any point in time, for example to free up computational resources for other tasks such as path planning.

As Fig. 3.5.a shows, the robot is coupled to the master in this architecture. As a result, despite the opposing force that is applied to the user as soon as a violating command is generated, it is possible to command the robot into the forbidden region by overcoming the applied force. Due to this property, the virtual fixture generated by this architecture belongs to the class of soft virtual fixtures [66], in which motion in the forbidden region is opposed but possible. In contrast, hard virtual fixtures display zero compliance and do not allow any violation of the constraint.

Recall that the haptic rendering is performed on the master position with the robot servoing to that position. Subsequently, this has the advantage of instantly penalizing commands inside the forbidden region even before they are executed. However, if for any reason, such as the presence of a disturbance or an imperfect control action, the robot deviates from the commanded position then the rendered forces can be inaccurate. For example, a force will be rendered when the robot is not violating the forbidden region but the master is (see Fig. 3.5.a). Similarly, no force will be rendered when the robot is violating the constraint but the master is not.

Architecture II: Master Controlled Robot - Robot Controlled HIP

$$\begin{aligned} X_{HIP} &\leftarrow X_r \\ X_r^* &\leftarrow X_m \end{aligned} \tag{3.2}$$

The architecture presented in Fig. 3.4.b performs the haptic rendering on the robot position i.e. $\mathbf{F} = k_s(X_{Proxy} - X_r)$. Therefore, in contrast to architecture I, an opposing force is only rendered when the robot violates the constraint. This overcomes the problem of inaccurate force rendering due to tracking discrepancy faced by architecture I. It must be noted that if the controller exhibits perfect tracking, the two architectures would be identical.

In comparison to the first architecture, a larger closed loop delay is observed in this architecture. This can be better understood by comparing the path from the master device position, X_m, to the rendered force, F, in Fig. 3.4. In architecture II, the controller and the robot are placed in the

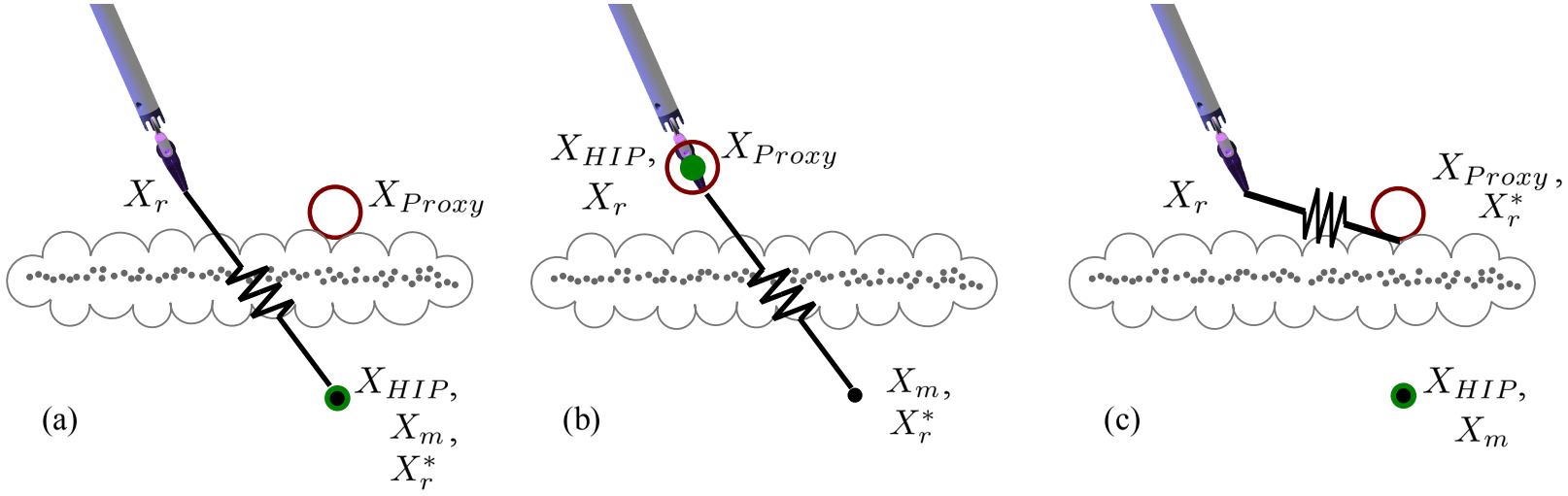

Figure 3.5: Couplings in the three architectures. a) Architecture I: Master Controlled Robot - Master Controlled HIP. HIP is in contact with the forbidden region and a force is rendered to the user. b) Architecture II: Master Controlled Robot - Robot Controlled HIP. HIP is in free motion and no force is rendered until the robot enters the forbidden region. c) Architecture III: Proxy Controlled Robot - Master Controlled HIP. HIP is in contact with the forbidden region and a force is rendered to the user.

force rendering loop between the master and the haptic rendering module. The increased delay can degrade the performance and stability of the virtual fixture. It must be noted that for a teleoperation system with perfect tracking, i.e. $X_m = X_r$, the two architectures become identical.

Similar to architecture I, the robot is coupled to the master, as shown in Fig. 3.5.b. Therefore, the user can override the virtual fixture by overcoming the applied force, making this architecture a soft vitural fixture.

Architecture III: Proxy Controlled Robot - Master Controlled HIP

$$\begin{aligned} X_{HIP} &\leftarrow X_m \\ X_r^* &\leftarrow X_{Proxy} \end{aligned} \tag{3.3}$$

The third architecture is fundamentally different from the first two in that the robot is coupled to the proxy instead of the master, as shown in Fig. 3.5.c. Recall that the proxy is a virtual sphere that never violates the constraints. Assuming negligible tracking error, the robot will not enter the forbidden region even if the user overcomes the rendered forces. Therefore, in contrast to the first two architectures, architecture III constructs a hard virtual fixture without the need for a large spring constant that would adversely affect the stability of teleoperation.

The proxy can undergo discontinuous jumps when falling off an edge, as shown in Fig. 3.6. This is a desirable feature in that it makes the rendering of edges more realistic by creating discontinuous forces. However, a discontinuity in the commanded position, X_r^*, can cause large jumps in the control action and may damage the robot and the environment. This problem can be solved by applying an upper limit on $|X_r^*|$ increments. This ensures the robot follows the proxy at a safe speed without sacrificing the realism of contact with the edges.

3.5.4 Kinematics and Control

The forward kinematics module calculates the robot end effector position, X_r, from joint encoder readings, θ. The inverse kinematics module converts the commanded position output, X_r^*, to joint commands that are executed by the controller. The forward kinematics was constructed from the

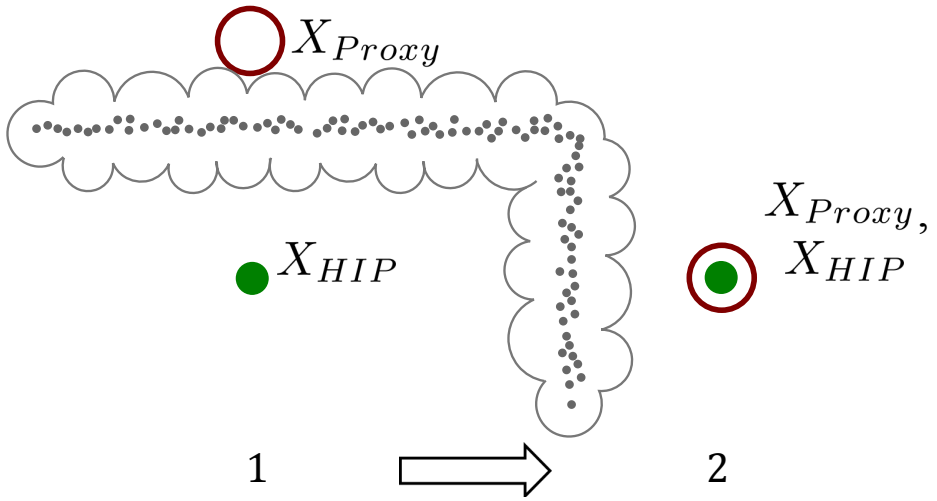

Figure 3.6: The proxy can have discontinuous motion in space when interacting with the edges.

DH parameters described in [59]. A closed form solution for the inverse kinematics was obtained from [67] and a PID controller was implemented on each joint at $1kHz$.

3.5.5 Visualization

The 3D visualization tool for ROS, i.e. rviz, provides a convenient way of presenting information to the user. Fig. 3.7.a shows the 2D view of the scene as captured by Kinect's RGB camera. The 3D point cloud view of the same scene is shown is Fig. 3.7.b. Using a 3D model of the robot and the transformations from the registration module, rviz can automatically overlay a virtual robot on top of the scene. In addition, virtual markers representing the proxy and force can be readily added to the scene. Aside from the point cloud view, the user has the option to view the RGB images.

3.6 Experiments

The effectiveness of the methods proposed in this chapter are demonstrated in a series of experiments with both static and moving environments. All experiments are carried out using the PrimeSense Carmine 1.09 close range camera. With the ability to capture point clouds of objects as close as $35cm$ from the camera, Carmine 1.09 offers a better close range performance compared to Kinect. RGB images were automatically overlaid on the point cloud in rviz by ROS *OpenNI* package. Both the RGB and IR cameras have been calibrated using the ROS *camera_calibration* package and a

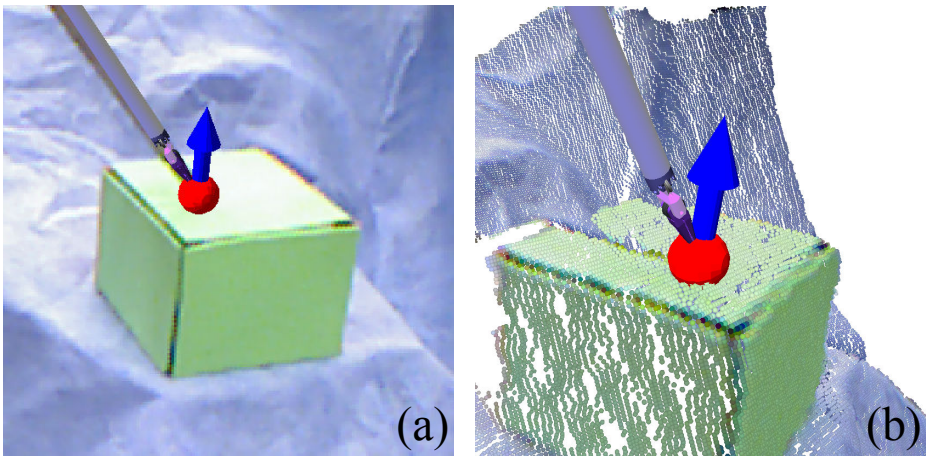

Figure 3.7: The virtual robot overlaid on (a) 2D Image captured by Kinect RGB camera, (b) 3D view of the scene (point cloud + RGB). The red sphere represents the proxy and the arrow represents the rendered force. The sphere has been enlarged for clarity and does not represent the true size of the proxy.

standard checkerboard. The calibration was essential in achieving the accuracy required for implementing virtual fixtures. The PID controller gains were hand tuned to achieve stable and accurate tracking performance. The spring constant for haptic rendering, k_s, was set to 800 N/m.

3.6.1 Architectures

A simple experiment was set up to demonstrate the performance of the three architectures. The robot end effector is positioned above an inclined flat surface as shown in Fig. 3.8.a. The master device is simulated by a computer program that generates a horizontal trajectory to bring the end effector closer to the surface. This trajectory is shown by a green line segment. The flat surface is segmented based on its color (i.e. white) and a virtual fixture with the radius of 2cm has been placed around the surface. In the absence of the virtual fixture, the robot will move closer to the surface and will violate the constraint.

Fig. 3.8.b-d respectively show the performance of architectures I through III in the point cloud view of the environment as seen in rviz. The robot end effector is not captured in the point cloud due to its small width and reflective material. However, a visualization of the robot is registered and positioned in the scene as described in section 3.7. It is important to note that this is not a simulation

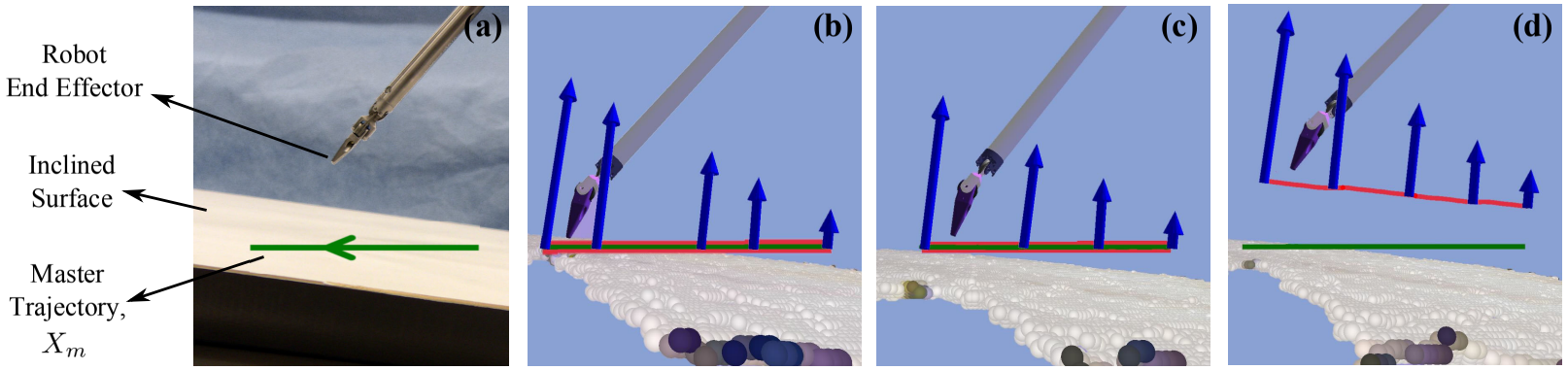

Figure 3.8: Performance of the three virtual fixture architectures for a computer generated trajectory. The green and red line segments display the position of the master, X_m, and the position of the robot, X_r, respectively. The blue arrows display the opposing force by the virtual fixture. (a) The robot end effector is moved horizontally towards an inclined surface. (b) Architecture I: The robot follows the master, i.e. soft virtual fixture. (c) Architecture II: Similar performance to architecture I due to small controller tracking error. (d) Architecture III: hard virtual fixture. The region in the bottom left corner of the images where points are absent from the point cloud is the shadow of the end effector.

of the robot, but a visualization of the actual position of the robot based on the its joint encoder readings.

As demonstrated by the blue arrows, in all three architectures the path towards the surface will generate an opposing force. The magnitude of the force increases as the master trajectory penetrates further inside the forbidden region. If this force was applied to the user by a haptic device, it would impede the violation of the constraint.

The red line segment displays the actual path of the robot. As Fig. 3.8.b and 3.8.c show, in architectures I and II, the robot is coupled to the master. Therefore, it can be driven into the forbidden region by overcoming the force. As discussed in section 3.5.3, architectures I and II implement a soft virtual fixture. In addition, due to small tracking error between X_r^* and X_r, architectures I and II do not demonstrate a significantly different performance. In contrast, architecture III (see Fig. 3.8.d) prevents the robot from violating the forbidden region regardless of the user's force, demonstrating a hard virtual fixture.

In order to demonstrate the effectiveness of the virtual fixture with a human in the loop, an experiment was designed in which the user tries to pop a balloon with a sharp tip attached to the robot end effector as shown in Fig. 3.9.a. The balloon was segmented based on its color and a virtual fixture of radius of $2cm$ was placed around every point of the balloon in the point cloud. The path of the robot along with the forces applied to the user are shown in Fig. 3.9.b for architecture III. Even though the user tries to pop the balloon with an increasing force, the tip is kept a safe distance away from the balloon. In contrast, architectures I and II resist the users motion into the forbidden region but allow the user to pop the balloon with the same action.

3.6.2 Dynamic Environment

A distinguishing feature of the methods proposed in this chapter is the ability to handle streaming point clouds from dynamic environments in real-time. A moving platform has been set up to evaluate the performance of the virtual fixture in protecting a dynamic object. The platform consists of a flat surface mounted on a rocking platform mixer that moves the surface through a range of rotations as shown in Fig. 3.10. A $2cm$ virtual fixture has been placed around every point in the point cloud belonging to the surface. Fig. 3.10 shows the position of the robot in architecture III as it moves

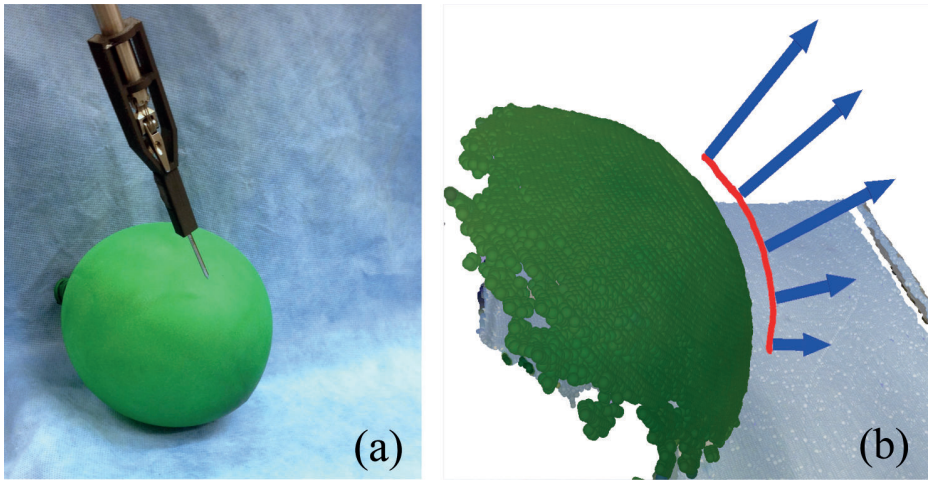

Figure 3.9: Performance of architecture III in protecting the balloon from a sharp end effector.

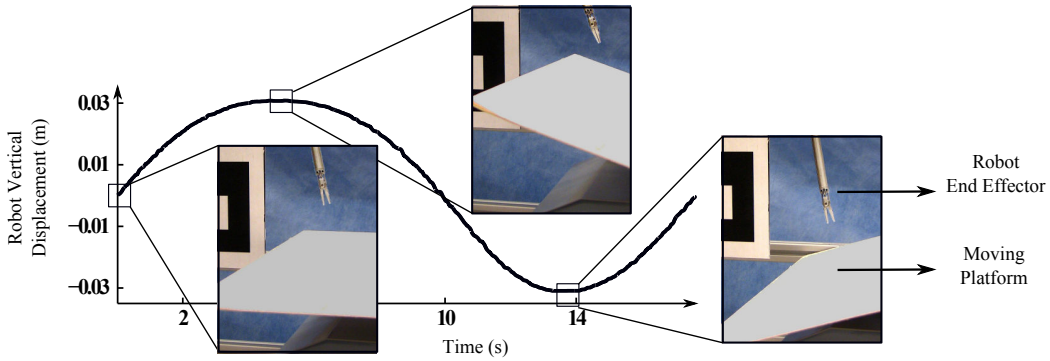

Figure 3.10: Position of the robot as it moves with the moving platform to maintain its distance with the forbidden region.

with the moving platform to keep a safe distance from the forbidden region. Other than the images obtained by the camera, no information about the motion of the platform and its pattern is provided. In addition, all processing is performed in real-time with forces rendered at $1kHz$ from images obtained at $30 frames/sec$. Thus the virtual fixture can be used as a method of teleoperating on a moving platform by compensating the motion.

This chapter presented a method for creating forbidden region virtual fixtures from streaming point clouds using constraint based haptic rendering. Experiments with a surgical robot verified that the methods can run in real-time and provide a stable interaction with a dynamic environment. Three architectures were demonstrated exhibiting both soft and hard virtual fixtures. A hard virtual fixture

prevented contact with the forbidden region. In contrast, a soft virtual fixture opposed motion inside the forbidden region but allowed the operator to violate the constraint by overcoming the opposing force. Both constraints could be desirable depending on the application. Our proposed method enables the operator to switch between the architectures on the fly.

Chapter 4

ADAPTIVE CONTROL IN CONTACT WITH TISSUE[1]

An effective implementation of the virtual fixtures in a surgical setting is dependent on the precise control of robot end effector. In today's robot assisted surgery the surgical robot control loop is closed by the surgeon. The desired end effector motions generated by the surgeon are translated to appropriate motor torques by a controller in order to match the end effector position with the desired position. However, an error can exist between the desired and actual end effector position due to unmodeled dynamics of the robot, contact with a variety of tissues, as well as improper design or tuning of the controller. Although undesirable, it does not pose a great problem in manual operation since the surgeon compensates for these errors by constantly monitoring the visual feedback from the surgical site and correcting for the robot's motion. If not compensated for, such end effector position errors can degrade the performance of virtual fixtures.

This chapter introduces a control framework for addressing the problem of controlling the robot in contact with tissue with unknown properties and in the absence of end effector sensors. In addition to its application in virtual fixtures, this framework is beneficial for the automation of surgical tasks. To this end, an optimal control strategy is proposed which takes into account the dynamics of the surgical robot as well as interactions with unknown environments such as soft tissue.

In order to simplify the problem we investigate it on a 1 DoF platform. We propose a method for automating the task of tissue compression in robotic surgery in the absence of contact force and position sensors. The goal is to design a controller that enables the surgeon to instantiate high level commands for execution of predefined tool trajectories for compression of soft tissue. The main challenges faced in achieving this objective are:

1. The biomechanical properties of soft tissue are nonlinear [69], time-varying and highly vari-

[1] ©[2012] IEEE. Parts of this chapter are reprinted, with permission, from [68] Nia Kosari, S., Ramadurai, S., Chizeck, H. J., & Hannaford, B. Robotic compression of soft tissue. In Robotics and Automation (ICRA), 2012 IEEE International Conference on (pp. 4654-4659).

able from one tissue to another depending on the organ as well as age and sex of the patient. In addition, these properties are often unknown in advance. The controller must adapt to the variations in tissue properties and be capable of executing precise motions in contact with such loads. Although methods for modeling the dynamics of soft tissue have been studied in the literature, e.g. [70, 71], these methods often rely on force measurements for identification of tissue properties and therefore suffer from the shortcomings of using force sensors described earlier. In addition, the obtained models have not been used in automating interactions with tissue.

2. The interaction forces between the grasper and the tissue are dependent on the dynamics of the grasper, e.g. inertia and friction. In the absence of distal sensors, these dynamics must be taken into account for automating interactions with tissue. Many surgical graspers are cable driven instruments. The elasticity of the cable as well as friction in the pulleys and the inertia of the instrument play an important role in achieving precise control in contact with tissue and must be considered in designing the controller.

The above challenges have been addressed for a very similar problem of automating grasping of objects with a compliant fingerpad in [72]. Two scenarios, with and without a fingertip sensor, were considered. It was concluded that the lack of fingertip sensors significantly increases the complexity of the control problem. The problem of automation of tissue compression was also investigated in [73]. Four different controllers were designed and their performances in the presence of noise were evaluated in simulations. It was shown that nonlinear Model Predictive Control (MPC) demonstrates the best performance under most noise conditions. However, the method relied on knowledge of tissue properties as well as contact force and position measurements and the dynamics of the grasper were ignored. In addition, methods presented in the above papers were limited to theory or simulations and were not evaluated in experiments. For a successful implementation issues such as the real-time capability of the methods have to be considered.

Our method builds on two threads of research to address the above problems; nonlinear estimation and Model Predictive Control. First, we investigate the use of nonlinear state and parameter estimation for identifying the dynamics of the cable driven manipulator. In order to achieve a realistic design it is assumed that contact force and position sensors are not available and these quantities

are estimated using the available measurements.

The dynamics of the manipulator and the estimated contact force and position are then employed by a Model Predictive Controller to control the grasper in contact with tissue as well as in free motion. Differential Dynamic Programming (DDP) is used to solve the nonlinear optimal control problem in real-time. Automatic Differentiation is proposed for obtaining fast and accurate derivatives required by DDP. The combination of the above methods results in an adaptive Model Predictive Controller for real-time automation of soft tissue compression.

The surgical grasper is a one DoF instrument and the task of tissue compression consists of motion in one direction. The proposed methods can be extended to control multiple degrees of freedom by replacing the grasper with a surgical robot. The latter part of this document focuses on extending the methods to control the position of a surgical robot. The complete dynamics of the robot including the cable transmission, gears and nonlinear friction at the joints must be taken into account. In addition, the lack of full state measurements are considered in the design. It is anticipated that this control algorithm enables the execution of predefined robot trajectories in free motion and in contact with unknown environments.

The rest of this chapter is organized as follows. Section 4.1 reviews the related literature. The Unscented Kalman Filter is presented in Section 4.2 and the dynamics of the grasper are described in Section 4.3. System identification on the grasper is performed in Section 4.4 and contact force estimation is presented in Section 4.5. The Model Predictive Controller using Differential Dynamic Programming is described in Section 4.6 and the implementation results on a one DoF robot is presented in Section 4.7.

4.1 Background and Literature Review

4.1.1 Tissue Modeling and Estimation

Grasping of soft tissue can be regarded as interaction with an unknown environment. One approach in achieving automation in such interactions is to model the characteristics of the unknown environment. To this end lumped models [74, 75] and Finite Element Methods [76, 77] have been studied in the literature. An extensive literature review on the behaviour of viscoelastic material was presented in [78]. A piezoelectric endoscopic tactile sensor was designed in [70] to estimate the parameters

of the linear Kelvin-Voigt model for soft tissue. This model and the nonlinear Hunt-Crossley model [79] were compared in [80] for contact with soft and rigid environments. It was shown that the Hunt-Crossley model provides a more accurate description of the contact specially for contact with soft materials such as silicone gel.

In [71] seven mathematical dynamic models of soft tissue were evaluated under palpation. Recursive Least Squares (RLS) technique was used to estimate the unknown parameters of the models from force and position measurements. The Hunt-Crossley model and a second order polynomial with the damping term were shown to have the minimum force estimation error in artificial tissue samples. The Hunt-Crossley demonstrated a correlation between the estimated stiffness parameter and the physical stiffness of the tissue. Therefore, the estimated stiffness from this model was used to distinguish between tissues with different stiffness and create a graphical overlay on the tissue images in order to highlight areas with higher stiffness representing artificial calcified artery.

The Hunt-Crossley model has nonlinear dependence on parameters. The RLS method has to be modified for identification of such models. A two stage RLS scheme with feedback [81] was used in both [80] and [71] to estimate the unknown parameters. [82] showed that this identification method is sensitive to initial conditions and dynamic parameter variations and suffers from slow convergence. The problem was then transformed to one that is linear in parameters and a single stage estimator was used to obtain the parameters. This method is valid when a certain relationship holds between the velocity in contact and the tissue parameters and when the power of noise is small. Although the method works in this case, in general the grasper may have nonlinear dynamics that are not linearizable through transformations.

In addition, the identification methods in [70, 71, 80, 82] rely on force measurements at the point of contact for estimating the characteristics of the tissue. In robotic minimally invasive surgery these measurements can be obtained by force sensors mounted on the laparoscopic instruments. The force sensor can be integrated in two ways: direct and indirect. In the indirect approach force sensors are placed away from the tool tip [83, 84]. Although the measured forces in this case are dependant on the end effector forces, factors such as friction, backlash and cable elasticity can lead to inaccuracies if not taken into account.

In the direct approach, sensors are placed at the point of contact resulting in more accurate measurements by eliminating the undesired consequences of the indirect method [23, 24, 85, 86].

However, sterilization is a challenge and severe size constrains must be satisfied. The direct and indirect methods of force sensing were compared in [23]. It was shown that the strain gauge placed on the handle of the laparoscopic tool (indirect) overestimates the tissue grasping force in comparison to the thin film force sensor mounted on the grasper jaw (direct). This can be caused by unmodeled/uncompensated dynamics of the grasper including inertia, friction and cable elasticity.

In an effort to address the shortcomings of the direct and indirect methods, actuated forceps with 6 degree of freedom force/torque sensing were designed in [25]. The sensor was placed between the gripper and joint and therefore was only subject to gripper cable force. This force was measured by an additional sensor and compensated for simultaneously. By placing the signal conditioning units in the shaft and apart from the tool tip the device was made sterilizable. Although the design does not alter the forceps' jaws and solves the problem of sterilizability it suffers from a shortcoming of the direct method. Laparoscopic tools have a very limited number of usages and the addition of force sensors to the instrument can dramatically increase the cost.

Contact force estimation has been studied in the literature as an alternative to force measurement. A model-based observer was proposed in [87] for estimation of environment forces acting on a rigid body. It was shown that if an accurate dynamic model of robot is available, environment forces can be estimated. RLS method was used to estimate the unknown inertia of the robot. in [88], contact force and inertial parameters of the load were estimated from wrist force measurements by employing an Kalman Filter. These methods are effective when the problem is linear in parameters, however, the same principle can be applied to models that are not linear in parameters by employing a nonlinear estimator. Contact force estimation was performed in [89] by employing an Extended Kalman Filter (EKF) and treating the environment force as an unknown input to the system. The nonlinear friction of the joint were neglected in this work.

4.1.2 Automated Grasping

Modelling and parameter estimation of tissue relaxation and grasper-tissue friction were proposed in [90] to detect incipient slip of tissue and distinguish it from tissue relaxation. A simple control law was employed to calculate the grasping force based on the tangential force and to compensate for incipient slip. However, a force sensor was employed to measure the grasp forces and the grasper

was not an laparoscopic grasper (linear motor with no cables).

Automation of tissue grasping was studied in [91] using a cable driven motorized grasper with no force sensor. Motor torque command was used in obtaining the contact force. A PD controller with feedforward friction compensation was implemented to control the position of the jaws. However, inertia of the tool and elasticity of the cable were ignored in designing the control law.

An optimal strategy for tissue retraction was proposed in [76] using finite element methods. The method was validated in simulations. Four different control strategies were proposed in [73]: 1) a well tuned PID, 2) feedback linearization in combination with deadbeat control, 3) an optimal open-loop control law and 4) a model predictive controller. It was shown that the model predictive controller performs best for the widest range of noise. However, the conclusions were purely based on simulations assuming knowledge of the tissue parameters and grasp position.

4.2 Unscented Kalman Filter

In the absence of end effector sensors, state estimation techniques can be employed to provide us with estimates of the desired quantities. The problem of state estimation from noisy measurements has been well studied in the literature. The Kalman Filter [92] is an optimal method when the system dynamics and observations are linear. However many real life systems can not be modelled by a linear system. The Extended Kalman Filter (EKF) has been widely used for state estimation with nonlinear dynamics and/or measurements. The EKF linearizes all the nonlinearities and applies the Kalman filter equations. The EKF has a number of drawbacks mainly due to the linear approximation [93]:

- It requires the nonlinearities to be differentiable.
- It requires the calculation of Jacobian matrices which can be nontrivial.
- Where the nonlinearity can not be approximated by local linearization the results are erroneous and may even cause the filter to diverge.

The Unscented Kalman Filter (UKF) [93] is an alternative to the EKF for nonlinear state estimation. The UKF does not require the Jacobian matrices and therefore is easier to implement and can

be applied to systems with non-differentiable nonlinearities. EKF and UKF are both of the same order of computational complexity, $O(L^3)$ [94], where L is the state dimension.

Consider the following discrete-time system

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{G}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{H}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{v}_k \end{aligned} \tag{4.1}$$

where $\mathbf{x} \in \Re^n$ is the vector of system states, $\mathbf{u}$ is the input vector, $\mathbf{y}$ is the measurement vector and the subscript k represents the time step. $\mathbf{w}$ and $\mathbf{v}$ are respectively the process and measurement noise with zero mean and covariance matrices R_w and R_v.

The unscented Kalman filter inherits its power from the unscented transformation. By evaluating a deterministic set of sample points, also known as sigma points, the unscented transformation calculates the statistics of a random variable undergoing a nonlinear transformation. Algorithm 4.2.1 shows the UKF equations. In these equations $10^{-4} \leq \alpha \leq 1$ determines the spread of sigma points around the mean, $\hat{\mathbf{x}}$, and β is a constant associated with the prior knowledge of the distribution.

The UKF requires knowledge of the dynamics of the system, i.e. functions $\mathbf{G}$ and $\mathbf{H}$ in (4.1), in order to perform the state estimation. However, in many cases the dynamics of the system are not known. The UKF can be used for joint (dual) state and parameter estimation. To this end, the parameters must be augmented as system states in the dynamic equations.

$$\begin{aligned} \mathbf{x}_k^a &= \begin{bmatrix} \mathbf{x}_k \\ \\ \mathbf{z}_k \end{bmatrix} \\ \mathbf{x}_{k+1}^a = \begin{bmatrix} \mathbf{x}_{k+1} \\ \\ \mathbf{z}_{k+1} \end{bmatrix} &= \mathbf{G}^a(\mathbf{x}_k^a, \mathbf{u}_k) + \mathbf{w}_k^a \equiv \begin{bmatrix} \mathbf{G}\left(\mathbf{x}_k, \mathbf{z}_k, \mathbf{u}_k\right) + \mathbf{w}_k \\ \\ \mathbf{z}_k + \mathbf{w}_{zk} \end{bmatrix} \\ \mathbf{y}_k &= \mathbf{H}(\mathbf{x}_k, \mathbf{z}_k, \mathbf{u}_k) + \mathbf{v}_k \end{aligned} \tag{4.2}$$

where $\mathbf{z}$ is the vector of parameters, $\mathbf{x}^a$ is the augmented vector of states and parameters, $\mathbf{w}_z$ is the process noise on parameters and $\mathbf{w}_z^a$ is the augmented parameter and state process noise vector.

The unscented Kalman filter will be used in this document to obtain combined state and parameter estimates of a cable driven manipulator in free motion and in contact with soft tissue. The next two chapters are devoted to deriving the dynamics of the manipulator and tissue.

Initialization:

$$\hat{x}_0 = E[x_0]$$
$$P_0 = E\left[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T\right]$$

Sigma point calculation:

$$X_{k-1} = \left[\hat{x}_{k-1}\ ,\ \hat{x}_{k-1} \pm \sqrt{(L+\lambda)P_{k-1}}\right]$$

Time update:

$$X^*_{k|k-1} = G(X_{k-1})$$
$$\hat{X}_{k|k-1} = \sum_{i=0}^{2n} w_i^m X^*_{i,k|k-1}$$
$$P_{xk|k-1} = \sum_{i=0}^{2n} w_i^c \left(X^*_{i,k|k-1} - \hat{x}_{k|k-1}\right)\left(X^*_{i,k|k-1} - \hat{x}_{k|k-1}\right)^T + R_w$$
$$X_{k|k-1} = \left[\hat{x}_{k|k-1}\ ,\ \hat{x}_{k|k-1} \pm \sqrt{(L+\lambda)P_{xk|k-1}}\right]$$
$$Y_{k|k-1} = H\left(X_{k|k-1}\right)$$
$$\hat{y}_{k|k-1} = \sum_{i=0}^{2n} w_i^m Y_{i,k|k-1}$$

Measurement Update:

$$P_{ykyk} = \sum_{i=0}^{2n} w_i^c \left(Y_{i,k|k-1} - \hat{y}_{k|k-1}\right)\left(Y_{i,k|k-1} - \hat{y}_{k|k-1}\right)^T + R_v$$
$$P_{xkyk} = \sum_{i=0}^{2n} w_i^c \left(X_{i,k|k-1} - \hat{x}_{k|k-1}\right)\left(X_{i,k|k-1} - \hat{x}_{k|k-1}\right)^T$$
$$K_{xk} = P_{xkyk} P_{ykyk}^{-1}$$
$$\hat{x}_k = \hat{x}_{k|k-1} + K_{xk}\left(y_k - \hat{y}_{k|k-1}\right)$$
$$P_{xk} = P_{xk|k-1} - K_{xk} P_{ykyk} K_{xk}^T$$

with

$$w_0^m = \frac{\lambda}{n+\lambda}\ , \quad w_0^c = \frac{\lambda}{n+\lambda} + \left(n - \alpha^2 + \beta\right)$$
$$w_i^m = w_i^c = \frac{\lambda}{2(n+\lambda)}\ , \quad i = 1, ..., 2n$$
$$\lambda = n\left(\alpha^2 - 1\right)$$

Algorithm 4.2.1: Unscented Kalman filter equations [95][93].

4.3 Grasper Dynamics

A large group of robot-assisted surgical instruments belong to the class of cable driven manipulators, also referred to as tendon-based manipulators. Cables enable the transmission of mechanical power over a distance and therefore allow the motors to be placed away from the joints. This is of great importance in robot-assisted surgery due to size and sterilization constraints. In addition, cables are flexible and in conjunction with pulleys can change the direction of force.

Despite the advantages of the tendon-based transmission, cables and pulleys introduce nonlinear dynamics into the system. Cables are elastic and as a result stretch under tension. The amount of this stretch depends on cable properties as well as the tension applied. In addition, sliding of individual strands within cables as they roll over pulleys introduces friction into the dynamics.

In this document we consider a cable driven surgical grasper and model it as a 1 degree-of-freedom (DoF) cable driven manipulator. Dynamics of such manipulators including the viscoelastic properties of cable and inertia of the pulleys were investigated in [96] and a nonlinear model of the friction between the cable and pulleys was derived in [97]. In this document we use the lumped model proposed in [98]. Although the grasping DoF of a surgical grasper may consist of several pulleys and complex cable routing, it is assumed that it can be modeled as a motor side, a link side and a massless elastic coupling connecting the two sides. It is further assumed that cables are sufficiently pretensioned to prevent them from going slack and that the friction and inertia of the pulleys can be lumped into either side.

A schematic drawing of this model is shown in Fig. 4.1. The motor and link sides are connected by two cables each modeled by a damper and a nonlinear spring. The relationship between the relative cable stretch, d, and the spring force, F_{spring}, is given by

$$F_{spring} = k_e(e^d - 1). \tag{4.3}$$

where k_e is the stiffness of the cable. The rest of symbols are described in Table 4.1. Taking q_m, q_l and $\dot{q_m}, \dot{q_l}$ as the states of the system, the dynamic equations of the grasper can be written in state-space form as

$$\mathbf{J}\ddot{\mathbf{q}} + \mathbf{T} + \mathbf{F} + \mathbf{N} = \tau \tag{4.4}$$

Grasper Parameters and States					
Symbol	Description	Unit	Symbol	Description	Unit
$q_{m,l}$	motor, link angle	rad	$F_{vm,vl}$	motor, link viscous friction coeff.	Nms/rad
$\dot{q_{m,l}}$	motor, link velocity	rad/s	m_l	link mass	kg
$\tau_{m,l}$	motor, link torque	Nm	L_{cm}	center of mass of the jaw	m
$J_{m,l}$	motor, link inertia	kgm^2	J_{cm}	moment of inertia of the jaw	kgm^2
$r_{m,l}$	motor, link capstan radius	m	k_e	cable stiffness	N/m
$F_{m,l}$	motor, link friction torque	Nm	b_e	cable damping	Ns/m
$F_{cm,cl}$	motor, link Coloumb friction coeff.	Nm	F^*	contact force	N

Table 4.1: Parameters and states of the grasper model. q_m, q_l and $\dot{q_m}, \dot{q_l}$ constitute the states of the system.

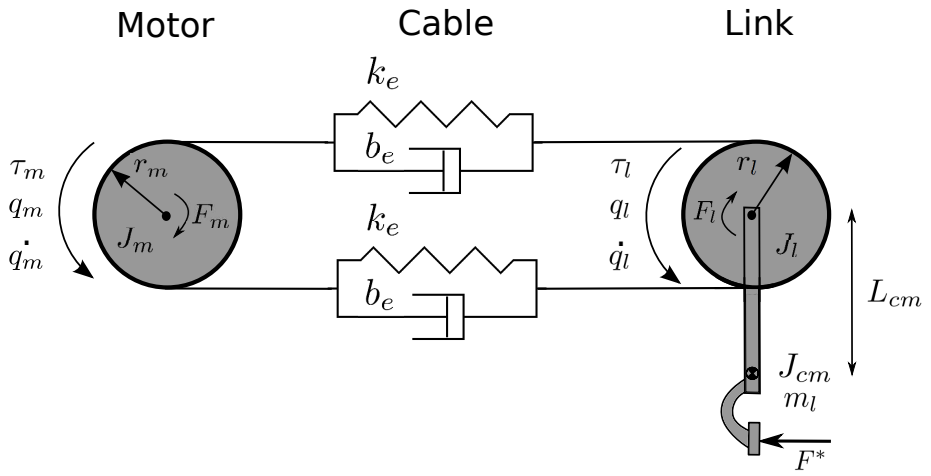

Figure 4.1: Lumped 1DoF model of a cable driven robotic grasper.

where τ is the vector of external torques, **T** contains the effect of cable stretch, **F** is the vector of friction torques, **N** contains the gravitational torques, **J** is the matrix of inertias and $\ddot{\mathbf{q}}$ is the vector of motor and link angular accelerations

$$\boldsymbol{q} = \begin{bmatrix} q_m \\ q_l \end{bmatrix}, \quad \boldsymbol{J} = \begin{bmatrix} J_m & 0 \\ 0 & J_l \end{bmatrix}, \quad \boldsymbol{T} = \begin{bmatrix} r_m\gamma \\ -r_l\gamma \end{bmatrix},$$

$$\tau = \begin{bmatrix} \tau_m \\ \tau_l \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} F_m \\ F_l \end{bmatrix}, \quad \boldsymbol{N} = \begin{bmatrix} 0 \\ m_l L_{cm} g sin(q_l) \end{bmatrix},$$

$$\gamma = k_e(e^{(q_m r_m - q_l r_l)} - e^{(q_l r_l - q_m r_m)}) + 2b_e(\dot{q_m} r_m - \dot{q_l} r_l),$$

$$F_i = F_{c,i} sign(\dot{q_i}) + F_{v,i}\dot{q_i}, \quad i \in \{m, l\}$$

While sensors can be employed for measuring the motor states, q_m and $\dot{q_m}$, it is assumed the link states, q_l and $\dot{q_l}$ cannot be directly measured due to size and sterilization constrains and have to be estimated. The motor torque, τ_m, can be controlled and is considered the input to the system.

The continuous-time dynamics described above must be descretized for the application of UKF and for implementation on a digital processor. This discretization is performed through numerical integration. Care must be taken in choosing the appropriate numerical integration method as certain methods yield inaccurate discretizations due to the stiff nature of the equations. Appendix A compares a number of discretization methods and describes one that is suitable for the problem at

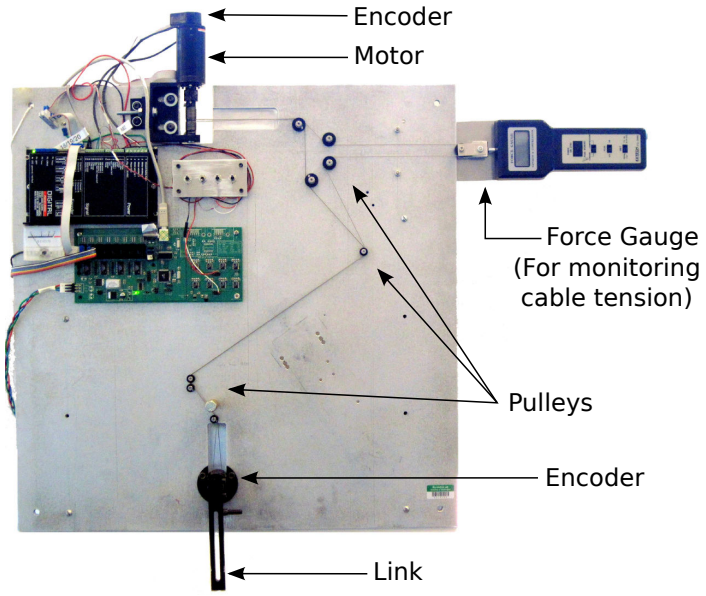

Figure 4.2: 1 DoF cable driven manipulator.

hand.

4.4 System Identification

Fig. 4.2 shows the 1 DoF cable driven manipulator employed in this work. A stainless steel cable of length 225 cm couples the motor and link side, passing through four idler pulleys with a diameter of 15mm and four idler pulleys with a diameter of 7.5 mm. The cable routing reflects that of one DoF of a surgical robot. This test setup will put the lumped model to test for a manipulator with complex cable routing and several pulleys.

The computer used to control the manipulator is a PC with a 64-bit Intel core i7 Sandy Bridge processor (3.4 GHz) with 4 GB of RAM. The system runs on Ubuntu 10.04 with RT-Preempt patch and the communication with the motor and data acquisition is through a custom-made USB I/O board.

Although the manipulator is equipped with both motor and link encoders, in this document we assume that the only available measurement is the motor angle. The link angle measurements will only be used as ground truth for model validation since they are usually not available in robot-assisted surgical systems. The resolution of the motor encoder is 2000 counts per revolution. Among

the parameters described in Section 4.3, cable parameters i.e. k_e and b_e and friction coefficients i.e. F_{cm}, F_{cl}, F_{vm} and F_{vl} are unknown and have to be estimated. The rest of the parameters can be obtained through straightforward measurements or by consulting the motor data sheet.

The model described by (4.4) exhibits nonlinear dependence on states. The Unscented Kalman Filter (UKF) [99] can be employed for solving the joint estimation problem by augmenting the unknown parameters and states as described in Section 4.2.

The parameters of the manipulator are identified in free motion, i.e. $F^* = 0$ and $\tau_l = 0$. First, the cable parameters are identified by applying a high frequency ($0.5 \sim 3.5$ Hz) input torque. The friction coefficients are then identified by fixing the obtained cable parameters and applying a low frequency ($0.2 \sim 1.2$ Hz) torque for which the effect of friction is more salient. Fig. 4.3 shows the estimated parameters from the UKF. All parameters show convergence within 100 seconds. In the rest of this document the parameters of the grasper will be fixed at the estimated values.

4.5 Contact Force Estimation

The identified dynamics of the manipulator can be used for estimating the external torque. Instead of estimating the contact force acting on the grasping jaw, F^*, we estimate the resulting torque on the link, τ_l. Since the length of the grasper is known, the relationship between the two quantities is straightforward. In addition, to compensate for the contact force knowledge of τ_l is sufficient for the controller.

One way of estimating the external torque is to solve (4.4) for this quantity. This method requires knowledge of motor and link angular accelerations in addition to the their states i.e. position and velocity. Although the state and parameter estimator employed in the previous section provides an estimate of the states, obtaining the angular acceleration requires differentiating the angular velocity and leads to noisy estimates. Another method is to treat the external torque as an unknown parameter and to estimate it along with the states. This can be done by implementing the dual state and parameter estimator described in Section 4.4.

To validate the torque estimation, a linear extension spring with a known spring constant is attached to the manipulator and an open loop torque is applied to extend the spring. The displacement of the spring is obtained from the link encoder measurements and is used to calculate the external

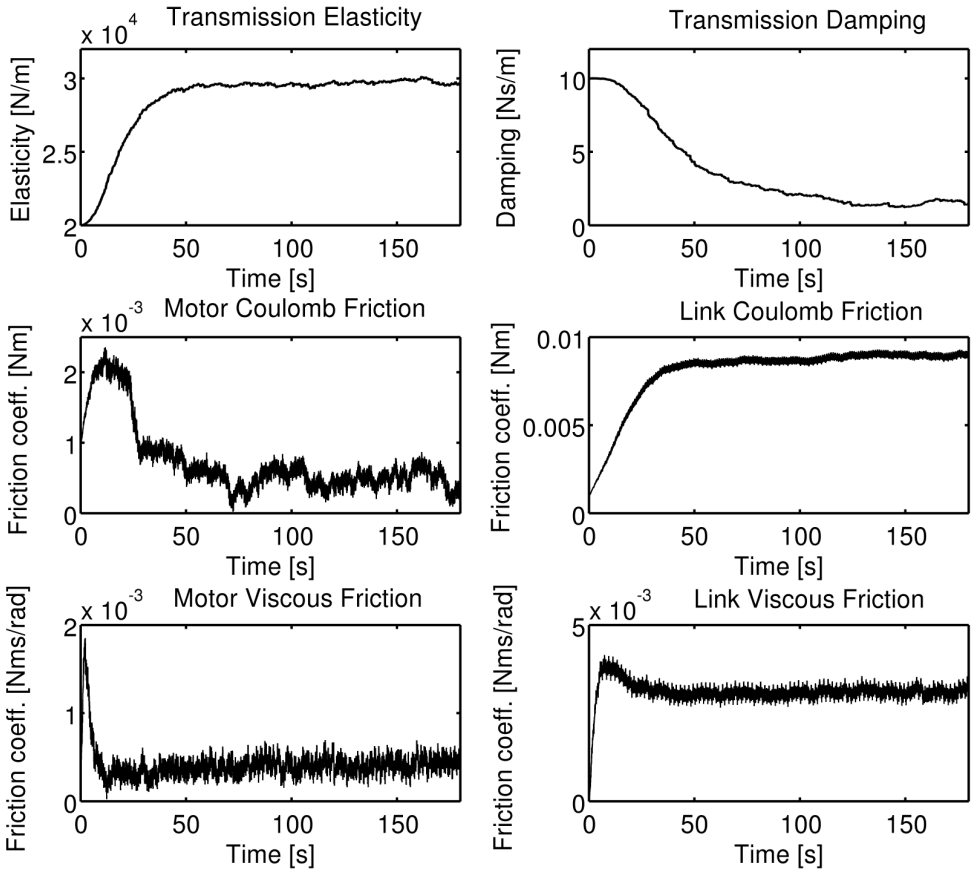

Figure 4.3: The estimated parameters of the grasper.

torque acting on the spring.

The UKF uses the motor encoder readings to estimate the external torque on the link. These values are shown in Fig. 4.4. Although no information about the spring constant is provided to the estimator, an external torque is detected upon applying the inputs. It was observed that the measured spring torques are slightly smaller than the estimated values. This might be due to imperfect spring characteristic including the spring's initial load requirement which reduces the effective torque on the spring as well as any unmodeled system dynamics including static friction. In essence any mismatches between the model and the actual system will be reflected in the external torque estimate. However, this should not pose a serious problem to the controller since this mismatch has to be compensated for regardless of its source.

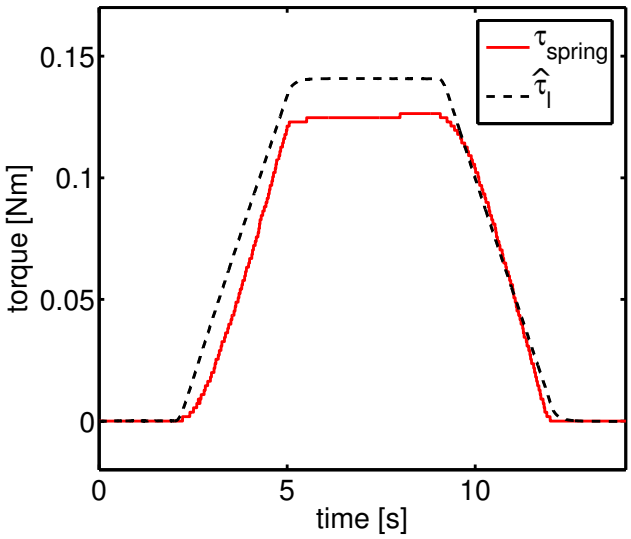

Figure 4.4: The measured spring torque, τ_{spring} vs. the estimated external torque, $\hat{\tau}_l$.

4.6 Model Predictive Control

4.6.1 Optimal Control Problem

In Model Predictive Control (MPC) the control plan over a fixed time horizon in the future is determined by solving an optimization problem over that horizon. The optimization problem typically includes minimizing a cost function of states and inputs. MPC incorporates a model of the system to predict the states in the future by propagating the dynamics forward in time. The first part of the control sequence is applied and the optimization problem is repeated for the shifted time horizon. This class of control algorithms are also known as Receding Horizon Control (RHC).

Ease of tuning and the ability to handle nonlinear dynamics as well as input and state constraints have made MPC a standard control strategy in many applications. The main drawback of MPC is that it requires solving an optimization problem at each time step. Until recently, applications of MPC to nonlinear problems had been limited to systems with sample times in the order of seconds, minutes or hours, due to the computational cost required for solving such optimization problems [100].

The goal of MPC is to minimize a cost-to-go at any time instant, k, given the dynamics of the

system

$$x_{k+1} = G(x_k, u_k). \tag{4.5}$$

In the above state-space description $x \in \Re^n$ is the vector of states and $u_k \in \Re^m$ is the vector of inputs. The cost-to-go at time k is a function of the state at that time and the inputs from time k to the end of the horizon, $N-1$. The input at time N only affects the state at time $N+1$ and therefore is not included in the cost-to-go. $J(x_k, u_{k,\cdots,N-1}, k)$ denotes this cost-to-go at time k with state x_k and inputs $u_{k,\cdots,N-1} \equiv \{u_k, u_{k+1}, \cdots, u_{N-1}\}$. This cost consists of the sum of instantaneous costs, $l(x,u,i)$, over the horizon and a final cost, $l_f(x)$, i.e.

$$J(x_k, u_{k,\cdots,N-1}, k) = \sum_{i=k}^{N-1} l(x_i, u_i, i) + l_f(x_N) \tag{4.6}$$

The optimization problem can therefore be formulated as

$$\begin{aligned} &\min_{u_1,\cdots,u_{N-1}} \quad J(x_1, u_1, \cdots, u_{N-1}, 1) \\ &\text{subject to} \quad x_{k+1} = G(x_k, u_k). \end{aligned} \tag{4.7}$$

Although the algorithm results in a control sequence over the entire horizon, in the receding horizon implementation only the first time step of the control sequence, u_1, is applied and the optimization algorithm is repeated at the next time cycle. The remaining steps of the control sequence are used as an initial guess to warm start the optimization problem at the next time step. This is done by shifting the sequence in time and appending the last element in the sequence, i.e.

$$u_1, \cdots, u_{N-1} \longleftarrow \{u_2, u_3, \cdots, u_{N-1}, u_{N-1}\}. \tag{4.8}$$

In the recent years, advancements in the computational power of processors has brought a great deal of interest in applying MPC to robotic applications that require millisecond sampling times. These approaches can be divided into two categories: simultaneous and sequential. Simultaneous algorithms search over the state and control space treating the dynamics as constraints whereas sequential algorithms search over the control space and propagate the states over the dynamics. Sequential algorithms optimize over a smaller space and therefore have an advantage in robotic applications [101].

4.6.2 *Differential Dynamic Programming*

Differential Dynamic Programming (DDP) [102] is a sequential method for solving the optimization problem of 4.7 that builds on the dynamic programming principle. DDP is an iterative method with each iteration consisting of two steps: a backward pass and a forward pass. In the backward pass quadratic models of the cost and the dynamics along the nominal states are employed for calculating a set of feedforward and feedback gains for the entire horizon. These gains result in a control sequence that is used in the forward pass to simulate the system to new nominal states. The backward and forward pass are repeated until convergence is achieved.

The notation used here in formulating DDP is that of [103]. Let $V(x,k)$ denote the optimal value function as time k, i.e. is the cost-to-go given the optimal control sequence

$$V(x,k) = \min_{u_{k,\cdots N-1}} J(x_k, u_{k,\cdots N-1}, k) \tag{4.9}$$

The cost to go at the end of the horizon consists of the final cost and is denoted by $V_N(x) = l_f(x_N)$. Applying the dynamic programming principle gives

$$V(x,k) = \min_{u} \left[l(x,u) + V(G(x,u), k+1) \right]. \tag{4.10}$$

Backward Pass

The backward pass of the DDP recursively constructs a quadratic approximation to the dynamics, $G(x,u)$, and the optimal value function, $V(x,k)$, and finds feedforward and feedback gains for generating the input, $u(x,k)$. Define $Q(\delta x, \delta u)$ as a perturbation of the arguments of the minimum in 4.10.

$$Q(\delta x, \delta u) \equiv l(x+\delta x, u+\delta u, k) - l(x,u,k) + V(G(x+\delta x, u+\delta u), k+1) - V(G(x,u), k+1) \tag{4.11}$$

Expanding to second order Taylor series yields

$$\frac{1}{2} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \times \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}. \tag{4.12}$$

Matching the equivalent coefficients results in

$$Q_x = l_x + G_x^T V_x' \tag{4.13a}$$

$$Q_u = l_u + G_u^T V_x' \tag{4.13b}$$

$$Q_{xx} = l_{xx} + G_x^T V_{xx}' G_x + V_x' \cdot G_{xx} \tag{4.13c}$$

$$Q_{uu} = l_{uu} + G_u^T V_{xx}' G_u + V_x' \cdot G_{uu} \tag{4.13d}$$

$$Q_{ux} = l_{ux} + G_u^T V_{xx}' G_x + V_x' \cdot G_{ux} \tag{4.13e}$$

where the dot operator $(\cdot)$ in (4.13c, 4.13d, 4.13e) represents contraction with a tensor. In the above equations time index k is dropped and prime denotes the next time step, i.e. $V' = V(k+1)$. Minimizing 4.12 with respect to δu yields

$$\underset{\delta u}{\operatorname{argmin}} \quad Q(\delta x, \delta u) = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta x). \tag{4.14}$$

The above input consists of a feedforward term $k_{ff} = -Q_{uu}^{-1} Q_u$ and a feedback term $k_{fb} = -Q_{uu}^{-1} Q_{ux}$. Replacing these values in 4.12 we get

$$V_x = Q_x - (Q_u^T Q_{uu}^{-1} Q_{ux})^T \tag{4.15a}$$

$$V_{xx} = Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{ux}. \tag{4.15b}$$

A feedforward gain and a feedback gain is obtained by recursive evaluation of equation set (4.13) and (4.16) for each time instant. The output of the backward pass is this sequence of gains calculated backward in time.

Forward Pass

The forward pass employs the gains calculated in the backward pass and simulates the system forward in time resulting in a new state trajectory:

$$\hat{x}_1 = x_1 \tag{4.16a}$$

$$\hat{u}_k = u_k + k_{ff,k} + k_{fb,k}(\hat{x}_k - x_k) \tag{4.16b}$$

$$\hat{x}_{k+1} = G(\hat{x}_k, \hat{u}_k) \tag{4.16c}$$

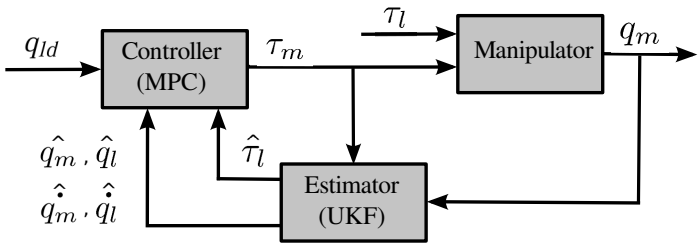

Figure 4.5: Block diagram of the closed-loop system. The estimates are denoted by $\hat{}$.

The feedforward and feedback passes are repeated for either a fixed number of iterations or until the change in the gains becomes smaller than a threshold. At the end of the DDP cycle the first elements of the feedforward and feedback sequence are used in generating the control action.

4.7 Implementation and Results

Fig. 4.5 shows a block diagram of the closed loop system. It is common in practice to design the estimator and controller separately. The UKF takes the motor torque and angle and estimates the states of the system along with any external torques as described in Section 4.5. The controller uses these estimates along with the desired link angle, q_{ld}, and generates the motor torque to be applied. In addition to these estimates, DDP relies on the knowledge of the dynamics of the system. The parameters identified in Section 4.4 are used to construct the dynamic equations of the system.

As a part of the MPC design process an instantaneous cost function has to be defined. One of the advantages of the DDP is its ability to handle input constraints by incorporating a barrier function in the cost. The role of the barrier function is to impose little or no cost when the constraints are met and a large cost when they are not. This is of great importance in robotic applications where the motor torques have to satisfy saturation and safety limits. Our instantaneous cost function, l, is constructed of a quadratic function of the link angle tracking error as well as a quadratic and a barrier function on the motor torque. The cost is defined by

$$l(q_l, \tau_m) = 80\ (q_l - q_{ld})^2 + 2 \times 10^{-3}\ {\tau_m}^2 + B(\tau_m) \tag{4.17}$$

where the weights were tuned for acceptable performance and stable operation. $B(\tau_m)$ is a barrier function on the motor torque. Since DDP uses quadratic approximation to the cost, the barrier function has to be twice differentiable. The following barrier function was employed to constrain

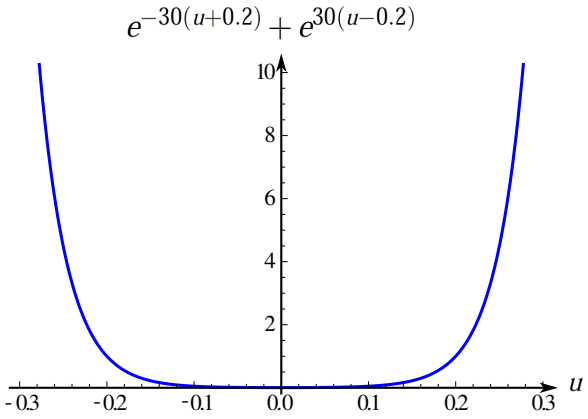

Figure 4.6: A barrier function for implementing input constraint.

the magnitude of the motor torque by τ_{max}

$$B(\tau_m) = e^{-\alpha(\tau_m+\tau_{max})} + e^{\alpha(\tau_m-\tau_{max})}. \tag{4.18}$$

Fig. 4.6 shows the barrier function for $\alpha = 30$ and $\tau_{max} = 0.2$ Nm.

The backward pass of DDP uses a second-order approximation to the cost and dynamics. This requires calculating the first and second order derivative of the cost function and dynamics and is the bottleneck of online optimization [101]. We propose the use of Automatic Differentiation for this purpose. In comparison to numerical and symbolic differentiation methods, derivative values obtained by automatic differentiation are exact (to round-off) and are often less expensive to compute [104]. This is essential in obtaining a real-time implementation of DDP considering the number of differentiations that have to be carried out during each backward pass.

In our speed tests with a number of automatic differentiation packages the CppAD package [105] performed the best and was used in this work. CppAD is an open source package for automatic differentiation of algorithms defined in C++. In this document the sampling time was set to 1 millisecond and at every sampling time three iterations of DDP were performed to solve the optimization problem with a horizon of 10 steps.

The model of Coulomb friction described in Section 4.3 uses a sign function. In order to make the dynamics differentiable this function was replaced by an approximation to the sign function given by

$$sgn(x) \approx \frac{10\,x}{\sqrt{1+100\,x^2}}. \tag{4.19}$$

In addition, to achieve the 1 ms sampling rate, the second derivative of the dynamics (Hessian) was ignored, resulting in a variant of DDP known as iterative Linear-Quadratic-Guassian (iLQG). This method uses a quadratic approximation to the cost and a linear approximation to the dynamics. iLQG enables faster implementations and is equally accurate in practice [106].

A sample of commercially available artificial tissue (Simulab Corporation, Seattle, WA) was used to evaluate the control performance in contact with tissue. Fig 4.7 shows the artificial tissue containing layers simulating skin and subcutaneous fat in contact with the manipulator.

MPC requires knowledge of the desired trajectory over a horizon in future. The desired link trajectory is designed to resemble that of a grasper compressing tissue. It consists of ramp and hold trajectories including free motion and contact with tissue as well as transitions between the two states. Ideally the link should track the desired trajectory in free motion and in contact with smooth transitions. The performance of the controller for such a trajectory is shown in Fig. 4.8.(a). The manipulator is in free motion for link angles less that zero and it is in contact for link angles above it. However, no information about the position or properties of the tissue is provided to the controller and no prior identification of tissue is performed. The link angle follows the desired trajectory both in free motion and contact with an r.m.s. error of 0.0069 radian or 0.4 degree. The tracking performance is near perfect given the encoder resolution of 0.0031 radian. In addition, the transitions between the two states are seamless.

Fig. 4.8.(b) shows the estimated external torques on the link. The estimated values show an increase in the torque as soon as the link moves into contact. A small torque (less that 0.02 Nm) is estimated in free motion. This can be due to any unmodeled dynamics of the manipulator including static friction or imperfect motor controllers. However, the tracking performance is not greatly affected since these unmodeled dynamics are captured in the estimated external torque and are compensated by the controller.

This chapter demonstrated the use of nonlinear estimation and Model Predictive Control for trajectory following in contact with tissue with unknown dynamics. The friction of the mechanism and the elasticity of the cable were taken into account in the design. In addition, contact sensors were omitted for a realistic implementation. The Unscented Kalman Filter was used to obtain a dynamic model of the manipulator in free motion, and to estimate the contact torques acting on the manipulator. These estimates were employed by a model predictive controller for generating the

Figure 4.7: (a) Tissue Sample consisting of layers for simulating skin and subcutaneous fat. (b) Manipulator in contact with artificial tissue sample.

appropriate motor torques for trajectory tracking in free motion and in contact with an artificial tissue sample. Differential Dynamic Programming with Automatic Differentiation was used for solving the optimal control problem in real-time. The method was evaluated in experiments with artificial tissue samples and showed good tracking in free motion and contact with seamless transitions. The templated software allows for extension to higher degrees of freedom on a surgical robot. The improved tracking performance in contact with tissue can benefit the implementation of virtual fixtures for robotic surgery.

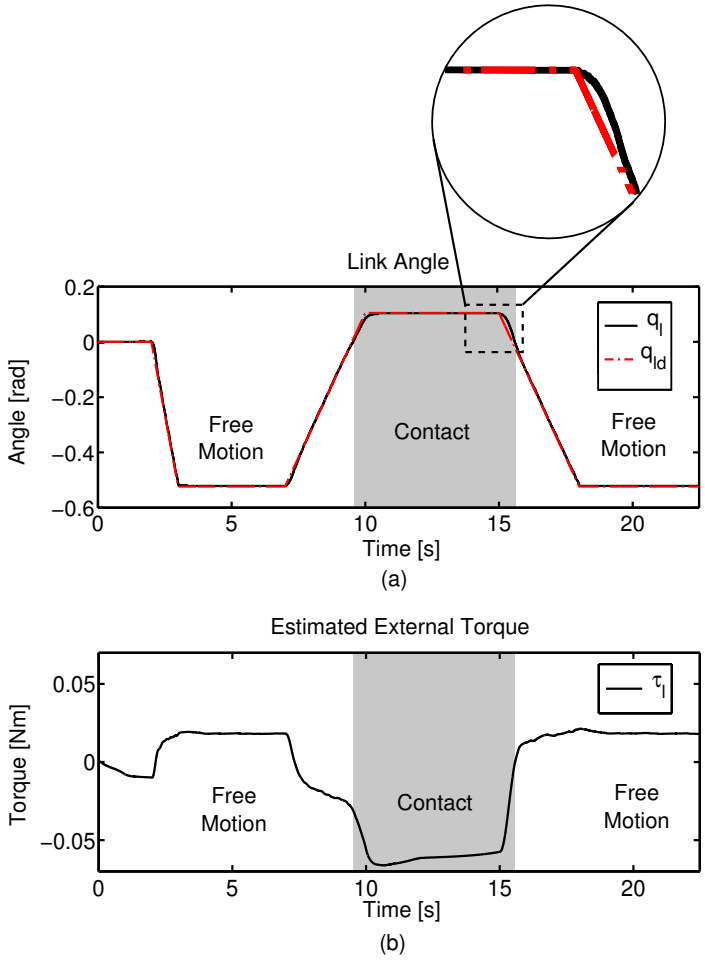

Figure 4.8: Tracking performance in free motion and contact with artificial tissue sample. The shaded region denotes contact with tissue. (a) Desired and actual link trajectory. (b) Estimate of external torque on the link.

Chapter 5

CABLE TENSION AND CONTROL PERFORMANCE[1]

During long term operation of a cable driven mechanism, tension can change due to wear of components or stretching of the cable. It can be an expensive maintenance operation, requiring expertise and special instruments, to re-tension all cables in a complex mechanism such as a surgical robot. To avoid frequent maintenance, the robot is normally designed to be robust to varying cable tension, but performance might then be sub-optimal. A method for tracking cable tension is therefore of value which 1) does not require expensive tension sensors, 2) can compensate the control system for changes in dynamics due to cable tension changes, 3) can alert the user when cable tension is outside of safe limits, and 4) can instruct maintenance workers on how to adjust tension (which axes and which direction to change tension).

This chapter investigates the effect of variations in tension on the controller performance and proposes a method to compensate for these effects. In addition, a method is developed to estimate cable tension from dynamic parameters of the system. This method provides a way to ensure that cable tension is within a specified range when direct measurement of the tension is not available due to cost, size and sterilizability constraints.

5.1 Method

The following experiments are carried out to investigate the effect of cable tension on controller performance. It is assumed that the changes in tension affect the dynamic parameters of the system described in the previous chapter, i.e. coefficients of friction and cable parameters.

1. The cable tension was set to a nominal value (743 gr) and the parameters of the system were estimated (nominal parameters).

---

[1]Parts of this chapter have appeared in [107] Nia Kosari, S., Ramadurai, S., Chizeck, H. J., & Hannaford, B. "Control and Tension Estimation of a Cable Driven Mechanism under Different Tensions." In 37th Mechanisms and Robotics Conference (MR), 2013.

2. The performance of the system at the nominal tension in response to defined target trajectories was obtained.

3. Cable tension was changed to 988, 1575 and 1929 grams and at each increment the off-line parameter estimation was repeated as described in Section 4.4 and the model parameters are updated. The response of the system to the defined target trajectories with both the original (nominal) and the adjusted parameter estimates were measured and compared.

The objective is to measure the effect of variations in cable tension on tracking performance a) when the changes are not compensated for by the controller, and b) when the changes are compensated by the controller through the updated model.

5.1.1 Results

In order to evaluate the performance of the system, two target trajectories for the link angle were defined: a step to 45 degrees and a recorded hand motion trajectory. The step response demonstrates defined characteristics that are of interest to control engineers (such as steady state error and overshoot). The hand trajectory represents typical input trajectory to a surgical robot. Approximately 12 seconds of hand motion trajectory was recorded by moving the end effector and recording the link angle while no motor torque was being applied.

The tension was increased in three steps to 988, 1575 and 1929 grams respectively and at each tension the pulleyboard parameters were estimated. At each tension, the performance of the controller in response to the two target trajectories was obtained, with and without adjusting to the updated parameters [1]. We call these the "adjusted response" and "nominal response" respectively. Fig. 5.1 shows the link angle in response to the step command for different tension levels. The system demonstrates a stable response in all cases. However, with nominal parameters the performance of the system deteriorates as the tension deviates from the 743 gram test mark. The adjusted response remains approximately unchanged with an overshoot of $9.1\% - 13.3\%$, rise time of $86 - 94$ms and a 5% settling time of $258 - 370$ms.

[1] the controller used in this chapter is a model predictive controller similar to the controller introduced in the previous chapter. However, the implementation is slightly different in the way the optimization problem is solved

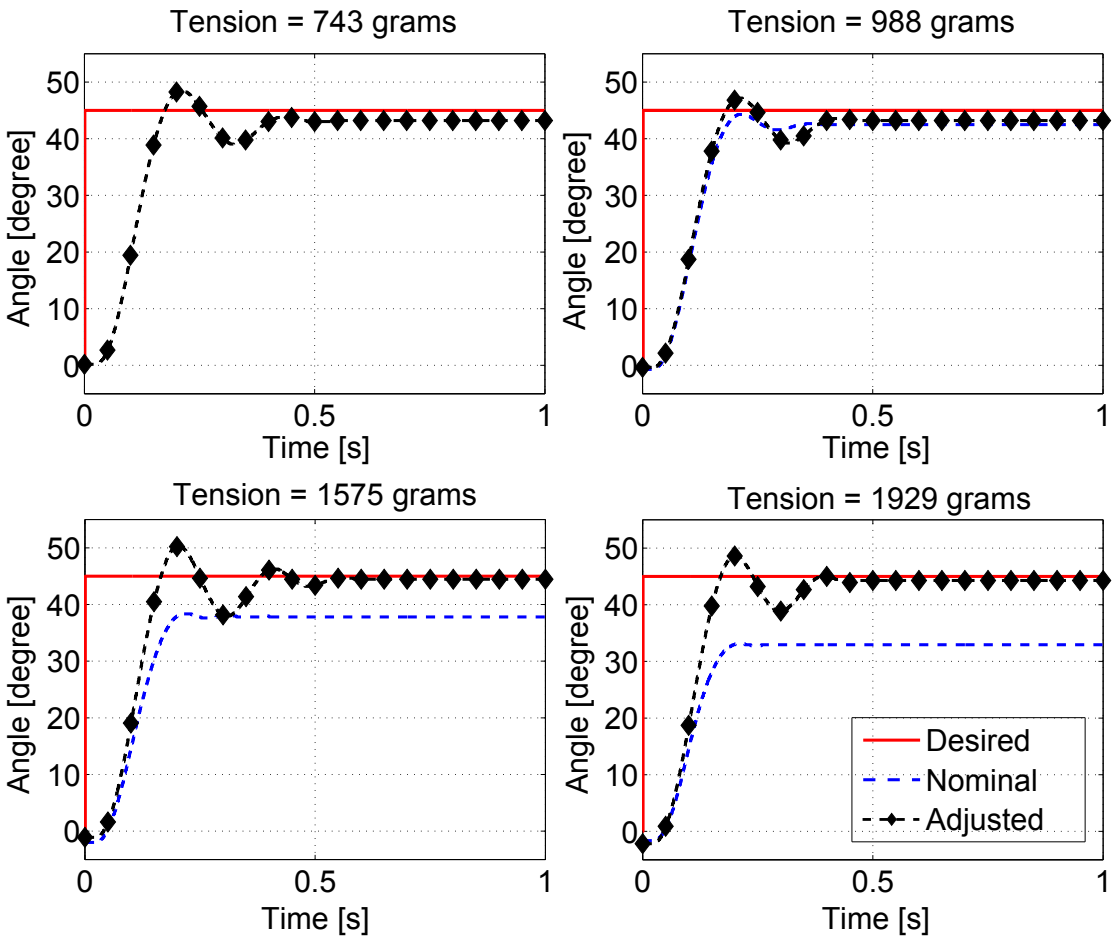

Figure 5.1: Step response of pulleyboard with nominal and adjusted parameters at four tension levels. The time axis is magnified to better illustrate the results.

Fig. 5.2 shows the steady state error in step response for all tensions with and without adjusting for parameter changes. When the nominal parameter values based on the 743 *gr* tension were used, the steady state error increased as a function of cable tension. When the estimated parameters were used to adjust the system, the steady state error did not increase as a function of cable tension. The link encoder readings were solely used for recording the data and not for control. The state estimates from the UKF are obtained using only input-output data (i.e. motor encoder measurements and motor torque commands).

In addition to the step response, the system was evaluated in following recorded hand trajectories. Similar to step response experiments, the link angle in response to this trajectory was obtained

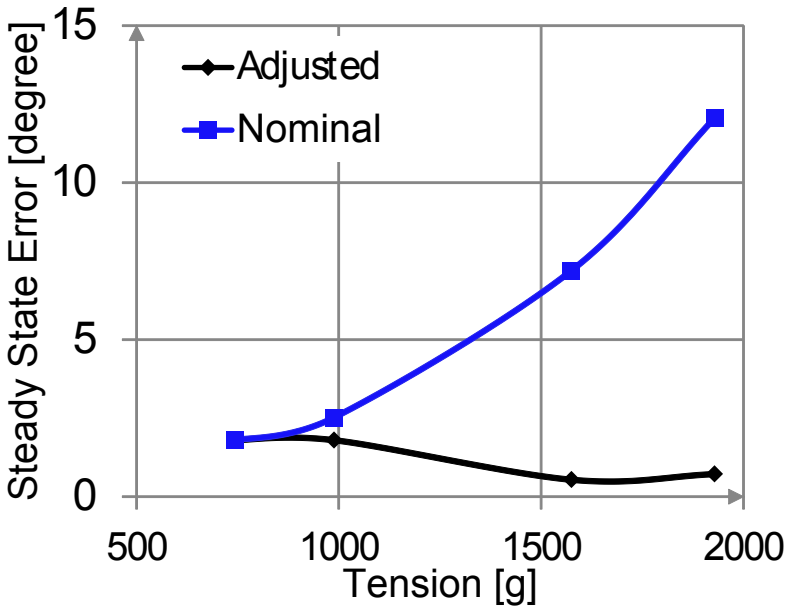

Figure 5.2: Steady state error in response to a 45 degree step command at four tension levels with nominal and adjusted parameters.

at each tension level with and without adjusting the parameters. The desired trajectory, along with the fixed and adjusted responses are displayed in Fig. 5.3 for the largest test tension of 1929 grams. Similar responses were obtained at all tension levels. With some delay, both responses follow the desired trajectory. However, the adjusted response has a smaller error in reaching the peaks. The error between the fixed response and the desired response at each peak, $|E_{p-Nom}|$, and the error between the adjusted response and the desired response at each peak, $|E_{p-Adj}|$, are defined to quantify this metric. Fig. 5.4 displays the average of these errors over the thirteen peaks of the trajectory, for all four tension levels. As the tension deviates from the nominal value, the peak error increases when using the nominal parameters. However, the errors in the adjusted response remain approximately the same, resembling the step response results of Fig. 5.2.

These results show the importance of compensating for variations in cable tension in order to maintain tracking performance. It may not always be feasible to alter the controller to achieve the desired performance. In such instances it is easier to keep the tension at the nominal value or nominal range. In the absence of cable tension sensors an estimate of the cable tension can be used

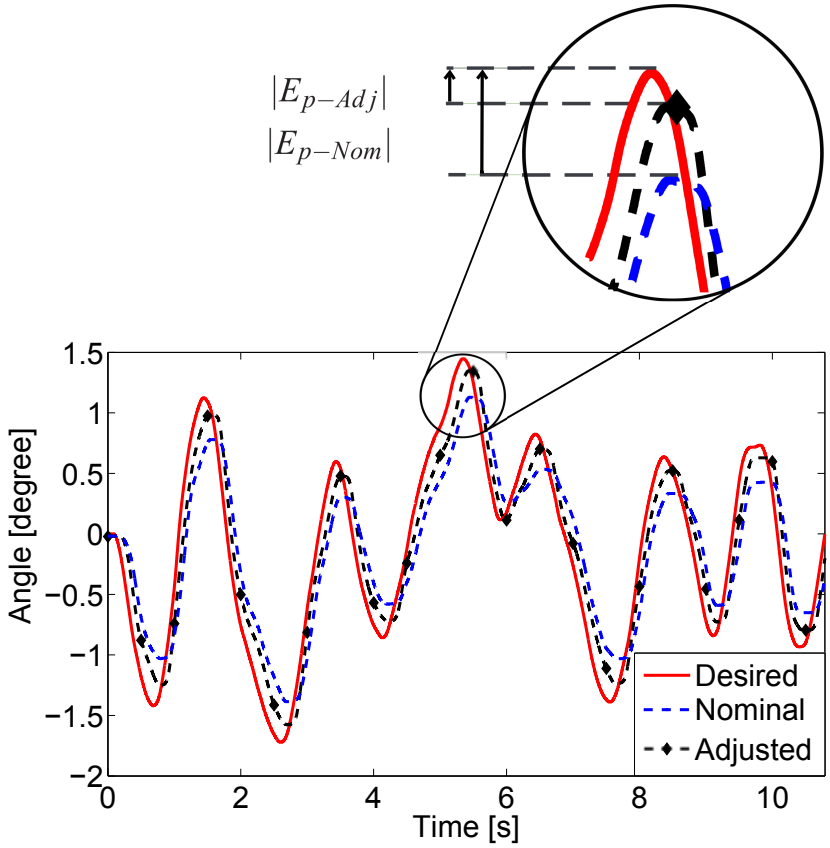

Figure 5.3: Time response of the pulleyboard to a recorded trajectory with nominal and adjusted parameters at cable tension of 1929 grams.

to ensure the system is operating in the desired range. Section 5.2 investigates this approach.

5.2 Tension Estimation

The above results suggests that changes in the cable tension are reflected in the estimated parameters of the system. We propose a method to estimate cable tension from the estimated parameters.

To this end, the cable tension was varied in steps and measured. At each step, system parameters were estimated and a curve was fit to the points. Among the parameters, cable stiffness showed strong correlation with tension. Compared to friction coefficients, it was less affected by external conditions such as lubrication of the pulleys. Therefore, cable stiffness estimates were used here for tension estimation.

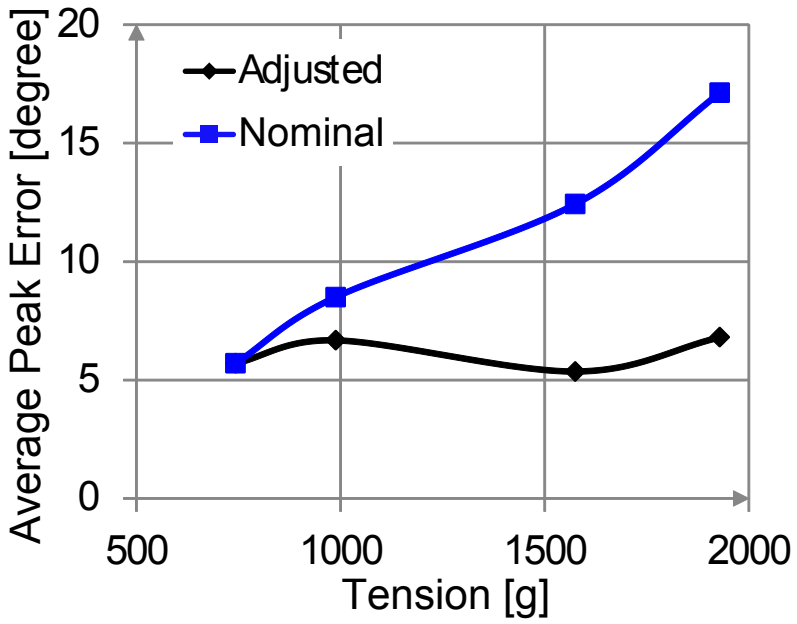

Figure 5.4: Average peak error in following the recorded hand trajectory of Fig. 5.3 at four tension levels with fixed and adjusted parameters.

Fig. 5.5.(a) shows the estimated stiffness for different levels of tension measured. A logarithmic fit along with the points used to derive the curve are displayed. The tension measurements varied with the position of the link. To compensate for this issue, tension values were measured for a range of different positions, at each tension level. The midpoint of the range was used in deriving the curve. The grey area on the curve represents these variations. To evaluate the mapping, the tension was then changed in four levels and parameters were estimated at each level from the curve. Fig. 5.5.(b) shows the tension measured and the tension estimated from the mapping. The estimated and measured tensions are presented in Table 5.1.

5.3 Conclusion

This chapter demonstrated a method to estimate and compensate for the changes of cable tension in the control of cable driven mechanisms. Only motor angle information is directly measured. The UKF is used for the estimation of system parameters under different tension conditions. Changes in cable tension are captured in the estimated parameters. The UKF is also used on-line, to obtain

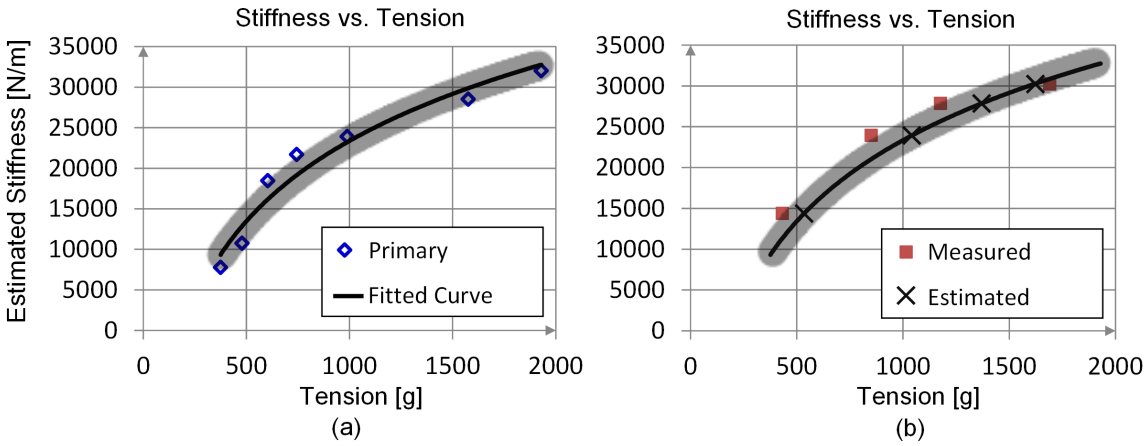

Figure 5.5: (a) Fitted logarithmic curve and points used in generating it. The shaded region represents the force gauge tension measurement error. (b) Estimating the tension from stiffness. □ represents the tension measured and × represents the tension estimated from the curve.

Measured [grams]	410 – 455	822 – 880	1152 – 1202	1677 – 1706
Estimated [grams]	534	1041	1370	1623

Table 5.1: Measured tension and estimated tension from the estimated stiffness. The two numbers reported as measured values are due to variations in the gauge reading.

estimates of the system state (based upon the off-line parameter estimates). The state estimates and parameter estimates (when they are also made) are used by a model predictive controller to generate appropriate control actions. A comparison is made of the controller performance with and without utilizing the adjusted parameters. It is shown that variations in tension affect the performance of the model predictive controller. Improvement in control performance after correcting the parameters of the model under different tensions was demonstrated.

This particular system exhibited a correlation between tension and cable stiffness. This correlation was used as a means of indirectly estimating the cable tension, based upon estimated stiffness parameters.

Chapter 6

DISCUSSION AND OPEN QUESTIONS

The previous chapters demonstrated a successful implementation of real-time virtual fixtures derived from streaming point clouds. This chapter presents ideas on improving the performance of the proposed methods. To this end, performance measures as proposed in the literature are presented, the major sources of error in our virtual fixture implementation are identified and a method to mitigate these effects is proposed. Evaluation of these methods is the subject of future work.

6.1 Performance Measures

Several metrics have been proposed in [108] for evaluating the effectiveness of virtual fixtures. These measures include:

Safety: The robot should always stay outside the forbidden region. *Safety* measure is defined as the maximum penetration of the forbidden region by the robot.

Submittance: Although the virtual fixture should prevent the robot from entering the forbidden region, it should not limit the robot from reaching other areas in the workspace. *Submittance* is a measure of how close the robot can get to the forbidden region and is defined as the minimum distance between the robot and the forbidden region when the robot fails to reach the forbidden region.

In addition, similar to any other teleoperation system the remote robot should follow the master outside the forbidden region, a measure referred to as *tracking*.

6.2 Sources of Error

The following factors are the main contributors to error and uncertainty in our virtual fixture implementation:

RGB-D camera: Camera calibration can eliminate distortions in the point cloud as well as misalignment between the color and depth camera. However, measurements of the depth camera, like any measurement, are subject to random error. In addition, the resolution of the camera is limited due to quantization of the data. The accuracy and resolution of Kinect data was investigated in [109, 110]. An error model for depth measurements, Z, and planar measurements, X, Y, was derived in [109]. It was shown that the standard deviation of measurements is proportional to the square distance from the sensor to the object.

$$\sigma_X, \sigma_Y, \sigma_Z \propto Z^2 \tag{6.1}$$

In addition, the depth resolution (i.e. the minimum depth difference that can be measured) was similarly found to be a quadratic function of the distance from the sensor ranging from 2 *mm* at the depth of 1 *m* to 7 *cm* at the depth of 5 *m* [109].

$$\Delta_X, \Delta_Y, \Delta_Z \propto Z^2 \tag{6.2}$$

Latency The point cloud obtained by RGB-D cameras has an inherent latency due to the frame rate and processing time of the camera. According to [111], the latency for the Kinect is approximately 160 *ms*. Point cloud haptic rendering methods such as the one presented in Chapter 2 are not severely affected by the latency due to their one way nature. In other words, the user feels the remote environment with a delay but since information is not transmitted to the remote world from the user it does not pose a problem for rendering. In these applications the frame rate of the camera is the limiting factor. Since no information is captured between the two frames a pop-through may occur due to the motion of the objects in the environment. Methods such as the one proposed in [27] have been used to predict the velocity of point cloud and move the proxy accordingly to compensate for the effects of the frame rate.

In contrast to haptic rendering methods, the accuracy of virtual fixtures can be severely affected by the latency because of the bilateral flow of information between the user and the environment. Due to this latency, the captured point cloud may no longer be an accurate representation of the environment at that time instant. As a result, the robot may end up inside a forbidden region before it is captured by the camera. In addition to the latency of the camera,

any latency in the network connecting the master and remote robot, as well as the latency of the controller contribute to this problem.

Assuming a constant latency of T, and a velocity estimate of $\hat{v}$ for the moving objects in the environment in between the frames, the error in the position of the forbidden region can be calculated as $v_m \times T$.

Robot Point cloud registration errors, imperfect control actions, kinematic uncertainties and sensor errors may reduce the tracking performance of the robot. This tracking error can drive the robot inside a forbidden region and reduce the accuracy of virtual fixture implementation. The sum of these errors can be represented an error in the position of the robot denoted by $\Delta X_r = X_r^* - X_r$, where X_r^* is the desired position of the robot and X_r is the actual position of the robot.

The above factors can be combined into a mathematical model of the uncertainty in the virtual fixtures. The model can represent this uncertainty as a random variable for each point in space and can be a function of the distance to the sensor, velocity of the objects and the pose of the robot. The effect of each source of error on the performance can be analysed for each virtual fixture architecture proposed in Chapter 3 and is beyond this work. However, we propose a method to mitigate the effect of errors on the safety measure of virtual fixtures.

6.3 Adaptive Virtual Fixtures

As was presented in Chapter 3, the virtual fixture for each point is a sphere with radius r_f. This radius was assigned arbitrarily and was kept constant in our implementation. However, there is no reason to limit the implementation to a fixed radius. In addition, the radius can be proportional to the uncertainty of the virtual fixture discussed above. Assume that the total error for position of point i in space is represented by a random variable with multivariate normal distribution, $\mathcal{N}(0,\Sigma)$[1], where Σ is the 3×3 covariance matrix. In order to compensate for the uncertainty in the position of point

[1] In its general form the normal distribution can have a non-zero mean, $\mathcal{N}(\mu,\Sigma)$, in which case the point can be shifted by μ to achieve a zero mean distribution.

i, the virtual fixture radius for this point can be assigned to be proportional to the largest eigenvalue of Σ. This way the virtual fixture radius will increase as the uncertainty increases[2].

By adaptively changing the radius of virtual fixture the safety measure can be improved. However, the submittance measure will inevitably deteriorate due to the inherent trade-off between safety and submittance in the presence of uncertainty.

6.4 Occlusions

Occlusions in the point cloud, either caused by the robot or other objects, pose a potential problem to the rendering of virtual fixtures from point clouds. If the proxy falls through a shadow caused by the occlusion the robot may violate the constraint and enter the forbidden region. Several techniques could be used to reduce the effect of occlusions. One approach is to use multiple cameras. The modular nature of our rendering techniques allows for the processing of multiple point clouds captured by cameras at different angles to reduce these effects. Our preliminary results suggest that even though point clouds captured by multiple cameras reduces the effect of occlusions, the interference between multiple cameras significantly affects the quality of the captured point clouds. This interference can be mitigated by vibrating the cameras as suggested in [112, 113]. The vibration eliminates most of the crosstalk between the devices but does not significantly affect the point cloud captured by each device since the IR source and camera move in harmony.

Another approach is to track the object or parts of the object that are blocked from the camera. Preoperative images can be used to track rigid objects. Probabilistic methods for tracking deformable objects such as the one presented in [114] can be used to reduce the effect of occlusions. The effectiveness of these methods for virtual fixtures remains to be investigated.

6.5 Extension of the Control Techniques to Higher Degrees of Freedom

The control and estimation techniques of Chapter 4 and 5 were demonstrated on a 1 degree of freedom platform. These methods can be extended to higher degrees of freedom of a surgical robot by taking advantage of the polymorphism of the software. The developed code templates allow

[2] A more accurate compensation would result by extending the virtual fixture from a sphere to an ellipsoid, compensating for uncertainty in each direction accordingly.

for the dynamic equations to be easily modified to reflect the characteristics of any robot. It is anticipated that by extending these methods to multiple degrees of freedom of a surgical instrument an improved control performance in contact with tissue and under various cable tensions can be achieved.

6.6 Broader Impact

Robotic surgery has become the standard of operation for several procedures and is rapidly finding its place in many others. The introduction of affordable and reliable sensing technologies together with the advances in the processing power of computers and the development of fast and novel algorithms have enabled new capabilities in robotic surgery not known previously. Virtual fixtures are among these technologies that can improve the speed and safety of operation. By protecting delicate structures from touch by the robot, virtual fixtures offload some of the operator's mental and physical workload onto the computer, freeing up those resources to be used for complex decision making. As new advancements are made in object recognition, image segmentation and development of depth endoscopes, the virtual fixture technology can find its way into the operating room in near future.

BIBLIOGRAPHY

[1] L. B. Rosenberg, "Virtual fixtures: Perceptual tools for telerobotic manipulation," in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*. IEEE, 1993, pp. 76–82.

[2] B. Rocco, D.-V. Matei, S. Melegari, J. C. Ospina, F. Mazzoleni, G. Errico, M. Mastropasqua, L. Santoro, S. Detti, and O. De Cobelli, "Robotic vs open prostatectomy in a laparoscopically naive centre: a matched-pair analysis," *BJU international*, vol. 104, no. 7, pp. 991–995, 2009.

[3] V. Ficarra, G. Novara, S. Fracalanza, C. DElia, S. Secco, M. Iafrate, S. Cavalleri, and W. Artibani, "A prospective, non-randomized trial comparing robot-assisted laparoscopic and retropubic radical prostatectomy in one european institution," *BJU international*, vol. 104, no. 4, pp. 534–539, 2009.

[4] C. T. Ho, *Robot-assisted surgery compared with open surgery and laparoscopic surgery: clinical effectiveness and economic analyses*. Canadian Agency for Drugs and Technologies in Health= Agence canadienne des médicaments et des technologies de la santé, 2011.

[5] L. Hohwu, O. Akre, K. V. Pedersen, M. Jonsson, C. V. Nielsen, and O. Gustafsson, "Open retropubic prostatectomy versus robot-assisted laparoscopic prostatectomy: A comparison of length of sick leave," *Scandinavian Journal of Urology and Nephrology*, vol. 43, no. 4, pp. 259–264, 2009.

[6] M. Menon, A. Tewari, B. Baize, B. Guillonneau, and G. Vallancien, "Prospective comparison of radical retropubic prostatectomy and robot-assisted anatomic prostatectomy: the vattikuti urology institute experience," *Urology*, vol. 60, no. 5, pp. 864–868, 2002.

[7] J. Miller, A. Smith, E. Kouba, E. Wallen, and R. S. Pruthi, "Prospective evaluation of short-term impact and recovery of health related quality of life in men undergoing robotic assisted laparoscopic radical prostatectomy versus open radical prostatectomy," *The Journal of urology*, vol. 178, no. 3, pp. 854–859, 2007.

[8] S. Carlsson, A. E. Nilsson, M. C. Schumacher, M. N. Jonsson, D. S. Volz, G. Steineck, and P. N. Wiklund, "Surgery-related complications in 1253 robot-assisted and 485 open retropubic radical prostatectomies at the karolinska university hospital, sweden," *Urology*, vol. 75, no. 5, pp. 1092–1097, 2010.

[9] D. Halliday, S. Lau, Z. Vaknin, C. Deland, M. Levental, E. McNamara, R. Gotlieb, R. Kaufer, J. How, E. Cohen *et al.*, "Robotic radical hysterectomy: comparison of outcomes and cost," *Journal of Robotic Surgery*, vol. 4, no. 4, pp. 211–216, 2010.

[10] M. C. Bell, J. Torgerson, U. Seshadri-Kreaden, A. W. Suttle, and S. Hunt, "Comparison of outcomes and cost for endometrial cancer staging via traditional laparotomy, standard laparoscopy and robotic techniques," *Gynecologic oncology*, vol. 111, no. 3, pp. 407–411, 2008.

[11] T. J. Bivalacqua, P. M. Pierorazio, and L.-M. Su, "Open, laparoscopic and robotic radical prostatectomy: optimizing the surgical approach," *Surgical oncology*, vol. 18, no. 3, pp. 233–241, 2009.

[12] Y. Lotan, J. A. Cadeddu, and M. T. Gettman, "The new economics of radical prostatectomy: cost comparison of open, laparoscopic and robot assisted techniques," *The Journal of urology*, vol. 172, no. 4, pp. 1431–1435, 2004.

[13] S. Breitenstein, A. Nocito, M. Puhan, U. Held, M. Weber, and P.-A. Clavien, "Robotic-assisted versus laparoscopic cholecystectomy," *Annals of Surgery*, vol. 247, no. 6, pp. 987–993, 2008.

[14] W. JD, A. CV, L. SN, and et al, "Robotically assisted vs laparoscopic hysterectomy among women with benign gynecologic disease," *JAMA*, vol. 309, no. 7, pp. 689–698, 2013.

[15] A. D'Annibale, E. Morpurgo, V. Fiscon, P. Trevisan, G. Sovernigo, C. Orsini, and D. Guidolin, "Robotic and laparoscopic surgery for treatment of colorectal diseases," *Diseases of the Colon & Rectum*, vol. 47, no. 12, pp. 2162–2168, 2004.

[16] R. E. Link, S. B. Bhayani, and L. R. Kavoussi, "A prospective comparison of robotic and laparoscopic pyeloplasty," *Annals of surgery*, vol. 243, no. 4, p. 486, 2006.

[17] A. Rawlings, J. Woodland, R. Vegunta, and D. Crawford, "Robotic versus laparoscopic colectomy," *Surgical endoscopy*, vol. 21, no. 10, pp. 1701–1708, 2007.

[18] P. Kazanzides, B. D. Mittelstadt, B. L. Musits, W. L. Bargar, J. F. Zuhars, B. Williamson, P. W. Cain, and E. J. Carbone, "An integrated system for cementless hip replacement," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 14, no. 3, pp. 307–313, 1995.

[19] W. L. Bargar, A. Bauer, and M. Börner, "Primary and revision total hip replacement using the robodoc (r) system," *Clinical orthopaedics and related research*, vol. 354, pp. 82–91, 1998.

[20] D. Camarillo, T. Krummel, J. Salisbury *et al.*, "Robotic technology in surgery: past, present, and future," *The American Journal of Surgery*, vol. 188, pp. 2–15, 2004.

[21] P. Hokayem and M. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, pp. 2035–2057, 2006.

[22] A. Okamura, "Methods for haptic feedback in teleoperated robot-assisted surgery," *Industrial Robot: An International Journal*, vol. 31, no. 6, pp. 499–508, 2004.

[23] G. Tholey, A. Pillarisetti, W. Green, and J. Desai, "Design, development, and testing of an automated laparoscopic grasper with 3-d force measurement capability," *Medical Simulation*, pp. 38–48, 2004.

[24] J. Dargahi, M. Parameswaran, and S. Payandeh, "A micromachined piezoelectric tactile sensor for an endoscopic grasper-theory, fabrication and experiments," *Microelectromechanical Systems, Journal of*, vol. 9, no. 3, pp. 329–335, 2000.

[25] B. Kubler, U. Seibold, and G. Hirzinger, "Development of actuated and sensor integrated forceps for minimally invasive robotic surger," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 1, no. 3, pp. 96–107, 2005.

[26] C. Basdogan and M. Srinivasan, "Haptic rendering in virtual environments," *Handbook of virtual environments*, pp. 117–134, 2002.

[27] F. Rydén and H. J. Chizeck, "A proxy method for real-time 3-DOF haptic rendering of streaming point cloud data," *Haptics, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2013, in press.

[28] W. McNeely, K. Puterbaugh, and J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 401–408.

[29] D. Levin, "The approximation power of moving least-squares," *Mathematics of computation*, vol. 67, no. 224, pp. 1517–1532, 1998.

[30] A. Adamson and M. Alexa, "Approximating and intersecting surfaces from points," in *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. Eurographics Association, 2003, pp. 230–239.

[31] J. Lee and Y. Kim, "Haptic rendering of point set surfaces," in *EuroHaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2007. Second Joint*. IEEE, 2007, pp. 513–518.

[32] N. El-Far, N. Georganas, and A. El Saddik, "An algorithm for haptically rendering objects described by point clouds," in *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on*, may 2008, pp. 001 443 –001 448.

[33] J. Cha, S. Kim, I. Oakley, J. Ryu, and K. Lee, "Haptic interaction with depth video media," *Advances in Mulitmedia Information Processing-PCM 2005*, pp. 420–430, 2005.

[34] D. Ruspini, K. Kolarov, and O. Khatib, "The haptic display of complex graphical environments," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 345–352.

[35] F. Rydén, H. J. Chizeck, S. N. Kosari, H. King, and B. Hannaford, "Using kinect and a haptic interface for implementation of real-time virtual fixtures," in *Proceedings of the 2nd Workshop on RGB-D: Advanced Reasoning with Depth Cameras (in conjunction with RSS 2011)*, 2011.

[36] C. Zilles and J. 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*, vol. 3. IEEE, 1995, pp. 146–151.

[37] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments," in *the 12th International Symposium on Experimental Robotics (ISER)*, 2010.

[38] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns," US Patent Application, Oct 02, 2008, uS 2008/0240502 A1.

[39] P. S. LTD, "Reference design," January 2011, http://www.primesense.com/?p=514.

[40] J. Colgate and G. Schenkel, "Passivity of a class of sampled-data systems: Application to haptic interfaces," *Journal of robotic systems*, vol. 14, no. 1, pp. 37–47, 1997.

[41] M. Kitagawa, D. Dokko, A. Okamura, and D. Yuh, "Effect of sensory substitution on suture-manipulation forces for robotic surgical systems," *Journal of Thoracic and Cardiovascular Surgery*, vol. 129, no. 1, pp. 151–158, 2005.

[42] N. V. Navkar, Z. Deng, D. J. Shah, K. E. Bekris, and N. V. Tsekos, "Visual and force-feedback guidance for robot-assisted interventions in the beating heart with real-time MRI," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 689–694.

[43] S. Payandeh and Z. Stanisic, "On application of virtual fixtures as an aid for telemanipulation and training," in *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on*, 2002, pp. 18 –23.

[44] K. Kwok, G. Mylonas, L. Sun, M. Lerotic, J. Clark, T. Athanasiou, A. Darzi, and G. Yang, "Dynamic active constraints for hyper-redundant flexible robots," *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009*, pp. 410–417, 2009.

[45] A. Bettini, P. Marayong, S. Lang, A. M. Okamura, and G. D. Hager, "Vision-assisted control for manipulation using virtual fixtures," *Robotics, IEEE Transactions on*, vol. 20, no. 6, pp. 953–966, 2004.

[46] P. Marayong and A. Okamura, "Speed-accuracy characteristics of human-machine cooperative manipulation using virtual fixtures with variable admittance," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 46, no. 3, pp. 518–532, 2004.

[47] M. Li, M. Ishii, and R. H. Taylor, "Spatial motion constraints using virtual fixtures generated by anatomy," *Robotics, IEEE Transactions on*, vol. 23, no. 1, pp. 4 –19, feb. 2007.

[48] D. Kragic, P. Marayong, M. Li, A. Okamura, and G. Hager, "Human-machine collaborative systems for microsurgical applications," *The International Journal of Robotics Research*, vol. 24, no. 9, pp. 731–741, 2005.

[49] S. Park, R. Howe, and D. Torchiana, "Virtual fixtures for robotic cardiac surgery," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001*. Springer, 2001, pp. 1419–1420.

[50] B. Davies, M. Jakopec, S. Harris, F. Rodriguez y Baena, A. Barrett, A. Evangelidis, P. Gomes, J. Henckel, and J. Cobb, "Active-constraint robotics for surgery," *Proceedings of the IEEE*, vol. 94, no. 9, pp. 1696–1704, 2006.

[51] M. Peshkin, J. Colgate, W. Wannasuphoprasit, C. Moore, R. Gillespie, and P. Akella, "Cobot architecture," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 4, pp. 377–390, 2001.

[52] B. Becker, R. MacLachlan, G. Hager, and C. Riviere, "Handheld micromanipulation with vision-based virtual fixtures," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 4127 –4132.

[53] M. Dewan, P. Marayong, A. Okamura, and G. Hager, "Vision-based assistance for ophthalmic micro-surgery," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2004*, ser. Lecture Notes in Computer Science, C. Barillot, D. Haynor, and P. Hellier, Eds. Springer Berlin / Heidelberg, 2004, vol. 3217, pp. 49–57.

[54] M. Li, A. Kapoor, and R. Taylor, "A constrained optimization approach to virtual fixtures," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, aug. 2005, pp. 1408 – 1413.

[55] J. W. Park, J. Choi, Y. Park, and K. Sun, "Haptic virtual fixture for robotic cardiac catheter navigation," *Artificial Organs*, vol. 35, no. 11, pp. 1127–1131, 2011.

[56] T. Yamamoto, N. Abolhassani, S. Jung, A. Okamura, and T. Judkins, "Augmented reality and haptic interfaces for robot-assisted surgery," *The International Journal of Medical Robotics and Computer Assisted Surgery*, 2011.

[57] T. L. Gibo, L. N. Verner, D. D. Yuh, and A. M. Okamura, "Design considerations and human-machine performance of moving virtual fixtures," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, may 2009, pp. 671 –676.

[58] J. Ren, R. Patel, K. McIsaac, G. Guiraudon, and T. Peters, "Dynamic 3-D virtual fixtures for minimally invasive beating heart procedures," *Medical Imaging, IEEE Transactions on*, vol. 27, no. 8, pp. 1061 –1070, aug. 2008.

[59] B. Hannaford, J. Rosen, D. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. Nia Kosari, and L. White, "Raven-II: an open platform for surgical roboticsresearch," *Biomedical Engineering, IEEE Transactions on*, no. 99, 2012.

[60] H. King, B. Hannaford, K.-W. Kwok, G.-Z. Yang, P. Griffiths, A. Okamura, I. Farkhatdinov, J.-H. Ryu, G. Sankaranarayanan, V. Arikatla, K. Tadano, K. Kawashima, A. Peer, T. Schauss, M. Buss, L. Miller, D. Glozman, J. Rosen, and T. Low, "Plugfest 2009: Global interoperability in telerobotics and telemedicine," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, may 2010, pp. 1733 –1738.

[61] L. Rosenberg, "The use of virtual fixtures to enhance operator performance in time delayed teleoperation." DTIC Document, Tech. Rep., 1993.

[62] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[63] I. Kato, M. Billinghurst, and I. Poupyrev, "ARtoolkit user manual, version 2.33," *Human Interface Technology Lab, University of Washington*, 2000.

[64] F. Rydén and H. Chizeck, "Forbidden-region virtual fixtures from streaming point clouds: Remotely touching and protecting a beating heart," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012.

[65] J. J. Abbott, "Virtual fixtures for bilateral telemanipulation," Ph.D. dissertation, Baltimore, MD, USA, 2006, AAI3197103.

[66] P. Marayong, M. Li, A. Okamura, and G. Hager, "Spatial motion constraints: Theory and demonstrations for robot guidance using virtual fixtures," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1954–1959.

[67] H. King, S. Nia Kosari, B. Hannaford, and J. Ma, "Kinematic analysis of the Raven-II research surgical robot platform," University of Washington Electrical Engineering Department Technical Report, vol. 2012-0006, June 29, 2012, Tech. Rep.

[68] S. Kosari, S. Ramadurai, H. Chizeck, and B. Hannaford, "Robotic compression of soft tissue," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4654–4659.

[69] Y. Fung, *Biomechanics: mechanical properties of living tissues*. Springer, 1993, vol. 12.

[70] N. Narayanan, A. Bonakdar, J. Dargahi, M. Packirisamy, and R. Bhat, "Design and analysis of a micromachined piezoelectric sensor for measuring the viscoelastic properties of tissues in minimally invasive surgery," *Smart materials and structures*, vol. 15, p. 1684, 2006.

[71] T. Yamamoto, B. Vagvolgyi, K. Balaji, L. Whitcomb, and A. Okamura, "Tissue property estimation and graphical display for teleoperated robot-assisted surgery," *IEEE International Conference on Robotics and Automation*, pp. 4239–4245, 2009.

[72] A. Annaswamy and M. Srinivasan, "The role of compliant fingerpads in grasping and manipulation: Identification and control," *Essays on mathematical robotics*, vol. 104, p. 1, 1998.

[73] X. Yu, H. Chizeck, and B. Hannaford, "Comparison of transient performance in the control of soft tissue grasping," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1809–1814, 2007.

[74] L. Love and W. Book, "Environment estimation for enhanced impedance control," vol. 2, pp. 1854–1859, 1995.

[75] S. Lin and K. Yae, "Identification of unknown payload and environmental parameters for robot compliant motion," pp. 2952–2956, 1992.

[76] S. Patil and R. Alterovitz, "Toward automated tissue retraction in robot-assisted surgery," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2088–2094.

[77] G. Picinbono, H. Delingette, and N. Ayache, "Nonlinear and anisotropic elastic soft tissue models for medical simulation," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 1370–1375.

[78] G. Gilardi and I. Sharf, "Literature survey of contact dynamics modelling," *Mechanism and Machine Theory*, vol. 37, no. 10, pp. 1213–1239, 2002.

[79] K. Hunt and F. Crossley, "Coefficient of restitution interpreted as damping in vibroimpact," *Journal of Applied Mechanics*, vol. 42, p. 440, 1975.

[80] N. Diolaiti, C. Melchiorri, and S. Stramigioli, "Contact impedance estimation for robotic systems," *Robotics, IEEE Transactions on*, vol. 21, no. 5, pp. 925–935, 2005.

[81] L. Ljung, *Theory and Practice of Recursive Identification*. The MIT Press, 1983.

[82] A. Haddadi and K. Hashtrudi-Zaad, "A new method for online parameter estimation of hunt-crossley environment dynamic models," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 981–986.

[83] M. N. S. B. H. P. Roan, A.S. Wright, "An instrumented minimally invasive surgical tool: Design and calibration," *Applied Bionics and Biomechanics, Special Issue on Surgical Robotics*, pp. 1–21, 2010.

[84] J. Rosen, J. D. Brown, L. Chang, M. Barreca, M. Sinanan, and B. Hannaford, "The bluedragon-a system for measuring the kinematics and dynamics of minimally invasive surgical tools in-vivo," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1876–1881.

[85] D. Pawluk, J. Son, P. Wellman, W. Peine, and R. Howe, "A distributed pressure sensor for biomechanical measurements." *Journal of biomechanical engineering*, vol. 120, no. 2, p. 302, 1998.

[86] G. Tholey and J. P. Desai, "A modular, automated laparoscopic grasper with three-dimensional force measurement capability," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 250–255.

[87] P. Hacksel and S. Salcudean, "Estimation of environment forces and rigid-body velocities using observers," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 931–936.

[88] S. Lin, "Force sensing using kalman filtering techniques for robot compliant motion control," *Journal of Intelligent and Robotic Systems*, vol. 18, no. 1, pp. 1–16, 1997.

[89] J. Jung, J. Lee, and K. Huh, "Robust contact force estimation for robot manipulators in three-dimensional space," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 220, no. 9, pp. 1317–1327, 2006.

[90] T. Alja'afreh, "Controlling force variations during soft-tissue grasping," *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 223, no. 6, pp. 749–753, 2009.

[91] G. Tholey and A. Castellanos, "Evaluating the role of vision and force feedback in minimally invasive surgery: New automated laparoscopic grasper and a case study," *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003*, pp. 198–205, 2003.

[92] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[93] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Proceedings of SPIE*, vol. 3068, 1997, p. 182.

[94] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2005.

[95] E. Wan and R. Van Der Merwe, "The Unscented Kalman Filter," *Kalman filtering and neural networks*, vol. 5, no. 2007, pp. 221–280, 2001.

[96] G. Prisco and M. Bergamasco, "Dynamic modelling of a class of tendon driven manipulators," *International Conference on Advanced Robotics, 1997*, pp. 893–899.

[97] R. Beira, A. Sengul, M. Hara, P. Schoeneich, and H. Bleuler, "Tendon-based transmission for surgical robotics: Systematic experimental friction modeling," *International Conference on Applied Bionics and Biomechanics.*, 2010.

[98] E. Naerum, H. H. King, and B. Hannaford, "Robustness of the Unscented Kalman Filter for state and parameter estimation in an elastic transmission," *Proceedings of Robotics: Science and Systems*, 2009.

[99] S. Julier and J. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," *Robotics Research Group, Department of Engineering Science, University of Oxford, Oxford, OC1 3PJ United Kingdom, Tech. Rep*, 1996.

[100] J. Mattingley, Y. Wang, and S. Boyd, "Code generation for receding horizon control," in *Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 985–992.

[101] Y. Tassa, T. Erez, and E. Todorov, "Fast model predictive control for reactive robotic swimming."

[102] D. H. Jacobson and D. Q. Mayne, "Differential dynamic programming," 1970.

[103] J. Morimoto, G. Zeglin, and C. G. Atkeson, "Minimax differential dynamic programming: Application to a biped walking robot," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2003, pp. 1927–1932.

[104] L. Rall and G. Corliss, "An introduction to automatic differentiation," *Computational Differentiation: Techniques, Applications, and Tools*, pp. 1–17, 1996.

[105] B. Bell, "CppAD: a package for C++ algorithmic differentiation, version 20110711, COIN-OR, http://www.coin-or.org/CppAD."

[106] E. Todorov and Y. Tassa, "Iterative local dynamic programming," *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, 2009*, pp. 90–95, 2009.

[107] S. Nia Kosari, S. Ramadurai, H. J. Chizeck, and B. Hannaford, "Control and tension estimation of a cable driven mechanism under different tensions," in *37th Mechanisms and Robotics Conference (MR)*. ASME, 2013.

[108] J. J. Abbott and A. M. Okamura, "Virtual fixture architectures for telemanipulation," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 2798–2805.

[109] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[110] J. Smisek, M. Jancosek, and T. Pajdla, "3d with kinect," in *Consumer Depth Cameras for Computer Vision*. Springer, 2013, pp. 3–25.

[111] E. N. Saba, E. C. Larson, and S. N. Patel, "Dante vision: In-air and touch gesture sensing for natural surface interaction with combined depth and thermal cameras," in *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 167–170.

[112] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim, "Shake'n'sense: reducing interference for overlapping structured light depth cameras," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*. ACM, 2012, pp. 1933–1936.

[113] A. Maimone and H. Fuchs, "Reducing interference between multiple structured light depth sensors using motion," in *Virtual Reality Workshops (VR), 2012 IEEE*. IEEE, 2012, pp. 51–54.

[114] J. Schulman, A. Lee, J. Ho, and P. Abbeel, “Tracking deformable objects with point clouds,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013.

Appendix A

NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS

The dynamic equations of the system described in Chapter 4.3 are in continuous time. However, our UKF and DDP implementations require discrete-time representations of the system. This objective can be achieved through numerical integration methods.

Consider the continuous-time state-space description given by

$$\dot{x(t)} = G_c(x(t), u(t), t) \tag{A.1}$$

The discrete-time equivalent of the system can be derived by

$$x[k+1] = G_d(x[k], u[k], k) \tag{A.2}$$

where

$$G_d(x[k], u[k], k) = x[k] + \int_0^{\Delta t} G_c(x(t), u(t), t)dt \tag{A.3}$$

and Δt denotes the sampling time of the system. In practice, the above integration is performed numerically for nonlinear systems.

Several methods exist for carrying out the above integration numerically. These methods can be divided in two categories: *variable step* and *fixed step*. Fixed step methods solve the problem by proceeding forward in time in fixed steps. Variable step methods change the step size to satisfy error tolerance requirements. Variable step methods can provide more accurate results, however, the computation time required for solving the problem can change from one time step to another depending on the dynamics at that particular step. Fixed step solvers have fixed execution time and are more appropriate for real-time implementations where deterministic timing is required.

The fixed step solvers can be further divided into *Explicit* and *Implicit*. Explicit methods provide an explicit solution for calculating $x[k+1]$ from $x[k],\ u[k]$, i.e.

$$x[k+1] = f(x[k], u[k]). \tag{A.4}$$

Implicit methods generate an equation relating $x[k+1]$ and $x[k],\ u[k]$ that needs to be solved for $x[k+1]$, i.e.

$$f(x[k+1], x[k], u[k]) = 0. \tag{A.5}$$

Implicit methods therefore are computationally more expensive. However, with a fixed step size implicit method provide more accurate solutions for a class of differential equations known as stiff equations.

It is difficult to precisely define stiffness for differential equations as it depends on certain parameters such as the step size, the dimension of the system and the eigenvalues of the Jacobian $\frac{\delta f}{\delta x}$. In this section we show that for certain parameters, the dynamics of the grasper described in Chapter 4.3 demonstrates characteristics that requires care in choosing an appropriate solver. To this end, step responses of the grasper were obtained in simulations using three solvers in Simulink:

- A variable step solver (ODE45). This is Simulink's default variable step solver.
- An explicit fixed step solver (ODE4): This is the fourth order Explicit Runga Kutta solver (ERK4).
- An implicit fixed step solver (ODE14x).

Fig. A.1 shows the step response of the manipulator in simulation with the above three solvers. The four states of the system are shown over a period of 2 seconds. The fixed step simulations are performed at the sampling rate of 1 ms, which is the sampling rate of the implementation. The variable steps solver (ODE45) employs a strategy to guarantee a desired error tolerance. However, due to its variable step nature it is not appropriate for real-time implementation and is used as a reference for comparison. The implicit fixed step solver generates results that are close to the variable step solver. However, the plots for the explicit fixed step solver deviate from those of the other two solvers. This suggests that although ERK4 is often regarded as an accurate solver and the system has a relatively fast sampling rate, for our system the method generates erroneous results and must be avoided.

It must be noted that the Coulomb friction present in the system plays an important role in the accuracy of the solver. For small coefficients of Coulomb friction the solution of the explicit solver becomes closer to the other two solvers.

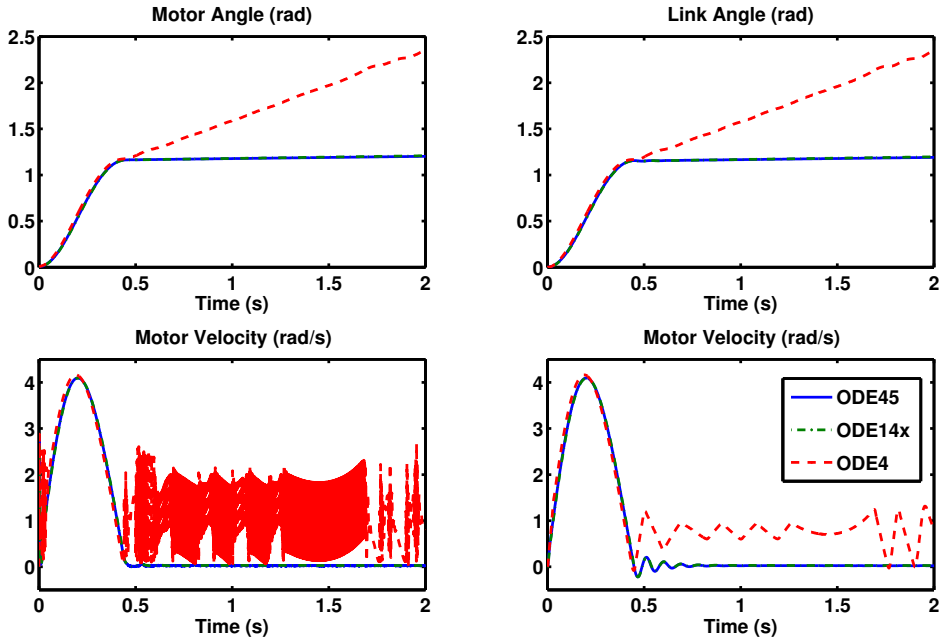

Figure A.1: States of the manipulator in simulation with three solvers in response to a step torque.

The integration in A.3 is carried out in several steps in the UKF and DDP implementation. The UKF uses a model of the system to propagate the sigma points through the dynamics. DDP uses the model in two places. The backward pass uses the dynamics to construct a quadratic (linear for iLQG) model of the system and the forward pass simulates the system forward in time by propagating through the dynamics. The above implementations all require discrete dynamics obtained by the integration in A.3.

The implicit solver in our real-time implementation uses 15 steps of forward Euler integration to derive a starting point for the solution. One iteration of Newton step is then performed with this starting point to obtain the solution. The Newton step requires the derivative of the dynamics which is obtained by Automatic Differentiation as described in Chapter 4.6. This implicit solver was used in the UKF and in the forward pass of DDP. However, due to real-time constrains, only a one step forward Euler integration was implemented in the backward pass of DDP. This did not impose a great problem in our implementation due to the more accurate model employed in the forward pass. Our future work will exploit the multi-core capability of the processor to achieve a real-time implementation of the backward pass with the implicit solver.

VITA

Sina Nia Kosari earned his B.Sc. degree from Amirkabir University of Technology in 2007 and his M.Sc. degree from McMaster University in 2009. Sina joined the BioRobotics Laboratory at University of Washington in 2009. He has been awarded the Natural Sciences and Engineering Research Council of Canada (NSERC) Post Graduate Scholarship in 2008 and 2010. Sina is an IEEE student member and a reviewer for several conferences and journals including Journal of Advanced Robotics, International Conference on Robotics and Automation, International Conference on Intelligent Robots and Systems and ASME International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference (CIE).