DIFFERENTIAL SURFACE KINEMATICS FOR TIME-VARYING, SURFACE-SURFACE FLOATING CONTACT INTERACTIONS

by

Donald Nelson

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

School of Computing
The University of Utah
May 2003
THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Donald Nelson

This dissertation has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Elaine Cohen

Christopher R. Johnson

Sanford S. Meek

Richard F. Riesenfeld

William B. Thompson
To the Graduate Council of the University of Utah:

I have read the dissertation of Donald Nelson in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Elaine Cohen
Chair, Supervisory Committee

Approved for the Major Department

Robert R. Kessler
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School
ABSTRACT

Algorithms to compute surface gradients are a key part of many constraint satisfaction and constrained optimization techniques. Methods to solve nonlinear problems use the surface Jacobian to help determine the next value in iterative schemes. In the past, graphics techniques that were intended for interactive rate solutions relied on numerical approximations to evaluate the Jacobian. To overcome the inaccuracies and instabilities that result from these numerical methods, we present a fast, closed form solution for obtaining the surface Jacobian. We have created a constraint framework in which to test contact, self-assembly, and surface impact problems using the analytical formulation presented here. In this work we develop the analytical solution in the context of the floating contact problem. By generalizing and reformulating a previous result in differential surface kinematics, we show that this technique also yields a parametric coordinate update for minimum distance for haptics and virtual reality applications. The utility of this formulation for surface impact dynamics and optimal surface configuration is also shown.
CONTENTS

ABSTRACT ................................................................. iv
LIST OF FIGURES ............................................................ ix
NOMENCLATURE .............................................................. xii
ACKNOWLEDGMENTS .......................................................... xiii

CHAPTERS
1. INTRODUCTION .......................................................... 1
   1.1 Overview ............................................................ 1
   1.2 Design Complexity .................................................. 2
   1.3 Computer-Aided Design (CAD) ...................................... 2
   1.4 Research Problem ................................................... 3
   1.5 Thesis ............................................................... 3
   1.6 Objectives ........................................................... 4
   1.7 Methodology ......................................................... 5
   1.8 Alpha Design Environment .......................................... 6
   1.9 Evaluation and Effectiveness ....................................... 6
   1.10 Document Summary ................................................ 6

2. BACKGROUND .......................................................... 8
   2.1 Analysis with Jacobians and Optimization Algorithms .......... 8
       2.1.1 Unconstrained Optimization .................................. 8
           2.1.1.1 Newton Iteration ...................................... 9
           2.1.1.2 Steepest Descent ...................................... 9
           2.1.1.3 Conjugate Gradient .................................... 10
           2.1.1.4 Davidon-Fletcher-Powell, BFGS ....................... 11
           2.1.1.5 Quasi-Newton Iteration ............................... 11
       2.1.2 Constrained Optimization ..................................... 11
           2.1.2.1 Penalty Methods ....................................... 12
           2.1.2.2 Method of Lagrange Multipliers ....................... 12
           2.1.2.3 Augmented Lagrangian Methods ....................... 13
       2.1.3 Global Optimization .......................................... 13
           2.1.3.1 Simulated Annealing .................................. 13
           2.1.3.2 Interval Analysis ..................................... 13
   2.2 Surface Representation ............................................ 14
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3</td>
<td>Mechanical Representation</td>
<td>15</td>
</tr>
<tr>
<td>2.4</td>
<td>Surface Kinematics</td>
<td>16</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Orthogonal Surface Parameterization</td>
<td>17</td>
</tr>
<tr>
<td>2.5</td>
<td>Automated Surface Placement</td>
<td>18</td>
</tr>
<tr>
<td>2.6</td>
<td>Haptics Tracing Algorithms</td>
<td>19</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Point Cloud</td>
<td>19</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Intermediate Tangent Planes</td>
<td>19</td>
</tr>
<tr>
<td>2.6.3</td>
<td>Ray-Surface Tracing</td>
<td>19</td>
</tr>
<tr>
<td>2.6.4</td>
<td>Point-Surface Tracing</td>
<td>20</td>
</tr>
<tr>
<td>2.6.5</td>
<td>Surface-Surface Tracing</td>
<td>20</td>
</tr>
<tr>
<td>2.7</td>
<td>Haptics Force Interaction</td>
<td>20</td>
</tr>
<tr>
<td>2.7.1</td>
<td>Surface Jacobians for Virtual Surface Contact Feedback</td>
<td>21</td>
</tr>
<tr>
<td>2.7.2</td>
<td>Inverse Dynamics Feedback</td>
<td>21</td>
</tr>
<tr>
<td>2.7.3</td>
<td>Three and Six Axes of Force Feedback</td>
<td>22</td>
</tr>
<tr>
<td>2.8</td>
<td>Nonholonomic Constraints</td>
<td>22</td>
</tr>
<tr>
<td>2.9</td>
<td>Geometric Satisfaction, Self-Assembly, Numerical Inverse Kinematics</td>
<td></td>
</tr>
<tr>
<td>2.10</td>
<td>Theoretical Kinematics</td>
<td>24</td>
</tr>
<tr>
<td>2.10.1</td>
<td>Calibration and Robotics Algebra</td>
<td>25</td>
</tr>
<tr>
<td>2.10.2</td>
<td>Body Velocity and Force Transformations</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Surface Contact Velocity Formulation</td>
<td>27</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Arbitrary Regular Parameterizations</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Surface Constraint Jacobian Extraction</td>
<td>31</td>
</tr>
<tr>
<td>3.3</td>
<td>Higher Order, Higher Dimension Tensors</td>
<td>34</td>
</tr>
<tr>
<td>3.4</td>
<td>Nonorthogonal Surface-Curve Velocity Formulation</td>
<td>34</td>
</tr>
<tr>
<td>4.1</td>
<td>Distance Extrema</td>
<td>37</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Comparing the Methods</td>
<td>38</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Local Collision Updates</td>
<td>39</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Surface-Surface Tracing Results and Analysis</td>
<td>39</td>
</tr>
<tr>
<td>5.1</td>
<td>Optimal Surface Placement</td>
<td>41</td>
</tr>
<tr>
<td>6.1</td>
<td>Initialization: Three-Step Process</td>
<td>48</td>
</tr>
<tr>
<td>6.2</td>
<td>Multisurface Models</td>
<td>50</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Surface-Curve Transitioning</td>
<td>50</td>
</tr>
<tr>
<td>6.3</td>
<td>Multiple Tracked Contacts</td>
<td>52</td>
</tr>
<tr>
<td>6.4</td>
<td>Algorithm Details: Implementation</td>
<td>54</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>INTERACTIVE MECHANICAL DESIGN VARIATION</td>
<td></td>
</tr>
</tbody>
</table>
| 7.1     | Interactive Design Variation ........................
| 7.2     | Interactive Numerical Optimization ...............|
| 7.3     | Generalized Inverse Kinematics ....................|
| 7.3.1   | Multifinger, Multihand Manipulation ..............|
| 7.4     | Alpha CAD Specification Virtual Mechanisms .....|
| 7.5     | Mechanical Design Objectives .....................|
| 7.6     | Surface Constraints .................................|
| 8.      | ANIMATING NONHOLONOMIC ROLLING CONTACTS ...........|
| 8.1     | Updating Parametric Surface Configurations from Rolling Contact Motion |
| 9.      | MULTIVARIATE SURFACE DERIVATIVES IN DYNAMICS ...|
| 9.1     | Dynamical Simulation ................................|
| 9.2     | Collisions and Momentum ..........................|
| 9.3     | Jacobian Computations ............................|
| 9.3.1   | Surface Constraint Jacobian Extraction ..........|
| 9.4     | Surface-Curve Constraint Jacobian Extraction ...|
| 9.5     | Partitioning: DAE → ODE ..........................|
| 9.6     | Surface Self-Assembly ..............................|
| 9.7     | Local Collision Detection and Extremal Distance Updates .............................|
| 9.7.1   | Backtracking in Time ................................|
| 9.8     | Inverse Dynamics: Applying the Surface Jacobian to Rendering Virtual Contact Force in Haptics ..................|
| 9.8.1   | Forward and Inverse Dynamics .....................|
| 9.8.2   | Computing the Surface Impact Response Force ....|
| 9.8.3   | Surface Impact Feedback Results ..................|
| 10.     | RESEARCH DISCUSSION AND RESULTS .................|
| 10.1    | Timing Complexity of Surface Kinematics ..........|
| 10.2    | Timing Complexity of the Surface Interaction Applications |
LIST OF FIGURES

2.1 Optimization landscape for objectives $f$ and constraints $C$. Isovalues for a two argument, scalar-valued function $f$ and $C$ are shown. 

3.1 Closest point surface contact frames. Velocity relations allow the contact coordinate velocities to be found.

4.1 Virtual proxies are important in the penetrating case. The maximal distance is required by the haptics tracing algorithm. Global solution discontinuities such as “chopping through” an object are not desirable. Because the velocity formulation (shown) is the “most local,” it is the method of choice for the penetrating case.

4.2 Curves of distance extrema embedded into the finger surface model and the CAD model.

4.3 The velocity formulation maintains the most local, or continuous, solution through a global change in penetration depth so that discontinuities in force feedback is avoided.

5.1 Optimal floating surface placement in a design variation framework. An additional condition is included to avoid placement in trimmed away areas, such as the boundary of the region on the right side of the left figure. The parametric constraint Jacobian for trimming constraints is obtained in closed form through the surface-curve velocity relations (Eqn. 3.33). The global method coordinates with the local, analytic method to add and remove global collision constraints.

5.2 Unsatisfied surface-surface contact constraint showing minimal distance between the surfaces and surface-curve minimal distance. Several ways to write the surface-curve inequality constraint are available.

5.3 OpenGL™ display and PHANToM™ force feedback interaction with a 3 paper clip surface constraint assembly.

5.4 PHANToM™ haptics interaction system with a virtual mechanism containing multiple surface constraints.

5.5 Kinematic manipulation of a spatial four-bar mechanism consisting of a universal joint, a spherical joint, and two revolute joints. The objective is to minimize the amount of separation of the top link with the hand grasp surface-surface floating contact.
6.1 The global polygonal minimal distance Proximity Query Package (PQP) (publicly available at http://www.cs.unc.edu/~geom/SSV) and the RAPID package [199] are used in the “far” regions or large surface - surface distances. This result, along with some parametric bookkeeping, starts a nonlinear root finger for tracking “near” cases. Each root is used and updated by the velocity formulation during the local tracking case. .......................................................... 47

6.2 Transitioning between the surface-surface distance tracking to the surface-curve case is shown. The surface-surface point of the trimmed away area is maintained so that the contact point can be used again if it falls onto a valid region again. ................................. 49

6.3 Surface-surface transitioning can occur in the concave case. The surfaces lose contact in the convex case, making the contact inactive. . 51

6.4 Close-up of a transition from a surface-surface constraint to a surface-curve constraint. Blue spheres are surface-surface constraints, while white are surface-curve constraints. ........................................ 52

6.5 Flow chart of the execution and data for the global management algorithm. ................................................................. 54

6.6 Manipulation sequence with surface-surface transitioning. Close-up view. Blue spheres are active constraints, while white are inactive, but still tracked. ......................................................... 56

7.1 Manipulation of the geometric parameters of a 6 DOF Stewart platform. The best platform position and orientation so that assembly constraints and additional constraints, like mechanical joint limit stops, are met fits naturally within this framework. Right: Joints are edges and bodies are nodes for the graph of a mechanism. This Stewart platform has 3 revolute, 3 spherical, 3 universal joints, and one constraint on a ground body holding it immobile. .......... 59

7.2 Quarter torus local body geometry. Two revolute joints are attachment features of this part. The assembled mechanism, or satisfaction of the geometric constraints is also shown. The ends of the linkage are attached, forming one closed loop. .................................. 62

8.1 Tick marks to show nonslipping between a convex, bumpy spheroid and a rounded surface. The distance between corresponding ticks on each surface is the same up to the order of accuracy of the integration method. .............................................................. 65

8.2 Angular velocity of the second surface is the cross product of the local contact velocity and the distance vector from the second surface body frame to the contact point. ................................. 67
9.1 The teapots approach and penetrate in multiple places, followed by backtrack to first contact. Local collision updates with the velocity formulation return maximal penetrating distance after impact occurs for algebraic collision analysis for smooth surfaces and better back-tracking. An alternative is to use the minimal distance to predict time of impact, but penetration can still occur no matter how small the timestep. ............................................................. 76

9.2 Surface constraint and joint constraint Jacobians and Hessians allow impacts between surfaces to be analyzed while respecting the algebraic joint constraints. ............................................................. 78

9.3 Impact sequence between a virtual finger (the Utah Teapot) and the virtual assembly. ............................................................. 81

9.4 Rendered, separate frame sequence. ............................................................. 81

10.1 An initial angular velocity is given for the Utah teapot as it drops onto the Lego assembly. ............................................................. 85

A.1 A universal joint is a spherical joint that can rotate in any way as long as both axes are perpendicular. .............................. 99
NOMENCLATURE

\( q \)  
generalized coordinates for articulated bodies

\( q_{tr} \)  
translation part of the generalized coordinates

\( q_{rot} \)  
rotational part of the generalized coordinates

\( C \)  
vector of all mechanical constraints, including joints and surface contacts

\( C_q \)  
partial derivative (Jacobian) of the constraints with respect to \( q \)

\( v \)  
relative linear velocity

\( \omega \)  
relative angular velocity

\( S \)  
surface evaluation point, in the local frame in which the body is modeled

\( S_{\text{rotated}} \)  
rotated, translated instance of \( S \)

\( f^i, g^i \)  
two surface instances in minimal distance computation pair \( i \)

\( u^i \)  
parametric extremal distance coordinates, for pair \( i \), containing 4 elements, \( u_f \) and \( u_g \) coordinate pairs on \( f \) and \( g \)

\( W \)  
generalized force, i.e., wrench

\( G \)  
matrix relating generalized velocity to linear/angular velocity

\( R_m \)  
\( 3 \times 3 \) rotation matrix, with a subscript \( m \) to indicate a rotation with respect to a frame or axis \( m \)

\( I \)  
first fundamental form of a surface

\( II \)  
second fundamental form of a surface

\( \hat{a} \)  
skew symmetric matrix performing the cross product operation \( \mathbf{a} \times \), i.e.,

\[
\begin{bmatrix}
0 & -a_z & a_y \\
-\hat{a}_z & 0 & -\hat{a}_x \\
\hat{a}_y & \hat{a}_x & 0
\end{bmatrix}
\]

\( \dot{u}, \dot{q} \)  
time derivative of \( u, q \)

\( R^T \)  
transpose of matrix \( R \)

\( M \)  
generalized mass matrix

\( \lambda \)  
undetermined multipliers

\( Q_e \)  
generalized external forces

\( Q_v \)  
generalized forces quadratic in velocity
ACKNOWLEDGMENTS

The author would like to thank Elaine Cohen for her years of patient advising and the valuable input from my committee members Chris Johnson, Sandy Meek, Rich Riesenfeld, and William Thompson. Thanks to Dave Johnson for his frequent insight and recommendations. Thanks to Richard Coffey for his video editing expertise. Thanks to Ali Nahvi for years of collaboration in various aspects of mechanical engineering applicable to haptics and calibration of huge robots. Thanks to Rod Freier for help in countless areas of wiring, network hacking and the like. Thanks to Tom Thompson for hacking on tracing algorithms and movable objects. Thanks also go to the students and staff of the Alpha_1 project, within which this work was developed. Support for this research was provided by DARPA grant F33615-96-C-5621, and by the NSF and DARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219).
CHAPTER 1

INTRODUCTION

1.1 Overview

Numerical techniques are used to analyze problems whose optimal solutions cannot be found efficiently in closed form. Many such problems are found in CAD modeling, virtual reality, and kinematic sensitivity analysis. Optimization algorithms that require derivative information have improved performance if the derivatives are computed robustly and quickly. This dissertation formulates special multivariate surface derivatives, or surface Jacobian, and higher order derivatives, Hessians and tensors, for improved performance and accuracy in numerical methods that evaluate parametric surface-surface contact interaction.

The set of geometric constraints of a mechanical system defines a hypervolume, the constraint manifold, of the valid configuration space for use in physically-based modeling. An optimal design of a virtual mechanical system must satisfy this manifold of constraints. We will develop specialized constrained optimization and derivative techniques for surface-surface interactions including optimal surface placement, nonholonomic rolling contact, and other kinematic effects.

The constraint manifold that characterizes surface interactions requires computation of extremal distance between two surfaces in the context of floating contact constraints. Extremal distance is the minimum distance or maximum penetrating distance between models, and floating contact refers to the change of the extremal parametric coordinates with respect to model configuration. The research presented here advances this contact relation and provides a compact, efficient tool that can serve as a basis for further research into surface kinematics. In the next few sections, we present a general surface velocity formulation and surface derivatives extraction. This information improves optimal surface placement, determines algebraic surface-
surface collisions, improves previous work [191] on Lagrange multiplier curve-curve impact dynamics, and generates kinematic or “prescribed” rolling motion.

1.2 Design Complexity

The complexity inherent in surface design and virtual mechanical design can be reduced when details are disassociated from the high level design. Being able to use abstract commands frees a designer from having to specify an overwhelming number of details. Automatic constraint specification operators augment a modeling vocabulary. For example, have the system automatically adjust surface position and orientation to avoid collisions, while simultaneously requiring two particular surfaces to touch, anywhere, but not specifying the contact locations. A system with such capabilities can free a human from cumbersome details. Without constraint satisfaction techniques, the human designer would have to determine an admissible configuration for the mechanical parts, and from the admissible set, determine the best choice.

The representation of mechanical systems in this work also simplifies the design process in that graph theory is used to specify mechanisms. Joints are graph edges and bodies are nodes in this abstraction. Mechanical part features such as joints are defined through local body frame geometry and body state world frame generalized coordinates. This description allows the joints or surface constraints to be defined as local constraint edges or connections, which can be incorporated into the optimization algorithm.

1.3 Computer-Aided Design (CAD)

A designer using the numerical techniques in this work can visualize and feel the CAD mechanisms through graphics and haptics feedback. Kinematic manipulation in the optimal surface placement application allows the mechanism reach, or kinematic workspace, to be visualized. Reaching out and touching a dynamical, articulated body system through the impact dynamics application allows haptic and graphical visualization of the effects of smooth surface collisions on a system. In addition, an algorithm for evaluating kinematic rolling is implemented in
MATLAB\textsuperscript{TM} for use in determining kinematically admissible rolling paths between surfaces.

1.4 Research Problem

Virtual prototyping applications for CAD add a sense of contact and manipulation to the process of interactively designing physical models and mechanical systems. The problem of extending the manipulation to systems with surface constraints and providing surface-surface contact interactions in the augmented reality framework can be cast as constraint satisfaction, including joint, surface, and finger contact constraints. The optimization algorithm must be able to update small changes (from manipulation) in the system at force control rates. The evaluation of the derivatives of the constraints and objectives on the system quickly for the optimization algorithm is a computational challenge.

To overcome the inaccuracies and instabilities that result from numerical approximations, we present a fast, closed form solution for obtaining the parametric floating contact surface Jacobian and higher order derivative tensors. Physically based modeling of certain surface-surface contact interactions, including interactive virtual manipulation of trimmed surface assemblies, nonholonomic contact conditions, and interrogation of models through surface-surface tracing, are investigated using closed form surface Jacobian techniques in this document.

The problem of extending this analysis to models, shells of trimmed NURBS surfaces, rather than a single surface, is investigated also. In support of this approach, management of global surface effects during manipulation takes advantage of advances in [205, 175].

1.5 Thesis

Surface contact constraint interactions can be evaluated, in sufficient generality to be useful (i.e., for nontrivial models), through differential surface kinematics relations at interactive rates.

The closed-form, symbolic modeling of surface-surface interactions through constraints and constraint Jacobians that define a kinematically admissible configura-
tion space and admissible motion are optimized, explored, and visualized with the design variation framework presented in this document. Two hurdles, resolved in Chapter 3, that we overcome to allow our applications to work in a general sense, are:

- Formulating differential surface kinematics relations for arbitrary surface parameterizations,
- Deriving the analytic “extremal distance parametric floating contact Jacobian” so that the surface interactions fit into the Alpha_1 [197] Design Variation and impact dynamics frameworks.

1.6 Objectives

The goals of this research are to achieve interactive force-control (multi-kilohertz) rates for surface contact interactions. The aim is then to develop an efficient arbitrary surface kinematics formulation and fast surface Jacobian formulation. In particular, the objective is to achieve:

- Numerical stability and efficiency from computing with this closed form representation over the alternative approach of numerical partial differentiation. We formulate a solution to compute all of the derivative matrix columns with two surface evaluations with the analytic solution and obtain a $10^3 - 10^4$ factor speedup over expensive high order finite differencing.
- Higher order/higher dimension closed form Jacobians and all surface second, third, .. nth partial derivatives w.r.t. floating contact coordinates. Higher order derivative tensors are needed in special optimization and dynamical simulation problems. Numerical methods cannot reliably evaluate second or higher order Jacobians.
- Optimal surface placement and surface placement within a parametric range (such as outside trimmed away areas). The fast Jacobian formulation allow us to write and implement surface constraint and surface design objectives in a mathematical programming framework.
• Unilateral nonholonomic, rolling contact surface constraints. We demonstrate an application of these methods for generating nonholonomic rolling contact motion, where one surface follows a curve embedded in another surface so that there is no slip between the surfaces.

• Local surface-surface collision updates for haptics. Being able to reach and touch a virtual surface through a force feedback device with a surface-surface contact model is the next frontier in the contact problem in haptics. We develop the surface kinematics relations as the next step in obtaining fast finger or hand penetration computations.

• Extend previous work on planar curve impact dynamics to the 6 DOF spatial case for surface impacts. Previous dynamics formulations require the Jacobian information from the surface kinematics relations.

1.7 Methodology

The approach taken to this point has been to debug the mathematics in a very high-level interpreted environment, Maple\textsuperscript{TM} and Matlab\textsuperscript{TM}. Once this is done, a standard set of header files and procedures has been created to quickly convert the interpreted solution into high performance C++ code (several orders of magnitude faster than interpreted Matlab\textsuperscript{TM}) within the Alpha_1 \cite{197} environment. The resulting code is often quite small; the optimal surface and design variation framework amounts to just over 20,000 lines of C++.

The constraint and constraint Jacobians have been developed for the framework outlined in Chapter 7. It is a hybrid symbolic and numerical solution since the constraint Jacobian objects are formulated symbolically at a high level, computed in closed form, and are then used in a more general numerical algorithm.

Our optimization algorithm utilizes intersample coherence in order to maintain the optimal solution over time. The initialization of the constraint systems is a different, more costly problem, deferred to the global management algorithm in Chapter 6.
1.8 Alpha_1 Design Environment

The Alpha_1 [197] environment is leveraged to provide a backbone of geometric surface evaluation and design tools. The implementation of the surface tracing, optimal surface configuration, and surface impact dynamics algorithms is invoked through the shape_edit modeling application. Geometry defined in shape_edit provides models (multisurface shells) for input into the tracing, optimization, and dynamics processes for use in the surface and curve constraint Jacobian evaluation.

1.9 Evaluation and Effectiveness

The surface kinematics problems in the next sections generally require the solution to systems of nonlinear equations. We want to measure the effectiveness of our solution to these systems in terms of interactivity and robustness. We use characteristics from the problem domain to reduce the problems caused by the nonlinearity. The system is linearized, without error, in the case of surface-surface tracing for the local case by looking at the time derivative of the distance between surfaces. The interframe coherence in the optimal surface placement solution allows the optimal solution to be maintained in the steady state, during manipulation. Global management is also required in the surface placement solution for initialization and for changing the set of constraints during changes in surface contact state. We use computation time and the size of the region of convergence to measure the level of interactivity and robustness.

The numerical conditioning of the velocity formulation to follow in Chapter 3 is measured. The issues caused by special cases where the condition number is known to be infinite are addressed in Chapter 10.

The level of immersion and realism achievable in our system is also explored. The scalability of increased size of the virtual environment and the availability of more processors are examined during this work with results given in Chapter 10.

1.10 Document Summary

The previous work on surface kinematics, haptics, sensitivity analysis and some general background is outlined in the next section. Then some differential relations
are developed so that the fast extremal distance parametric floating surface contact constraint Jacobian can be extracted for use in various surface interactions. This technique is then be applied in a surface-surface tracing framework. The fast Jacobian formulation is embedded in a design variation framework for virtual mechanical systems with surface and curve constraints. We note that the differential surface contact relations allow us to generate nonholonomic rolling contact motions and improve previous works in unilateral, nonholonomic surface rolling contact constraints. Finally, we illustrate the use of higher order surface contact derivatives in previous work on impact and inverse dynamics algorithms and conclude the work.
CHAPTER 2

BACKGROUND

2.1 Analysis with Jacobians and Optimization Algorithms

The problem of finding the Jacobians of the objectives and constraints is central to many optimization, sensitivity analysis, and other algorithms. Numerically derived derivatives are often computationally expensive and can deviate from the true solution or become unstable. For example, in Serban and Haug [148], it is shown that certain closed form derivatives are orders of magnitude faster to evaluate than the numerically obtained derivatives. For our application, having a symbolic, rather than numerical form, for the surface constraint Jacobian is important in achieving high update rates for interactive CAD and prototyping applications.

In principle, quantities that can be encoded as mathematical objectives can be cast within an optimization framework. The numerical framework for constrained optimization is useful for expressing a wide variety of problems. A review of the optimization algorithms for constrained and unconstrained minimization is presented here. These numerical methods are central to the body geometry and assembly optimization research that follows in later chapters.

The general setting of the optimization problem is to find a minimum value of a function $f$ with arguments $x$. For the constrained optimization problem, the minimum is found so that additional conditions, constraints $C(x)$, are met.

2.1.1 Unconstrained Optimization

The applications in this work require constrained optimization. The constrained methods typically recast the objective function so that the unconstrained algorithms may solve the new objective.
2.1.1.1 Newton Iteration

Newton’s method (called Newton-Raphson iteration for the specialized problem of locating a root of a real-valued function of a real variable) for finding the minimum of a function $f$ with a vector of arguments $x$ is cast as finding the roots of the first derivative of $f$. This method is derived in many introductory texts using geometric and Taylor’s series arguments [180]. The general equation for finding the minimum of $f$, or the zeros of a system of equations $g$, where $g = df/dx$, is to iterate over the sequence

$$x^{k+1} = x^k - J^{-1}(x^k)g(x^k)$$

(2.1)

where $J$ is the Jacobian matrix of $g$ with respect to $x$. Newton’s method exhibits quadratic convergence properties given suitable conditions [180]. This method is used to find an optimal assembly configuration of the minimal distance between two surfaces, with appropriate starting values around some locally convex region for convergence [116, 174]. The surface-surface starting points allow the closed-form velocity formulation presented in Chapters 3, 4 and 5 to work.

In general, global convergence analysis for Newton’s method is a difficult problem. However, when the initial $x$ is within a convex region, Newton’s method is known to converge. Further, this can be checked since when the Jacobian $J(x)$ is positive definite, then $x$ is in a convex region.

2.1.1.2 Steepest Descent

To use Newton’s method requires evaluation of the second derivatives when minimizing a function $f$ because the Jacobian of $g$ is required. This evaluation can be expensive, especially for a large vector $x$. For this reason, simpler methods that descend into a solution by “skiing downhill,” or moving in the negative of the slope of $f$, are sometimes used. That is,

$$x^{k+1} = x^k - k\partial f/\partial x^k$$

(2.2)
incrementally improves the minimal solution for some weight \( k \). For a system of equations \( f \), the scalar objective \( l(x) = f^T f \) corresponding to “optimization energy” can be minimized with steepest descent [95]. The energy scheme is closely related to the penalty methods for constrained optimization, presented in the next section. The steepest descent method does not converge as quickly as Newton’s algorithm since steepest descent exhibits at best linear convergence [24].

2.1.1.3 Conjugate Gradient

While using steepest descent methods avoids having to compute expensive second derivative information, a large number of iterations may be required for highly non-linear objectives. The Conjugate Gradient method improves the convergence rate by, at the \( k^{th} \) iteration, using a “search direction” \( p^k \) (where \( x^{k+1} = x^k + \alpha p^k \)) that is orthogonal to previous search directions \( p^1, p^2, ..., p^{k-1} \). Suppose the objective \( f \) is approximated by a quadratic function

\[
f(x) = \frac{1}{2} x^T H x - b^T x. \tag{2.3}\]

As an intermediate step, it is necessary to find the optimal \( \alpha \) for minimizing \( f(x^{k+1}) \) for \( x^{k+1} = x^k + \alpha p^k \).

The conjugate direction is given by [232]

\[
p^{k+1} = -g^{k+1} + \sum_k \beta^k p^k, \tag{2.4}\]

for various solutions for \( \beta^i \), for example [232],

\[
\beta^k = (g^{k+1} - g^k)^T g^{k+1} / (((g^{k+1} - g^k)^T p^k). \tag{2.5}\]

The conjugate gradient method is also used to find the inverse of a sparse matrix more quickly than the standard cubic cost methods by using a similar quadratic function approximation [180, 8]. These methods are most helpful for matrices with known sparse structure [180].
2.1.1.4 Davidon-Fletcher-Powell, BFGS

Other widely used schemes attempt to approximate the second derivative for improved convergence rate over the Steepest Descent and Conjugate Gradient methods. The Hessian matrix $H$ can be approximated by a first difference quotient, where $g^{k+1} = g(x^{k+1})$,

$$H(x) = (g^{k+1} - g^k)^T(x^{k+1} - x^k). \quad (2.6)$$

The inverse of the Hessian, $B$, is computed according to the Davidon-Fletcher-Powell formula [232],

$$B^{k+1} = B^k + \frac{p^{k+1} \times p^k}{p^{k+1} \cdot p^k} - \frac{Bq^{k+1} \times Bq^k}{q^{k+1} \cdot Bq^k} \quad (2.7)$$

and a line search along $p = -Bg$ as in the CG method is used. The Broyden-Fletcher-Goldfarb-Shanno formula [232] for $B$ can also be used for more advanced implementations.

2.1.1.5 Quasi-Newton Iteration

Suppose the Jacobian matrix of Eq. 2.1 is not square. This can happen when a system of equations $g(x)$ is overdetermined or underdetermined. For such systems, a generalized inverse, or pseudo-inverse, replaces the matrix inverse in Eq. 2.1. For the overdetermined case, one possibility is

$$\Delta x^k = -(J^TJ)^{-1}J^Tg \quad (2.8)$$

A full discussion on pseudo-inverse forms is deferred to [9].

2.1.2 Constrained Optimization

The applications of optimal surface placement and assembly design variation in the following chapters impose constraints $C$ on the mechanical and graphics design objectives $f$. 
2.1.2.1 Penalty Methods

The constrained optimization problem can be cast as combining the objective $f$ with a positive “constraint energy” term $C^T C$ [95, 187] to form a new optimization objective

$$l_{pen}(x) = f + \frac{k}{2} C^T C.$$  \hfill (2.9)

The unconstrained optimization methods can be leveraged to find the best value for $l$.

2.1.2.2 Method of Lagrange Multipliers

Lagrange methods for constrained optimization use the fact that at the optimal point $x^*$ in Fig. 2.1, the vectors $\nabla f$ and $\nabla C$ are facing away from each other. The equality can be written in terms of some unknown multiplier $\lambda$ as

$$\nabla f + \lambda^T \nabla C = 0.$$  \hfill (2.10)

The objective $l(x)$ in view of this is

$$l(x) = f + \lambda^T C.$$  \hfill (2.11)

Figure 2.1. Optimization landscape for objectives $f$ and constraints $C$. Isovalues for a two argument, scalar-valued function $f$ and $C$ are shown.
2.1.2.3 Augmented Lagrangian Methods

The Augmented Lagrangian technique [190, 188, 187] combines the Penalty and the Lagrange multiplier methods,

\[
l_{\text{aug}}(x) = f + \lambda^T C + kC^T C.
\]  

(2.12)

When the constraints are satisfied (at \( C = 0 \)), the first order conditions for the Augmented Lagrangian, Penalty method and Lagrange methods are in agreement.

\[
\nabla l = \nabla l_{\text{aug}} = \nabla l_{\text{pen}} = \nabla f + \lambda^T \nabla C = 0.
\]  

(2.13)

During optimization iteration, both \( x \) and \( \lambda \) are updated by minimizing the objectives \( l \). The unconstrained methods can be used to achieve this minimization.

2.1.3 Global Optimization

The unconstrained methods (which are the backbone of the constrained methods) have the problem that they may only find locally optimal solutions or they may find only one of the globally optimal solutions. A variety of approaches have been used to combat this difficulty.

2.1.3.1 Simulated Annealing

Simulated annealing is the mathematical analog to the process of cooling metal in stages so that it converges to the proper shape. A statistical or point sampling analysis can be determined to provide an array of initial guesses for solving multiple instances of the local optimization problem. The local solutions are allowed to proceed without perturbation to the extent of the “temperature” or level of convergence of the system.

2.1.3.2 Interval Analysis

Interval arithmetic [6, 75] is a description of elementary arithmetic operations on numbers bounded to be within a certain range, say \([x, \bar{x}]\). Addition of two bounded numbers \([x, \bar{x}] + [y, \bar{y}]\) is (guaranteed to be) \([x+y, \bar{x}+\bar{y}]\) when + takes into account numerical rounding artifacts. This arithmetic can be used to subdivide the
optimization space to satisfy a variety of global optimization problems. The initial guess for an optimization algorithm can be $[-\infty, \infty]$ so that any and all solutions are guaranteed to be found. Round off error can be modeled with the bounds as well.

The method is absolutely robust but is very slow. The elementary operations are usually 20-30 times slower than the associated hardware operations. The subdivision approaches such as generalized bisection and binary search can be combined with interval Newton iteration to speed up the process. In that sense, it is different from the subdivision operations in graphics and Alpha_1 since intersection operations with generalized Newton methods is possible since the interval arithmetic has been defined. Interval arithmetic can solve a manifold of surface contact conditions or place guarantees on the convergence of the optimization algorithm. We note the utility of the parametric contact Jacobian for the development of the inclusion function of $u_q$, defined in Chapter 3. The interval analog to $u_q$ is easily computed since that Jacobian is evaluated using elementary operations such as matrix multiplication and $2 \times 2$ matrix inverse operations.

An efficient means for obtaining the interval surface constraint Jacobian is obtained using these operations. Otherwise, it would be obtained using subdivision operations. These operations are expensive for reasons analogous to those for finite differencing operations in our numerical surface constraint Jacobian.

The interval analog to Eqn. 2.1 is, for $f : x = [\underline{x}, \overline{x}] \rightarrow \mathbb{R}^n$,

$$\tilde{x}^{k+1} = \tilde{x}^k - J^{-1}(x^k)g(\tilde{x}^k),$$

where $\tilde{x}^k = \frac{1}{2}(\underline{x} + \overline{x})$. $x^{k+1}$ is incremented through $x^{k+1} = x^k \cap \tilde{x}^{k+1}$. If $\tilde{x}^{k+1} \subset x^k$ then there is a unique root in $x^k$. The method converges quadratically given certain initial conditions [76].

2.2 Surface Representation

The piecewise tensor product non-uniform rational B-spline definition of a surface $S$ is a mapping from $\mathbb{R}^2 \rightarrow \mathbb{R}^3$, i.e., a function from parametric $(u,v)$ space to
Cartesian \((x,y,z)\) space.

\[
\mathbb{S}(u,v,p) = \frac{\sum_{i,j} w_{i,j} B_{i,\kappa_u}(u) B_{j,\kappa_v}(v)}{\sum_{i,j} w_{i,j} B_{i,\kappa_u}(u) B_{j,\kappa_v}(v)} = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}
\]

where the B-spline blending functions \(B\), control mesh \(p\), knot vectors \(\kappa_u, \kappa_v\) and weights \(w\) are used. The over-line notation such as \(\mathbb{S}\) indicates the local body coordinate system (or coordinate frame).

The position and quaternion orientation coordinates for body \(k\) are denoted by \(q^k\). A surface \(S(q^k, u^k, v^k, p^k) = T(q^k)\mathbb{S}(u^k, v^k, p^k)\), where \(T(q^k)\) is the coordinate transform that transforms surface \(S^k\) from the local frame to the global frame.

Suppose two surfaces \(S^i\) and \(S^j\) are in contact, denoted \(f\) and \(g\). The parametric contact coordinates \(u = [u^i v^i u^j v^j]^T\) for \(f\) and \(g\) are completely dependent on \(q^{i,j} = [(q^i)^T (q^j)^T]^T\) and are therefore removed from the generalized coordinates in this dynamical simulation. However, the dynamics algorithm still tracks all pairs of potential contacts \(u\) through the velocity formulation (Appendix A [231]).

Because the contact coordinates on one surface are functions of the position and orientation of that surface and the surface it contacts, \(f\) is a function of \(q^i\) and \(q^j\). The surface floating contact Jacobian is \(f(q^{i,j})_{q^{i,j}} = \frac{\partial f}{\partial q^{i,j}}\). \(g_{q^{i,j}}\) is computed similarly. Because the Jacobians are dependent on both sets of generalized coordinates (denoted as simply \(q\) hereafter), there is some “cross talk” between the surfaces. \(f_q\) and \(g_q\) are of size \(3 \times 14\) for quaternion generalized coordinates, or \(3 \times 12\) for Euler angles or the exponential map. The Jacobians have a closed form solution which is derived in Chapter 3.

### 2.3 Mechanical Representation

Simple joints and surface contact can be mathematically encoded as constraints, as derived in Section A.3, which yields a simple universal joint constraint. The vector of constraints \(C(q,t)\), expressed in augmented generalized coordinates \(q\), (referring to all the bodies, not just those in surface contact) defines the mechanical kinematics [28, 61, 148, 158]. Coordinate approaches such as the Denavit-
Hartenburgh or Hayati representations cannot be used since they do not support computations on surface contacts [59, 177, 107, 26]).

The augmented generalized coordinate representation for describing the kinematics of mechanisms for a rigid or flexible body is used in this work. Each model’s coordinates are defined as a vector of world frame position, orientation, and other variables such as its deformation state. We prefer augmented coordinates due to the generality of expression of surface constraints that are not included in minimal coordinates. Joints in augmented coordinates are defined as a system of constraint equations, \( C \), that are a functions of coordinates and design variables, concatenated in \( q \). \( C_q \) refers to the partials of the constraint functions with respect to the variables in the vector \( q \).

Each constraint (joint) between bodies is represented by a constraint equation \( iC \). We demonstrate a specialized method to augment such systems with design objective functionals to solve kinematic systems in the manner of a generic constrained optimization problem:

\[
\text{minimize } f \text{ subject to } C = 0.
\]

Example objectives include maximizing a mechanical character’s dexterity or maximal collision avoidance. A fast solution (to follow) is used for mechanisms that are assembled and do not suffer local minima problems during manipulation, i.e., in the steady state. Accelerations derived from the assembly manipulation create inertial forces and other effects for use in haptics force feedback to the operator.

### 2.4 Surface Kinematics

Much research has been directed at solving the surface contact problem. Force computations for polygonal-surface contact have been addressed by several authors [204, 191, 17]. The kinematics of contact for smooth, parametric surface representations were first derived by [163, 4] for a special class of surface parameterizations. This derivation has been used by several other works, for dynamical impact analysis.
[11], rolling path planning [32] and robotics applications [86]. Chapter 3 provides a new, generalized derivation of the original, special-case, result [163, 4].

2.4.1 Orthogonal Surface Parameterization

Under the assumption that both surfaces \( \mathbf{f} \) and \( \mathbf{g} \) are parameterized so that each surface’s partials are everywhere orthogonal, \( \mathbf{f}_u \cdot \mathbf{f}_v = 0 \) and \( \mathbf{g}_u \cdot \mathbf{g}_v = 0 \), a result developed for robotics applications [163] relates the change in parametric contact coordinates \( \mathbf{u} = [\mathbf{u}_f^T, \mathbf{u}_g^T]^T \) with the velocity of the surfaces \( \mathbf{f}, \mathbf{g} \) in contact. Let the relative linear and angular surface velocities of surface \( \mathbf{g} \) relative to surface \( \mathbf{f} \) be denoted by \( \mathbf{v}, \omega \), and the surface contact velocities be \( \mathbf{v}^f, \omega^f \) and \( \mathbf{v}^g, \omega^g \). Letting \( \mathbf{I} \) be the first fundamental form and \( \mathbf{II} \) be the surface curvature or second fundamental form, with subscripts for surfaces \( \mathbf{f} \) and \( \mathbf{g} \). If \( \theta \) represents the angle between parametric axes \( \mathbf{g}_u \) and \( \mathbf{f}_u \), let \( \mathbf{R}_\theta = \begin{bmatrix} \cos \theta & -\sin \theta \\ -\sin \theta & -\cos \theta \end{bmatrix} \) and \( \mathbf{\tilde{II}} = \mathbf{R}_\theta \mathbf{II} \mathbf{R}_\theta \).

The time derivative of the contact coordinates \( \dot{\mathbf{u}} \) is

\[
\begin{align*}
\dot{\mathbf{u}}_f &= \mathbf{I}_f^{-1} \mathbf{R}_\theta (\mathbf{II}_g + \mathbf{\tilde{II}}_f)^{-1} \left( \begin{bmatrix} -w_y \\ w_x \end{bmatrix} + \mathbf{II}_g \begin{bmatrix} v_x \\ v_y \end{bmatrix} \right) \\
\dot{\mathbf{u}}_g &= \mathbf{I}_g^{-1} (\mathbf{II}_g + \mathbf{\tilde{II}}_f)^{-1} \left( \begin{bmatrix} -w_y \\ w_x \end{bmatrix} - \mathbf{\tilde{II}}_f \begin{bmatrix} v_x \\ v_y \end{bmatrix} \right).
\end{align*}
\]

(2.15)

This relation is generalized in Chapter 3 so that models from Alpha\textsubscript{1} can be used directly.

Another, more general work in the surface kinematics area is given by [204], which is not limited by orthogonal surface parameterization. However, the method is more expensive computationally than the techniques in Chapter 3. More importantly, the Jacobian determinant methods in [204] update the parametric contact coordinates through nonlinear terms in velocity. While the parametric contact Jacobian is not used in [204], the inability to extract the Jacobian due to the nonlinearity in velocity makes the method unsuitable for optimal surface placement and dynamics applications. For reference, the determinant method uses the implicit function theorem to find the time derivative of the extremal parametric contact coordinates from the determinant of the Jacobian of the minimal distance objective.
functions $E_1$, $E_2$. Several mathematically encoded objectives can be used. The contact coordinate velocities are then given by

$$\dot{u} = \frac{\left| \frac{\partial (E_1, E_2)}{\partial (t, u)} \right|}{\left| \frac{\partial (E_1, E_2)}{\partial u} \right|}$$

(2.16)

The determinant of the Jacobian method gives the parametric contact velocities, but is not written as an equation linear in $\dot{q}$. The parametric contact Jacobian cannot be extracted with any amount of revision since Eq. 2.16 is not a linear equation. For the purposes of Jacobian extraction (not the concern of [204]), this is the critical distinction. We formulate an equation linear in $\dot{q}$. It is also notable that the dynamics formulation from [204] is solving the resting contact problem, different from our impact dynamics problem in Chapter 9.

### 2.5 Automated Surface Placement

Previous research in interactive methods for surface placement uses springs and collision detection for pushing single objects at a time into nonpenetrating configurations [201, 16]. Using surface and parametric contact Jacobian presented here, it is possible to generate generalized optimal surface placement at interactive rates without resorting to approximations outside the mathematical programming framework. We implement a solution for updating numerous contacts simultaneously at force-control rates for real CAD models in the following sections, which improve previous .1Hz-20Hz single-surface update rates (scaled for current machine speeds).

Depth and model maps have been used for automated surface placement in registering real, camera-tracked objects and augmented reality objects [16]. These methods are approximations and separate from the smooth contact analysis that is the concern of this work.

Interval analysis has been used very effectively to place surfaces, but at very high cost, without problems of running into local minima [75, 76, 201, 16]. These methods make use of “interval Jacobians” and numerical methods for evaluating these Jacobians. The goals of our research do not focus on finding initial valid,
optimal configurations because of the existence of global optimization algorithms such as interval methods. We will focus on maintaining and updating the state of the optimization algorithms with a fast local and hybrid global management scheme.

2.6 Haptics Tracing Algorithms

To establish a sense of presence in augmented reality applications it is important to solve the virtual contact problem. While not the primary focus of this research, a relation that improves local surface-surface collision updates and surface-surface extremal distance tracing updates builds upon previous work. Solving the general surface tracing problem is an important precursor to establishing a realistic grasp for manipulation of virtual mechanical models discussed in Chapter 7. The geometry of haptic surface tracing provides the amount of virtual penetration into a surface for force feedback computations, which are usually linearly related to penetration depth, although nonlinear viscous damping models have also been used [234].

2.6.1 Point Cloud

The geometry of early approaches for finger-surface interaction was a scan from a finger point to the nearest of a set of cloud points. The normal at each cloud point was also stored. Force feedback can be evaluated depending on the penetration depth into the “polygon” representation of the cloud point and normal.

2.6.2 Intermediate Tangent Planes

Another early method is to compute the finger-model collisions with polygonal methods. The single closest polygon can be used to compute fast collision updates when the force controller is a very slow processor [235]. Force shading akin to Phong shading in computer graphics has been used to smooth the flat polygonal representation [236].

2.6.3 Ray-Surface Tracing

While not widely used for haptics, ray-surface algorithms can be used when a finger “line” model is used. This method was the first direct numerical algorithm
for smooth surface tracing [237].

2.6.4 Point-Surface Tracing

A fast numerical algorithm for the point-smooth surface tracing model using derivative information from the surface has been employed in several works [175, 176, 174]. Offset surfaces can be applied as a preprocess to the model to give the effect of tracing a surface with a solid sphere instead of a point. The sphere representation and inability to change the surface due to the cost of the offsetting operation are limitations of this quasi-surface-surface tracing method.

2.6.5 Surface-Surface Tracing

Previous work in haptics surface-surface tracing for virtual prototyping and surface design applications has used discrete voxels and polygon methods [199, 276]. We develop a tracing method for smooth surface-to-surface interactions. A straightforward extension of the point-surface formulation to surface-surface can yield extraneous, undesirable solutions. We derive an alternative novel velocity formulation for use in a surface-surface tracing paradigm that exhibits additional stability beyond the Newton methods. Both methods require evaluating the surface point and first and second surface partial derivatives for both surfaces, an efficient kilohertz rate computation. These methods are integrated into a three-step tracking process that uses a global minimum distance method, the local Newton formulation, and the new velocity formulation.

2.7 Haptics Force Interaction

Open and closed loop inverse dynamics computations are important considerations in determining the force to apply to realize a prescribed motion for control applications [159, 60]. They are also important for rendering the virtual inertial and constraint forces during interaction with augmented reality systems. The display of forces that are felt during manipulation of a virtual mechanical assembly using inverse dynamics in reduced coordinates was previously investigated [59]. This method is easily extended to the coordinates used in Chapter 7.
The surface-surface contact problem for virtual prototyping applications usually adopts a penalty, or stiffness, model for rendering force feedback to a user [25]. The use of a compliance, or admittance, model has been developed to display force during manipulation of simple virtual objects [222, 221]. This method is a unique departure from standard stiffness or impedance model where virtual position violation is the basis of force display. With compliance, force is sensed and position is displayed or “rendered.” For devices with force sensors, this is a possible alternative. Our methods apply to devices that do not have force sensors. We use previously developed techniques [176] under the stiffness model category for surface-surface tracing force feedback implementation and new developments that are departures from the stiffness method.

### 2.7.1 Surface Jacobians for Virtual Surface Contact Feedback

The multivariate surface derivatives derived in the next chapter can be used to provide force feedback for use in virtual prototyping applications. The Lagrange multipliers associated with the surface contact constraint constraint for the transient and maintained contact case can be multiplied by the surface Jacobians to provide force feedback. The resulting force is generalized force in the mechanisms’ configuration space that can be mapped back to force/torque space for feedback to a user.

### 2.7.2 Inverse Dynamics Feedback

Constraint forces and inverse dynamics forces are computed according to methods for virtual mechanical assemblies previously developed by the author [247]. Force in configuration space of the mechanism \( \mathbf{W}_c \), such as in the \{Cartesian, quaternion\} set of coordinates, may be expressed in terms of body space force and torque \( \mathbf{W} \) through standard transformation operators, denoted by \( \mathbf{G} \) [158, 28],

\[
\mathbf{W} = \frac{1}{4} \mathbf{G}(q) \mathbf{W}_c
\]  

(2.17)

\( \mathbf{G} \) is given in Eqn. 2.25. This force and torque may be projected onto the joint axes of a haptics device to provide the amount of torque required by a controls system.
to produce a motion. For our interactive haptics force feedback application, $W$ can be transmitted to the user.

### 2.7.3 Three and Six Axes of Force Feedback

Our system drives a PHANToM TM haptics device that senses 6 DOFs of position and renders three DOFs of force. The generalized wrench to be rendered to the user, $W$, is input into the algorithm. The first three components of $W$ are taken as output for the three DOF PHANToM case.

For a more complicated anthropomorphic haptic mechanism providing feedback to a user’s entire arm, we use a relation from statics to render the virtual wrench to the appropriate actuators [176]. The following equations allow the use of a tool frame for the finger without changing the manipulator Jacobian.

$$W = \begin{bmatrix} f_1 \\ r_1 \times f_1 \end{bmatrix},$$

$$J = \begin{bmatrix} z_0 \times b_0 & \ldots & z_4 \times b_4 \\ z_0 & \ldots & z_4 \\ 0 & 0 & 0 & z_5 & z_6 & z_7 \end{bmatrix},$$

$$\tau = J^T W,$$

where $W$ is the wrench at the wrist, $f_1$ is the force at the finger tip, $r_1$ is the moment arm of the finger on the wrist, $z_i$ is the axis of rotation for joint $i$, $b_i$ is a vector from the origin of joint $i$ to the wrist, $\tau$ is the haptics control torque vector, and $J$ is the manipulator Jacobian. The force reflecting arm can be an exoskeleton worn by the user. In this example, the shoulder is spherical attachment from revolute joints 1, 2, 3. The elbow is joint four. The wrist is spherical, composed of revolute joints five, six, and seven.

### 2.8 Nonholonomic Constraints

Nonholonomic means nonintegrable in the mechanical engineering and virtual reality context. These are joints (or constraints) on the velocity or acceleration state of an object that cannot be written in terms of position alone. Driving a car is nonholonomic—the car can get to any position and orientation on a parking
lot, making position constraints impossible to impose, but the wheels cannot slide sideways at an instant, which is the velocity constraint that can be written [28]. Rolling contact conditions in the spatial case are nonholonomic.

The relations from our development of differential surface kinematics is the key to the analysis of general rolling, nonholonomic contact constraints in Chapter 8. Animating rolling contact through forward dynamics algorithms that apply friction forces, friction cone analysis, and other direct dynamics methods have also been used in many works. This computation is much more expensive and general than our purely kinematic application presented here. Automated means of rolling path generation has previously been studied for the case where one surface is flat and the other has an orthogonal parameterization [32]. In our work, we have a given path to follow and devise a method to compute the surface positions and orientations that allow rolling, or moving contact without slipping, along this path for arbitrary surface shapes and parameterizations.

The proof of whether or not a constraint is absolutely nonholonomic can be established using recent results in holonomy [243] or the Integrable Differential Constraint Theorem [28].

2.9 Geometric Satisfaction, Self-Assembly, Numerical Inverse Kinematics

The self-assembly, numerical inverse kinematics, and geometric satisfaction problems are essentially solving the same problem. A number of works in the robotics community have proposed solutions to the inverse kinematics problem [126, 26, 225, 177, 239], based on reduced (DH parameter) generalized coordinates. Some approaches from the graphics community have leveraged numerical techniques for geometric constraint satisfaction [161, 162, 7, 252]. The graphics methods have duplicated much of the work in the robotics literature but with a focus on interactivity.

Approaches taken in the computer aided design community have employed analytic, graphical methods [23, 73, 20]. However, these latter methods are not very general in scope since closed loops are a problem, and lack of choice of optimization
variables is a limitation.

The optimal inverse kinematics of closed loops was previously addressed [59] in reduced coordinates. We solve the general closed and open loop case in augmented coordinate representation of joints as constraints. In addition, we will generalize the open and closed kinematics so that all geometric variables are available as free variables, so that an optimal solution to any geometric parameter can be found. This generalization can include flexible coordinates as geometric parameters.

Mechanical design variation, known as kinematic and dynamic sensitivity analysis in the mechanical engineering literature, is the optimization of a mechanical design objective with respect to any design parameter so that additional constraints, typically assembly constraints, are met. The variational mechanical design methodology presented here allows interactive manipulation and optimization of an assembly. A designer might concurrently change assembly geometry and see the effects, for example, on the rest of the assembly. The framework has similarities to differential manipulation [161], self-assembly [95], and to the kinematic manipulation of open and closed loop systems in reduced coordinates [59, 177]. However, previous special-case assembly interaction techniques are generalized in Chapter 7 for all aspects of mechanical design variation so that any mechanical parameter can be a manipulation or optimization "free" variable.

2.10 Theoretical Kinematics

Several relations in the theory of kinematics are important in the next sections on differential geometry, haptics, and geometric constraints.

Kinematics is the study of motions of bodies without considering mass and force, while dynamics is the general study of motion. Forward kinematics maps from a body’s configuration variables to positions in Cartesian space and forward dynamics maps forces to body acceleration. Inverse kinematics and dynamics are projections in the other direction.
2.10.1 Calibration and Robotics Algebra

The derivations of surface kinematics relations in the next sections rely on calibration and robotics algebra. Several relations with skew-symmetric matrices are particularly important, and we will review them here.

Because a rotation matrix $R$ is orthogonal, we have

$$R^T R = I_{3\times3}. \quad (2.21)$$

Taking the time derivative,

$$\dot{R}^T R + R^T \dot{R} = 0_{3\times3}, \quad (2.22)$$

$$\dot{R}^T R = -R^T \dot{R}, \quad (2.23)$$

$$\dot{R}^T R = -(\dot{R}^T R)^T. \quad (2.24)$$

Thus $\Omega = \dot{R}^T R$ is skew-symmetric. It contains 3 independent components. By comparison with elementary relations from mechanical engineering, the independent components can be shown to correspond to the 3 components of angular velocity $\omega$. We may write the matrix $\Omega$ as $\omega \times$, since the matrix performs the cross product operation of angular velocity crossed with another vector.

A matrix that is known to be skew-symmetric may have its components extracted one by one [230]. If the matrix is only approximate, the diagonal components may be averaged (with the appropriate sign). Such methods occur in calibration optimization algorithms. We see that in Chapter 3 (where the matrix is exact) why that is a key method in deriving the surface kinematics for arbitrary surface parameterizations.

2.10.2 Body Velocity and Force Transformations

Several mappings from body-space angular velocity to velocity in various generalized coordinates are important in haptics and animation [28, 158].

The conversion of angular velocity $\omega$ to quaternion rates $\dot{q}$ is given by
\[ \dot{q} = G^T \omega. \]  
(2.25)

where \( G(q_{\text{rot}}) = \begin{bmatrix} -q_{\text{rot}2} & q_{\text{rot}1} & q_{\text{rot}4} & -q_{\text{rot}3} \\ -q_{\text{rot}3} & -q_{\text{rot}4} & q_{\text{rot}1} & q_{\text{rot}2} \\ -q_{\text{rot}4} & q_{\text{rot}3} & -q_{\text{rot}2} & q_{\text{rot}1} \end{bmatrix}. \)

It can also be shown that \[ \omega = \frac{1}{4} G \dot{q}. \]  
(2.26)

Suppose that a dynamics computation is done in Cartesian and quaternion generalized force coordinates \( Q \). It may be desirable to render the load (i.e., wrench 6-vector) to the haptics user. The mapping to convert these generalized coordinates is given by

\[ F = GQ/4. \]  
(2.27)

Also,

\[ Q = G^T F. \]  
(2.28)

\( G \) matrices are also available for generalized coordinates of Euler angles, Rodriguez parameters and other coordinate selections.
CHAPTER 3

DIFFERENTIAL SURFACE KINEMATICS

The formulation for extracting the floating contact surface Jacobian is derived here. By generalizing and reformulating a result in differential surface kinematics, derived in [11], we show that this relation yields a parametric coordinate update for minimum distance for haptics and virtual reality applications. In Chapter 5, the formulation is used to extract the extremal distance parametric floating contact Jacobian for trimmed surface placement optimization in Chapter 5, and for generalizing surface constraint derivatives for impact dynamics.

The mechanical engineering community has developed simple geometric constraints in augmented Cartesian coordinates, including the formulations, or catalogs, of typical constraint and constraint derivatives [28, 61, 148, 158]. We have chosen to adopt augmented coordinates (see Section 2.3) for the development of an analytical solution for the Jacobian of the surface constraint object.

3.1 Surface Contact Velocity Formulation

3.1.1 Arbitrary Regular Parameterizations

Eqn. 2.15 is based on the assumption that the surface partials of $f$ and $g$ are everywhere orthogonal, leading to simplifications based on a relation of $\dot{u}$ with the first and second fundamental forms, $I, II$. We reformulate these relations for regular parametric surfaces whose partials are not orthogonal, the common case, so the result in Eqn. 2.15 can be used for real models, not just surfaces of revolution or flat planes.

The minimum distance frames in Fig. 3.1 are defined with an orthonormal set of vectors. $R_f = [x_f \ y_f \ z_f]$ is the rotation matrix from the local contact frame to
Figure 3.1. Closest point surface contact frames. Velocity relations allow the contact coordinate velocities to be found.

The world frame, where

\[
x_f(u) = f_u / \| f_u \|
\]
\[
z_f(u) = f_u \times f_v / \| f_u \times f_v \|
\]
\[
y_f(u) = z_f \times x_f.
\]

\[R_g\] is similarly defined. Since \( u \) is the vector of closest point locations, \( z_f(u) \) and \( z_g(u) \) are parallel free vectors (Fig. 3.1).

An algebra of relative velocities\(^1\) describes the situation at the parametric contact coordinates \( u_f \) and \( u_g \), with surface contact velocities \( v^f \) and \( v^g \) and relative surface velocity \( v \),

\[
v^{g}_{x,y} + v_{x,y} = R_\theta v^f_{x,y} + \beta R_\theta \omega^f_{y,-x},
\]
\[
\omega^g_{y,-x} + \omega_{y,-x} = -R_\theta \omega^f_{y,-x}.
\]

These equations can be used to solve for \( \dot{u}_f \) and \( \dot{u}_g \) even when parametric surfaces do not have orthogonal partial derivatives in the parametric directions. We extract matrices \( E_{x,y}^f \) and \( F_{x,y}^f \) for surface \( f \) (and analogously for surface \( g \)) from the relations of linear and angular velocity,

\(^1\) A subscript with \( x \) or \( y \) such as \( a_{x,y} \) denote the first component of the vector. Several subscripts, such as \( a_{-x,y} \) represent a two-vector containing the negative of the first component and the second component of \( a \). A superscript \( T \) as in \( a^T \) denote a vector or matrix transpose. The operator \( S \) retrieves 3 independent components from 9 elements of a skew symmetric matrix.
\[ \mathbf{v}_{x,y}^f = R_{f_{x,y}}^T \dot{\mathbf{f}}_{x,y} \]  \hspace{1cm} (3.3) \\
\[ = \begin{bmatrix} x_f & y_f \end{bmatrix}^T \mathbf{f}_{u_f} \]  \hspace{1cm} (3.4) \\
\[ = \mathbf{E}_{x,y}^f \dot{\mathbf{u}}_f, \]  \hspace{1cm} (3.5)

where \( \mathbf{E}_{x,y}^f = \begin{bmatrix} x_f & y_f \end{bmatrix}^T (\mathbf{f}_{u_f})_{2 \times 3} \). The term \( \mathbf{f}_{u_f} \) is of size \( 2 \times 3 \) because it is the partial derivative of a 3-vector with respect to \( u_f \) and \( v_f \).

\[ \omega_{y,-x}^f = S(R_f^T \dot{R}_f)_{y,-x} \]  \hspace{1cm} (3.6) \\
\[ = \begin{bmatrix} x_f^T \mathbf{z}_{u_f} \\ y_f^T \mathbf{z}_{u_f} \end{bmatrix}_{2 \times 2} \dot{\mathbf{u}}_f \]  \hspace{1cm} (3.7) \\
\[ = \mathbf{F}_{y,-x}^f \dot{\mathbf{u}}_f. \]  \hspace{1cm} (3.8)

where \( \mathbf{F}^f = \begin{bmatrix} x_f^T \mathbf{z}_{u_f} \\ y_f^T \mathbf{z}_{u_f} \end{bmatrix} \).

Using Eqns. 3.1,3.2,3.5,3.8, the general nonorthogonal case is reduced to the 4 \( \times \) 4 system,

\[
\begin{bmatrix}
R_{\theta}(\mathbf{E}_{x,y}^f - \beta \mathbf{F}_{-y,x}^f) & -\mathbf{E}_{x,y}^g \\
(-R_{\theta}\mathbf{F}_{y,-x}^f)_{x,y} & -\mathbf{F}_{x,y}^g
\end{bmatrix}
\begin{bmatrix}
\dot{\mathbf{u}}_f \\
\dot{\mathbf{u}}_g
\end{bmatrix}
= \begin{bmatrix}
\mathbf{v}_{x,y} \\
\omega_{x,y}
\end{bmatrix}.
\]  \hspace{1cm} (3.9)

Thus is possible to relate the extremal distance parametric contact coordinates to the relative linear and angular velocity,

\[ \dot{\mathbf{u}} = \mathbf{A} \begin{bmatrix}
\mathbf{v}_{x,y} \\
\omega_{x,y}
\end{bmatrix}, \]  \hspace{1cm} (3.10)

where

\[
\mathbf{A} = \begin{bmatrix}
R_{\theta}(\mathbf{E}_{x,y}^f - \beta \mathbf{F}_{-y,x}^f) & -\mathbf{E}_{x,y}^g \\
-(R_{\theta}\mathbf{F}_{y,-x}^f)_{x,y} & -\mathbf{F}_{x,y}^g
\end{bmatrix}^{-1}.
\]  \hspace{1cm} (3.11)

This system can be solved quickly for \( \dot{\mathbf{u}} \) using the following algorithm for arbitrary surface parameterizations. We rewrite the inverse of the 4 \( \times \) 4 \( \mathbf{A} \) to be solved even more efficiently as a series of 2 \( \times \) 2 matrix inverses and multiplications.
Algorithm 3.1 Compute $A$ from $2 \times 2$ matrix inverses:

$$
H = F^g_{xy}(E^g_{xy})^{-1}(R_\phi E^f_{xy} + dR_\phi F^f_{xy}) - R_\phi F^f_{xy}
$$

$$
J = H^{-1}[F^g_{xy}(E^g_{xy})^{-1} - I_{2 \times 2}]
$$

$$
A = [J(E^g_{xy})^{-1}(R_\phi E^f_{xy}J + dR_\phi F^f_{xy}J - [I_{2 \times 2}, 0_{2 \times 2}])]
$$

**Proof:** Algorithm 3.1 can be justified using Eqs.3.1,3.5. We write

$$
\dot{u}_g = E^g_{x,y}^{-1}[R_\theta E^f_{x,y} \dot{u}_f + F^f_{-y,x} \dot{u}_f - v]. \quad (3.12)
$$

Substituting into Eqn. 3.2, we have

$$
F^g E^g_{x,y}^{-1}[R_\theta E^f_{x,y} \dot{u}_f + F^f_{-y,x} \dot{u}_f - v] + \omega = -R_\theta F^f \dot{u}_f. \quad (3.13)
$$

Collecting the $\dot{u}_f$ terms,

$$
F^g E^g_{x,y}^{-1}[R_\theta E^f_{x,y} + \beta F^f_{-y,x} + R_\theta F^f] \dot{u}_f = F^g E^g_{x,y}^{-1}v - \omega. \quad (3.14)
$$

Expressing this equation as a linear system, we have

$$
H \dot{u}_f = F^g E^g_{x,y}^{-1}v - \omega, \quad (3.15)
$$

which can be solved for the contact coordinates for surface $f$,

$$
\dot{u}_f = H^{-1}[F^g E^g_{x,y}^{-1} - I_{2 \times 2}] \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (3.16)
$$

$$
= J \begin{bmatrix} v \\ \omega \end{bmatrix}, \quad (3.17)
$$

where $J_{2 \times 4} = H^{-1} \begin{bmatrix} F^g E^g_{x,y}^{-1} - I_{2 \times 2} \end{bmatrix}$.

Now substituting this solution back into Eqn. 3.12,

$$
\dot{u}_g = E^g_{x,y}^{-1}[R_\theta E^f_{x,y} J \begin{bmatrix} v \\ \omega \end{bmatrix} + \beta F^f_{-y,x} J \begin{bmatrix} v \\ \omega \end{bmatrix} - v]. \quad (3.18)
$$

$$
= E^g_{x,y}^{-1}[R_\theta E^f_{x,y} J + \beta F^f_{-y,x} J - [I_{2 \times 2}, 0_{2 \times 2}]] \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (3.19)
$$
\[ J_b \begin{bmatrix} v \\ \omega \end{bmatrix}, \]  

(3.20)

Where \( J_b = E_{x,y}^{-1}[R_\theta E_{x,y}^f J + \beta F_{x,y}^f J - [1_{2x2} \ 0_{2x2}]] \). Thus, we can write

\[ \dot{u} = \begin{bmatrix} J \\ J_b \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \]  

(3.21)

The matrix \( A \) from Eqn. 3.11 is \( \begin{bmatrix} J \\ J_b \end{bmatrix} \), completing the proof. A comparison of this solution with previous special-case methods can be made. The inverse of two \( 2 \times 2 \) matrices and nine matrix multiplications, rather than four \( 2 \times 2 \) matrix inversions and six matrix multiplications, is required. Thus, our method requires 124 floating point operations as compared to 104 floating point operations in the previous methods. The hand-coded matrix inverse and multiplication implementation is a constant cost and is a small fraction of the cost associated with a surface evaluation.

### 3.2 Surface Constraint Jacobian Extraction

Optimization techniques use the Jacobians of the constraints and objectives to resolve the search space. An important Jacobian \( u_q \) is used in Chapter 5 on optimal surface placement and Chapter 9 on NURBS impact dynamics.

\( u_q \) is the Jacobian with respect to the generalized coordinates. To obtain it, we express the relative surface velocities in terms of world frame body coordinate velocities. \( \mathbf{q} = [\mathbf{q}_{tr}^T \ \mathbf{q}_{rot}^T]^T \) is mapped to the local contact frame through the rotation matrix

\[ R_{toc} = \begin{bmatrix} x_g & y_g & z_g \end{bmatrix}^T. \]  

(3.22)

Let \( \mathbf{G}_f = G(q_{f,rot}) \), \( \mathbf{G}_g = G(q_{g,rot}) \), where \( G(q_{rot}) \) is the matrix operator mapping quaternion velocities to angular velocities [28, 158].

The velocity \( [v^T \omega^T]^T \) is the motion of surface \( g \) relative to surface \( f \). We write our world space velocities in terms of the local frame so that relations in terms of our generalized coordinates is achieved. The relative velocities between the surfaces
can be related to the relative velocities between the surfaces at the contact frame by

\[
\begin{align*}
\mathbf{v} &= \mathbf{R}_{\text{loc}}(\mathbf{v}_{\text{sr},f} + \mathbf{\omega}_{\text{sr},f} \times (\mathbf{g}(\mathbf{u}) - \mathbf{q}_{g,\text{tr}})) \\
\mathbf{\omega} &= \mathbf{R}_{\text{loc}}\mathbf{\omega}_{\text{sr},f},
\end{align*}
\]

or, as a linear system,

\[
\begin{bmatrix}
\mathbf{v} \\
\mathbf{\omega}
\end{bmatrix} =
\begin{bmatrix}
\mathbf{R}_{\text{loc}} & -\mathbf{R}_{\text{loc}}(\mathbf{g}(\mathbf{u}) \mathbf{q}_{g,\text{tr}}) \\
\mathbf{0}_{3x3} & \mathbf{R}_{\text{loc}}
\end{bmatrix}
\begin{bmatrix}
\mathbf{v}_{\text{sr},f} \\
\mathbf{\omega}_{\text{sr},f}
\end{bmatrix}
\]

where \( \hat{\mathbf{a}} \) denotes the 3x3 skew symmetric matrix that performs the operation of a cross product (obtained from the three components of a vector, i.e., \( \hat{\mathbf{a}} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \)).

The relative surface velocities can be written in terms of the linear and angular velocities of the surfaces in the world frame. The angular velocities in the world frame for surface \( f \) in terms of generalized coordinates are \( \mathbf{\omega}_{f,\text{gl}} = \mathbf{R}_f \mathbf{G}_f \dot{\mathbf{q}}_{f,\text{rot}} \) and similarly for \( g \).

\[
\begin{align*}
\mathbf{v}_{\text{sr},f} &= (\mathbf{q}_{g,\text{tr}} - \mathbf{q}_{f,\text{tr}} + (\mathbf{\omega}_{g,\text{gl}} - \mathbf{\omega}_{f,\text{gl}}) \times (\mathbf{q}_{f,\text{tr}} - \mathbf{q}_{g,\text{tr}})) \\
\mathbf{\omega}_{\text{sr},f} &= \begin{bmatrix} -\mathbf{R}_f \mathbf{G}_f & \mathbf{R}_g \mathbf{G}_g \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_{f,\text{rot}} \\ \dot{\mathbf{q}}_{g,\text{rot}} \end{bmatrix},
\end{align*}
\]

Finally, we have the relative surface velocities \([\mathbf{v}^T, \mathbf{\omega}^T]^T\) as a function of Cartesian and quaternion velocities.

\[
\begin{bmatrix}
\mathbf{v} \\
\mathbf{\omega}
\end{bmatrix} =
\begin{bmatrix}
\mathbf{R}_{\text{loc}} & -\mathbf{R}_{\text{loc}}(\mathbf{g}(\mathbf{u}) \mathbf{q}_{g,\text{tr}}) \\
\mathbf{0}_{3x3} & \mathbf{R}_{\text{loc}}
\end{bmatrix}
\begin{bmatrix}
\dot{\mathbf{q}}_f \\
\dot{\mathbf{q}}_g
\end{bmatrix}
\]

From Eqn. 3.11, the truncated part \([\mathbf{v}^T_{x,y}, \mathbf{\omega}^T_{x,y}]^T\) is all that is required. Let \( \mathbf{B} \) contain the first two rows and rows four and five of the coefficient matrices in the right side of Eqn. 3.28. Substituting \([\mathbf{v}^T_{x,y}, \mathbf{\omega}^T_{x,y}]^T\) into Eqn. 3.10, yields

\[
\dot{\mathbf{u}} = \mathbf{A} \mathbf{B} \begin{bmatrix} \dot{\mathbf{q}}_f \\ \dot{\mathbf{q}}_g \end{bmatrix}.
\]

\( \mathbf{AB} \) is the Jacobian \( \mathbf{u}_q \) relating differentials \( \dot{\mathbf{u}} \) and \( \dot{\mathbf{q}} \). Further justification of this extraction is explained in Section A.1.
\( \mathbf{u}_q \) obtained from the analytical solution can be compared with the columns established by an expensive numerical finite differencing method. The numerical technique requires recomputing several minimum distances for each Jacobian column, as was done in [95]. Empirical evidence obtained over a number of examples suggests that the numerical method often loses 4 digits of accuracy when double precision computations are used. That is, the two methods are accurate for 12 digits. Since the differencing requires 4 samples of each surface for fourth order accuracy, and there are 14 columns, the differencing technique is two orders of magnitude more expensive than our analytic expression.

When writing the surface as a function of its augmented coordinates, the Jacobian of the surface contact point in the minimal and maximal context is the matrix

\[
\mathbf{S}_q = \begin{bmatrix}
\frac{\partial x}{\partial q} \\
\frac{\partial y}{\partial q} \\
\frac{\partial z}{\partial q}
\end{bmatrix}.
\tag{3.30}
\]

Since closed forms for the quantities \( \mathbf{S}_p, p_q, \) and \( \mathbf{S}_u \) are easily derived, the solution is given by

\[
\mathbf{S}_q = \mathbf{S}_u \mathbf{u}_q + \mathbf{S}_p p_q.
\tag{3.31}
\]

A summary of the advantages of this analytical surface constraint Jacobian \( \mathbf{S}_q \) and parametric contact Jacobian \( \mathbf{u}_q \) over numerical partial differentiation are

- It is more robust to compute than unstable numerical differentiation methods;
- It is two orders of magnitude faster than the numerical differentiation (three orders of magnitude for the parametric contact Jacobian \( \mathbf{u}_q \) to be defined later);
- It is possible to compute accurate higher order derivative tensors.
3.3 Higher Order, Higher Dimension Tensors of Derivatives

The gradient to the surface partials, for example, the third-order tensor \((S_u)_q = S_{uu}u_q + S_{up}p_q\), is also conveniently obtained once \(u_q\) is known. \((S_{uu})_q\) and so on is obtained similarly.

\(u_{qq}\) is needed in some engineering and optimization methods, for example computing Hessian matrices. This quantity is the Jacobian of a \(4 \times 14\) matrix, and this is a tensor, i.e., a 3D matrix array of size \(4 \times 14 \times 14\). The time derivative of Eqn. 3.29 is readily derived, producing

\[
\dot{u} = u_q\dot{q} + (u_q\dot{q})_q\dot{q} = u_q\dot{q} + u_{qq}\dot{q}\dot{q},
\]

which contains the tensor \(u_{qq}\). Numerically obtaining such a quantity \(u_{qq}\) would be very expensive and unstable since twice differentiating with finite differences is usually a sensitive operation.

In sum, we have implemented a computationally efficient, numerically robust, compact method for obtaining surface constraint Jacobian which are introduced in a number of applications in the next chapters. The surface kinematics relations are used in the local collision update for haptics interactions in the next chapter.

3.4 Nonorthogonal Surface-Curve Velocity Formulation

It can be shown that the surface-curve extremal distance equations may be extended to for arbitrary surface parameterizations. By writing the time derivative of the parametric contact coordinates for surface \(f\) and curve \(g\) as a function of linear and angular velocity multiplying a matrix operator, we obtain

\[
\dot{u}_{3x1} = \left[ \begin{array}{c} R_\phi(E_u^f + dE_{x,y}^f) - g_{u_{x,y}} \\ -R_\phi F_{x,y}^f - k_g \sin(\phi) g_{u_{x,y}} \end{array} \right]^{-1} \begin{bmatrix} v_{x,y} \\ \omega_y \end{bmatrix}_{3x1},
\]

where
\[ k_g = \frac{\|g_u \times g_{uu}\|}{\|g_u\|^3} \] (3.34)

and \( \phi \) is the angle between the curve normal (which is \(-z^f\) for the extremal distance case) and the curve binormal \( b \), about the curve’s tangent \( g_u \). The binormal is

\[ b = \frac{g_u \times g_{uu}}{\|g_u \times g_{uu}\|} \] (3.35)

Eqn. 3.28 is again used to establish this relation in terms of quaternion and Cartesian velocities, using rows 1, 2, and 5 of matrix \( B \) from Eqn. 3.28.
CHAPTER 4

SURFACE-TO-SURFACE TRACING

There is no closed form computational solution to an extremal distance between two arbitrary surfaces \( f \) and \( g \). Solving for a solution requires finding the roots to highly nonlinear systems of equations so numerical and polygonal approximation measures are used to find the roots. However, once a solution is obtained, the differential surface kinematics methods can be used to give a closed form solution to the problem of updating the solution. That is, given an exact current solution and the velocity equations of the models under consideration, Eqn. 3.29 provides local, instantaneously exact collision increments. This algorithm, coupled with a polygonal global “management” scheme and numerical initialization presented in Chapter 6, supports local collision updates with high order numerical integration of Eqn. 3.29. This analysis is used here for use in haptics surface-surface sculpted model tracing.

The integration of Eqn. 3.29 is useful for force computations in haptics virtual contact applications. Here we address the geometric portion of the surface-surface haptic rendering problem, namely, computation of proper penetration depth between two surfaces (Fig. 4.1). Once the penetration vector has been obtained, rendering force feedback is done with established haptics techniques [175]. (An alternative to this penetration depth-based force feedback, or stiffness/impedance display, is given in Chapter 9.)

The penetration depth computation is placed within a framework for reliably finding and tracking multiple contact points between models. This framework breaks the haptic rendering problem into several phases — distant tracking using global minimum distance methods (Chapter 6), nearby tracking using local
Figure 4.1. Virtual proxies are important in the penetrating case. The maximal distance is required by the haptics tracing algorithm. Global solution discontinuities such as “chopping through” an object are not desirable. Because the velocity formulation (shown) is the “most local,” it is the method of choice for the penetrating case.

Newton methods, and tracking during contact using the novel velocity formulation (Eqn. 3.29).

4.1 Distance Extrema

When a user touches a virtual surface with his virtual finger model, a curve $\gamma(t)$ embedded on the finger surface $f$ and a curve $\zeta(t)$ on the embedded on a model surface $g$ define the path of distance extrema required by the haptics tracing algorithm (Fig. 4.2).

$$distance \ extrema = ||f(\gamma(t)) - g(\zeta(t))|| \quad (4.1)$$

Our goal is to find the piecewise continuous curves $\gamma(t)$ and $\zeta(t)$ for penetration
Figure 4.2. Curves of distance extrema embedded into the finger surface model and the CAD model.

depth computations in a real-time haptics tracing environment.

When the finger model penetrates into the CAD model, the maximal distance is required for force computations. The minimal distance is required when outside the surface. When the finger does not penetrate the surface, the curves $\gamma$ and $\zeta$ may be discontinuous in general. The global monitoring and restarting mechanism is required for these curve discontinuities.

In the graphics literature, the minimum distance between models is zero during penetration; we desire the penetration depth for haptic force computations. The minimum distance between parametric surfaces $f(u_f)$ and $g(u_g)$ has been found using Newton optimization methods, in the four-dimensional parameter space [201], with global, high-dimensional resultant methods [185], and with Euclidean space bounding methods [205].

The assumption that the situation upon penetration is local for haptics tracing is fortunate because the haptics controller often requires the greatest update rates precisely at the impact or penetrating event. Greater stability is achieved with high update rates.

4.1.1 Comparing the Methods

The advantage to using Newton’s iterative method is that it has reasonable convergence to a solution given a close initial guess (which we obtain from a polygonal analysis). We use the Newton method during the nonpenetrating case so that we obtain an accurate starting point for the velocity method.
The advantage to the velocity space method is that it is an exact relation at that instant in time; it is not an iterative numerical method. Integrating $\dot{u}$ provides a highly accurate, strictly continuous tracing update. It is very well conditioned and does not suffer from the optimization problems of Newton’s method. However, it does not converge to the true minimal distance if it is given only an approximate starting point. It is a good algorithm for generating virtual proxy information because it is a strictly local distance update; the curves $\gamma(t)$ and $\zeta(t)$ are continuous, where they may be discontinuous in small intervals when they are computed with the Newton method. Discontinuities may be confusing to the user, as shown in Fig. 4.3.

4.1.2 Local Collision Updates

Integrating $\dot{u}$ over time is similar to using a local numerical technique for surface-surface extremal distance computations [204, 174]. Such methods are fast, as required for interactive and real-time control applications, but may get stuck in local minima. One resolution of this difficulty is to rely on fast refinement methods as a global monitoring mechanism [231]. The monitoring methods are used to detect jump discontinuities and can restart the velocity space method. The authors have found that the local method is the required computation for the penetrating case since discontinuities such as the global solutions of punching through objects in haptics is not desirable. Thus, the monitoring mechanism is used only for the nonpenetrating case.

4.1.3 Surface-Surface Tracing Results and Analysis

Running times on an SGI R10000 17.0 SPECfp95 machine for surface evaluations inside Alpha are approaching .07 milliseconds. Other operations, including computing the inverse of the $4 \times 4$ matrix and other restacking required in the velocity and Newton formulation, are not insignificant, but run in .01 milliseconds (that is, the cost of the methods excluding the surface evaluations). Thus, a single processor system can easily perform control and surface-surface analysis at several kilohertz update rates.
Figure 4.3. The velocity formulation maintains the most local, or continuous, solution through a global change in penetration depth so that discontinuities in force feedback is avoided.

We have used standard fourth-order numerical integration techniques. A very long tracing sequence, on the order of $10^8$ seconds for typical user motions, can be performed in practice with this integration technique without accumulating noticeable errors. Periodic “restarts” needed when the user transitions to the nonpenetrating condition and subsequent tracking by Newton’s method occur quite often. Even higher order integration methods can be employed if some unusual circumstance or application requires it without excessive cost due to the efficiency of our tracking techniques.

The tracing update formulations presented here provide improved surface tracing interactions. A fast, compact method for surface-surface updates for the nearly penetrating and penetrating case have been developed and analyzed. The velocity formulation has been introduced for use in haptics surface tracing.
CHAPTER 5

OPTIMAL SURFACE PLACEMENT

Another application of the surface kinematics relations from Chapter 3 is the extraction of the surface Jacobian from Eqn. 3.29 for use in surface configuration optimization problems (Fig. 5.1). General surface placement is important in CAD and animation applications, especially in the optimization framework as in our work, so that the best surface configuration for a design objective can be found or so that a surface configuration subject to kinematics constraints can be solved.

Surface Placement Application

Figure 5.1. Optimal floating surface placement in a design variation framework. An additional condition is included to avoid placement in trimmed away areas, such as the boundary of the region on the right side of the left figure. The parametric constraint Jacobian for trimming constraints is obtained in closed form through the surface-curve velocity relations (Eqn. 3.33). The global method coordinates with the local, analytic method to add and remove global collision constraints.
The parametric floating contact constraint Jacobians for the surface-surface and surface-curve cases allow a large class of optimal surface placement constraint Jacobians to be developed in closed form. This analytic solution is the key to achieving interactive rates in our system.

Previous interactive research has used springs, heuristics, and dynamics for surface placement applications [76, 16]. We formulate an analytical technique for kilohertz rate manipulation of CAD models with virtual surface and trimming constraints. The optimization approach allows best placement and sensitivity analysis for mechanical design objectives and parametric domain objectives. Such objectives cannot be incorporated into previous interactive methods.

The constraint analysis approach presented here allows the combination of surface assembly constraints and other mechanical constraints. We have implemented previous methods to solve constraint satisfaction problems from the graphics and CAD literature and extended them to handle surface constraints with our techniques, including the self-assembly or geometric satisfaction techniques [95, 141, 158, 162, 126], optimal surface placement [75, 76, 201, 16], and surface-surface floating attachment [95].

The constraint framework presented here is organized into:

- Symbolic constraint and constraint Jacobian objects (analytical evaluation of the constraint Jacobians),
- Numerical optimization on the constraints using the constraint Jacobian objects.

An augmented Lagrangian approach [252, 247, 187] has been implemented to carry out the solution to the constrained optimal surface configuration problem. The details are discussed in Chapter 7. The augmented Lagrangian method uses constraint derivative information to achieve interactive rates and convergence. The rest of this chapter shows the derivation of the Jacobian of the surface-surface and surface-curve constraints for the augmented Lagrangian algorithm.

The Jacobian of the constraint that two surfaces must touch, but without further constraint the location, is readily obtained as the partial derivatives of
\[ f(u_f) - g(u_g) = 0 \]  \hspace{1cm} (5.1)

since \( f_q \) and \( g_q \) are available from Eqn. 3.31.

If \( f \) or \( g \) has a curve embedded in it (Fig. 5.2), such as a trimming curve, we can write the constraint Jacobian of the following curve-surface constraint since we can extract the Jacobian from the curve-surface velocity method in Chapter 3.

\[ f^{trim}(u_{\text{curve}}^f) - g(u_g) = 0. \]  \hspace{1cm} (5.2)

Fig. 5.2 illustrates an initial configuration of an assembly. It has a surface-surface contact with trim constraints, although the initial configuration does not meet the constraint.

**Figure 5.2.** Unsatisfied surface-surface contact constraint showing minimal distance between the surfaces and surface-curve minimal distance. Several ways to write the surface-curve inequality constraint are available.
There are several options for writing the surface-curve unilateral or bilateral constraint for placement in a parametric area or avoiding trimming loops. We have identified and implemented three versions:

1. Parametric domain: \( u^f_{\text{extrema}} - u^f_{\text{trim}}(u^f_{\text{curve}}) = 0 \)

2. Cartesian domain: \( f(u^f_{\text{extrema}}) - f^{\text{trim}}(u^f_{\text{curve}}) = 0 \)

3. Cartesian domain between surfaces: \( g(u^g_{\text{extrema}}) - f^{\text{trim}}(u^f_{\text{curve}}) = 0 \),

where the subscript \( \text{extrema} \) refers to the extremal distance parametric coordinate between surfaces and \( \text{curve} \) denotes the extremal distance coordinate of the curve on one surface with respect to the other surface. \( f^{\text{trim}} \) is a 3D curve on \( f \). \( u^f_{\text{trim}}(u^f_{\text{curve}}) \) is a 2D curve embedded in \( f \).

Options 1 and 2 affect both surfaces \( f \) and \( g \) since the Jacobians of these constraints have 14 columns corresponding to the generalized coordinates of \( f \) and \( g \). The first two options are also coupled with a surface-surface contact constraint. The effect of the two constraints together is that surface-curve contact is maintained so that tangency between \( f \) and \( g \) is maintained as well, where \( f \) has the curve embedded in it.

The method of detecting when a surface contact point is on a trimmed away area, i.e., where a surface-surface constraint needs to transition to that of a surface-curve constraint, is the topic of the global management algorithm in Chapter 6. In summary, the optimal surface placement application employs constraints encoded as mathematical objectives and achieves interactive performance for haptics applications (Figs. 5.3, 5.4, 5.5) through the derivative techniques from Chapter 3.
Figure 5.3. OpenGL™ display and PHANToM™ force feedback interaction with a 3 paper clip surface constraint assembly.

Figure 5.4. PHANToM™ haptics interaction system with a virtual mechanism containing multiple surface constraints.
Figure 5.5. Kinematic manipulation of a spatial four-bar mechanism consisting of a universal joint, a spherical joint, and two revolute joints. The objective is to minimize the amount of separation of the top link with the hand grasp surface-surface floating contact.
CHAPTER 6
GLOBAL MANAGEMENT AND
TRANSITIONING

To compute the minimal distance between smooth NURBS surfaces requires the solution of a high-order nonlinear system of equations, in the general case. Because this solution cannot be found directly, previous research uses polygonal analysis or robust interval analysis [75]. The polygonal methods (Fig. 6.1) find an approximate solution quickly, while the interval methods find the solutions in a more general

Figure 6.1. The global polygonal minimal distance Proximity Query Package (PQP) (publicly available at http://www.cs.unc.edu/~geom/SSV) and the RAPID package [199] are used in the “far” regions or large surface - surface distances. This result, along with some parametric bookkeeping, starts a nonlinear root finger for tracking “near” cases. Each root is used and updated by the velocity formulation during the local tracking case.
setting but require a factor of approximately $10^8$ more time to evaluate.

Algorithms presented in this dissertation for computing surface-surface interactions, including the surface-surface haptics tracing algorithm, optimal trimmed surface placement computations, and surface impact dynamics implementation require a polygonal and numerical management algorithm. We leverage research results for global polygonal solutions [205, 175, 199] and use a fast numerical method to make the approximate results precise.

In addition, because the multisurface shells, or models, change position and orientation over time, the point of closest contact can change surfaces. A new transitioning algorithm has been derived to manage the global effects on a CAD model (as shown in Fig. 6.2) rather than for only single surfaces.

### 6.1 Initialization: Three-Step Process

We have broken our global management algorithm into three phases. In the first phase, the far phase, a global monitoring mechanism returns all portions of a surface within some distance of the other surface. The approximate solution from the global polygonal analysis provides an approximate starting point for the second and third phases. In the second phase, the near phase, local Newton methods are used to determine the closest points between all of the surfaces that were returned from the far phase. The third phase, the penetration phase, uses these closest points to initiate the velocity formulation that then is used to maintain the proper minimal distance or penetration depth vector between surfaces.

We have the following algorithm for the nonpenetrating and penetrating case [231]:

- **Far**: Global distance refinement, obtain approximate $u$
- **Near**: Newton iteration, obtain accurate $u$
- **Very Near and Penetrating**: Velocity formulation, updating (instantaneously) exact $u$.

Many CAD models consist of a set of surfaces, stitched together with trimming curves and topology information, forming a solid shell, or boundary representation.
We use prior results from polygonal approximations to monitor the global situation so that both the “Near” and the “Penetrating” cases can be initialized. The “Near” and “Penetrating” cases are algebraic computations which provide information to the optimization algorithm. In the “Far” case, for the global collision and transitioning algorithm, we tessellate, or polygonalize, the solid model, the set of trimmed NURBS surfaces. A modification to the Alpha_1 adaptive tessellation algorithm, sh\_mash, allows the polygonal manager to associate polygons with the surfaces they represent. Extremal distance schemes between unstructured polygonal soups.
can be thought of an approximation to the extremal distance between the smooth trimmed surfaces themselves. This method allows entire models, not just single surfaces, to be managed.

6.2 Multisurface Models

Transitioning between surfaces in a model during tracing or optimal surface placement is different from point-surface transitioning as used in [261]. With surface-surface transitioning in the concave case, a change in which surface is in contact does not involve crossing trims since there is another tracked point on the next surface if that patch is close enough to be part of the solution (see Fig. 6.3). The price we pay for the simpler, extrema-correct algorithm is tracking more potential points. When a surface contact moves outside of the trimming boundary, as can occur in the convex case, the contact is removed from the “active” list, but is still on the “tracked” list. The parametric contact values of inactive contacts are still updated in case they become active again.

Surface constraints may become inactive during the course of the optimization or manipulation of the virtual surface assembly. This research performs the detection and tracking of these events in the global management algorithm. When a contact event occurs, the corresponding constraint is added to or deleted from in the optimization algorithm, as appropriate.

The active contact point is changed in the surface-surface and dynamics impact algorithm. If a point becomes inactive, that point is still tracked since it is locally the closest one, but the point may have fallen into a trimmed away region as shown in Fig. 6.2. The point can become active again when it falls outside of the trimmed away region.

6.2.1 Surface-Curve Transitioning

When a point falls off either of the models, such as being in a trimmed area in the convex case, a surface-curve analysis takes priority and generates an active surface-curve constraint (and makes the surface-surface constraint inactive). Fortunately, the initial guess for the surface-curve case is largely solved: the surface parametric
Figure 6.3. Surface-surface transitioning can occur in the concave case. The surfaces lose contact in the convex case, making the contact inactive.

Contact coordinates for the surface are known because, at that instant of transition, the surface-curve and surface-surface coordinates have the first surface parametric contact coordinates in common (see Fig. 6.4). At that instant, both surfaces are touching as are the surface and curve, and they are touching at the same point. The Newton solver [116, 174] for the surface-curve case still needs to be run, but only one parameter, that of the curve contact, remains instead of 4 as in the initial surface-surface Newton iteration. This is fortunate since a global polygonal initialization is expensive to run, relative to the local surface kinematics relations, to initialize a Newton surface-curve solver.

Two algorithms have been used to detect when a surface-surface contact point is inside a trimmed away region. The even-odd edge crossing counter algorithm [280] in parametric space has been used as implemented in the Alpha_1 CAD system. The second algorithm is a point-to-tessellated surface test. This method finds the distance between the Cartesian surface contact point and the tessellated surface. The surface polygons do not contain regions that have been trimmed away. A Cartesian contact point that drifts off the valid surface region but is still in domain of the surface is detected by comparing its distance with the surface polygons. Since the trimming curves are piecewise linear themselves, this is a good approximation. The results obtained with this method depends on having
Figure 6.4. Close-up of a transition from a surface-surface constraint to a surface-curve constraint. Blue spheres are surface-surface constraints, while white are surface-curve constraints.

An optimal tessellation method available, such as Alpha 1’s shl mash utility, so that regions of high curvature have a sufficiently flat polygonal representation. A guaranteed bound is set when running the shl mash tessellator.

A modified version of the shl mash utility retains surface information with the polygonal representation. Each polygon has the closest parametric values associated with each vertex, parametric trimming value for polygons on trimming curves, and a surface identification information so that each polygon is associated with one of the surfaces on the multisurface model. When the PQP minimal polygonal distance analysis is run, the surface identification number and parametric contact information is used in the second (Newton) step to make the approximate distance precise.

6.3 Multiple Tracked Contacts

Several instances of the three-step process and transitioning algorithm are maintained or tracked in our system to allow multiple potential contacts between surfaces to be managed. A simple bookkeeping method is used to update each of the contacts
and check for transitions for each contact.

The initialization of more than one contact between surfaces requires additional logic over that needed for the single-distance PQP package to detect multiple locally minimal distance. A spatial normal cone hierarchy analysis is a fully general method to obtain multiple minimal distances \[205\]. Our algorithm has simpler requirements: it requires just the initialization for potential collisions. Outward offset surfaces are used to detect imminent collisions, using the publicly available RAPID and H-Collide packages \[199\] (both have been tried with similar results). Collision detection is also faster than a minimal distance query by an order of magnitude (a result we have observed in the lab for examples tried to date). For the dynamics application, collisions on the outward offset surface are used. For optimization, we need one PQP check for maintaining at least one contact constraint, and then the offset surface collision check to introduce additional constraints so that no part of the models interpenetrate. The global polygonal distance analysis needs to be run much less frequently than the fast, algebraic update methods from Chapter 3 because it is in the step 1 “Far” region. Currently, good results have been obtained by running this method on a separate processor though shared memory at 50 Hertz. The shl_mash program has been modified again to link offset parametric contact information from the offset surface polygons to the original surface through some bookkeeping measures.

In summary, the multitracked contact algorithm is

- **Far**:
  
  \[
  \begin{align*}
  \text{Optimization} & : \text{PQP(on original)} + \text{RAPID/HCollide(on offset)} \\
  \text{Dynamics} & : \text{RAPID/HCollide(on offset)}
  \end{align*}
  \]

  Global distance refinement, obtain all approximate \( u^i \)

- **Near**: Loop through \( i \) potential or active contacts; Newton iteration, obtain accurate \( u^i \)

- **Very Near and Penetrating**: Loop through \( i \) contacts; Velocity formulation, updating (instantaneously) exact \( u^i \).
6.4 Algorithm Details: Implementation

The details of the global management for the dynamics application are simpler than that of the optimal surface placement algorithm. In the dynamics, all potential contacts are tracked, and actual contacts are added as algebraic unilateral constraints in the Lagrange multiplier formulation. With optimal surface placement, at least 1 bilateral contact constraint is desired to remain between models, even if the contact degenerates into a surface-curve contact. The details of this optimization global management algorithm are given here, and illustrated in Fig. 6.5, since it is a super-set of the functionality required in the dynamics.

![Flow chart of the execution and data for the global management algorithm.](image)

**Figure 6.5.** Flow chart of the execution and data for the global management algorithm.
Each contact is referred to as being contained inside a sphere of neglect, since we are interested in the case where at most a single contact is considered inside a certain region, or sphere. Numerous contacts, as would be found from polygonal collision detection, are not desired since they are essentially redundant. The sphere is quite small in our implementation, on the order of .1 millimeter in radius for designs on the order of 1 cubic meter in volume. Several constants, such as the size of this region and the level of tessellation for initialization, are selected depending on engineering considerations.

First the particular bilateral constraint in our optimization algorithm is dereferenced for speed, with a local pointer handling this particular instance of the surface constraint. Better software engineering and performance characteristics are obtained from the local consideration inside the optimization evaluation loop.

The algorithm then invokes the bilateral constraint, which can be simple joint constraints, such as revolute joints, or the surface-surface or surface-curve floating contact constraints.

We examine the case of the surface and surface-curve constraints, since the other constraints are straightforward to evaluate. Tessellation information from the bodies is dereferenced and fed into the PQP global minimal distance package, and the offset tessellation information is input to the RAPID to check for imminent contact and to do initialization. The PQP and RAPID packages are only run periodically, since it is more expensive and is not used for initialization or global checks at every cycle.

At the beginning of each global management iteration, we set up model (multisurface shell) coordinate transforms. The translation and quaternion rotation evaluation for an instance of each model are evaluated. For this instance, we evaluate the global polygonal minimal distance and collisions on the model offset.

Next, we take all potential contacts returned from the polygonal analysis and obtain accurate results. Each result is checked to see whether it lies within a sphere of neglect (Fig. 6.6). The contacts are also checked for crossing trimmed away areas. When a contact falls off of a trim, the surface - surface constraint is
converted into a surface-curve constraint. The results from the global management and algorithm determine whether the surface constraints should be added to the potentially active list. Finally, the augmented Lagrangian optimization is run to perform the optimization.

**Transitioning Sequence**

![Transitioning Sequence](image)

**Figure 6.6.** Manipulation sequence with surface-surface transitioning. Close-up view. Blue spheres are active constraints, while white are inactive, but still tracked.
CHAPTER 7

INTERACTIVE MECHANICAL DESIGN VARIATION

A design variation technique for mechanical systems in augmented coordinates that advances techniques for the kinematic and force feedback interaction with virtual mechanical assembly design optimization at force control rates is presented. Its purpose is to interactively optimize mechanical characteristics while “self-assembling” or satisfying large systems of mechanical constraints. Such a high speed method is central to providing inverse dynamics force feedback in haptics and control applications. By using augmented coordinates for inverse dynamics of closed loop topologies further, performance advantages occur. The interaction framework supports manipulation of complex assemblies while maintaining kinematically admissible configurations though linkage and joint limit constraints. Furthermore, design variables such as link length can be treated as free variables and optimized to meet design criteria such as assembly dexterity.

This work focuses on high performance kinematic sensitivity analysis for assemblies and inverse dynamics force feedback for haptic interaction. A primary concern is scalability for large systems. The solution presented here applied to a system of geometric design objectives and constraints represented in augmented coordinates results in force control interaction updates. Kilohertz rate updates of kinematic and dynamic analysis make it possible to have more natural manipulation of assemblies in CAD, animation, and virtual prototyping applications. A general form of the interactive geometric design optimization problem is solved within this framework. An example is the perturbation of joint geometry for maximal dexterity so that the assembly constraints and additional mechanical joint limit constraints are met. For
example, one might want to set an object in a scene in a highly dextrous area to be picked up by an armature.

This approach provides an intuitive, natural setting for assembly design. The method includes elements that appear in immersive virtual environments:

- Concurrent viewing, manipulation, optimization, and force feedback,
- Navigation through the configuration space of a mechanism,
- Six DOF input extraction from a mouse (Fig. 7.1) and use of a *PHANToM*™ haptics device.

### 7.1 Interactive Design Variation

To solve or self-assemble the system using augmented coordinates, we construct a software constraint object [161, 148, 44] of symbolic equations that do not change form once they are formulated. A new system of equations is not needed for each different set of geometric design parameters that an animator or designer may want to choose. At a high level, we require the following constraint information:

\[
\begin{bmatrix}
C & C_\mathbf{q}, C_t, (C_\mathbf{q} \mathbf{q}) \mathbf{q}, C_\mathbf{q} \dot{\mathbf{q}}, C_{tt}
\end{bmatrix}
\]

The last grouping is required only for dynamic analysis, not kinematic design variation, but is included for completeness. The rows of \( C \) are the constraint equations, which when concatenated form a constraint manifold. In defining joints as constraints, we are building a mechanical graph in which bodies are nodes and constraints are edges. The graph indicates which constraints are used in the optimization.

In this work, elements of the design parameters \( \mathbf{q} \) may be removed and treated as constants. The user may set the variable parameters by “clicking” on a particular piece of geometry. The constants set by the user are manipulation parameter constants. We have found it intuitive to render the nonconstant parameters (except for part positions), such as variable link length, in wireframe to provide a clear indication of what the optimization algorithm is doing. The user is effectively
Figure 7.1. Manipulation of the geometric parameters of a 6 DOF Stewart platform. The best platform position and orientation so that assembly constraints and additional constraints, like mechanical joint limit stops, are met fits naturally within this framework. Right: Joints are edges and bodies are nodes for the graph of a mechanism. This Stewart platform has 3 revolute, 3 spherical, 3 universal joints, and one constraint on a ground body holding it immobile.

interacting with the graph of the mechanism by selecting which parameters are variable and which are constant.

The system includes appropriate columns $\mathbf{C}_q$, the partial of $\mathbf{C}$ with respect to each design parameter $q$, depending on which parameters the user wants to optimize (rather than manipulate or hold constant). Design parameters, and therefore Jacobian columns, can be removed or turned off with flags without reformulating the system equations. The elements of $q$ and $\Delta q$ are adjusted to correspond to the columns in the constraint Jacobian.
7.2 Interactive Numerical Optimization

A fast method of solving system equations is critical to enabling interactive editing. Haptic force feedback has a computational requirement at kilohertz rates. Our approach has been to abandon the more sophisticated, global optimization methods in favor of a simple, fast algorithm that maintains convergence in practice given solution coherence between iterations. We may maintain the assembly constraints by leveraging the Jacobian of the constraint manifold in an augmented Lagrangian optimization, similar to geometric satisfaction solutions from [95]. The Jacobian of the constraint manifold can be thought of as joint constraint force. Scaling this direction by the constraint violation, as in Eqn. 7.3, we can satisfy the joint constraints within some radius of convergence at high update rates.

Starting from the augmented Lagrangian method [187, 190] for optimizing $f$ subject to constraints $C$,

\[
\dot{q}_i = -f_{q_i} - \sum_j (\lambda_j \frac{\partial C_j}{\partial q_i} + k_1 C \frac{\partial C_j}{\partial q_i}), \\
\dot{\lambda} = k_2 C.
\]

(7.1)

(7.2)

Now restack Eqn. 7.1 into a concise form for use with the sparse Jacobian $C_q$,

\[
\Delta q = -f^{T}_{q} - C^{T}_{q}(\lambda + kC),
\]

(7.3)

and integrate $\dot{\lambda}$, $\Delta q$ to maintain the constraints and minimize $f$.

In practice, assembly constraints can be maintained once an admissible configuration is found by using quaternions as elements of orientation in $q$. The quaternions help linearize the optimization search space. Note that the use of Euler angles causes Eqn. 7.3 to fail in most circumstances. The effect of quaternion renormalization [1] is a concern in the iterative algorithm; we address this issue by adding the unit length quaternion condition $q_{rot}^{T}q_{rot} - 1 = 0$ as an additional constraint, one for each body. The constant $k$ in Eqn. 7.3 is a diagonal matrix of weights derived following the robotics calibration literature on scaling and rank.
Since the user’s hand does not move very far in the short update time, only a single iteration has been required. However, more sophisticated techniques are needed to get the system to converge initially. A separate processor runs graphics updates remotely at a much slower rate for visual feedback.

### 7.3 Generalized Inverse Kinematics

Manipulation using inverse kinematics is often useful in animation operations and kinematic workspace visualization. Some robust solutions from the literature do not require numerical means [23, 73] but are not very general in scope.

A largely neglected inverse kinematics problem is the manipulation of closed loop mechanisms (Fig.7.1). When a part is constrained to be attached to a finger, but no configuration can reach the position or orientation of the finger, some “best” or optimal solution is required. Closed chain inverse kinematics are found from the geometric satisfaction solution given in Eqns. 7.1-7.2. Using objectives \( f \) that indicate the norm of the point-finger distance must be minimized, or by adding finger constraints to the assembly, we have an effective means for manipulating a virtual mechanism with multiple fingers and hands. The best finger objectives are maintained during manipulation in a least “constraint-energy” energy sense, due to the formulation of the augmented Lagrangian formulation from [95].

#### 7.3.1 Multifinger, Multihand Manipulation

When multiple PHANToM™s or a MotionStar™ is available, it may be useful to incorporate more than one point contact or grasping contact on the assembly. In fact, these constraints are added as part of building the assembly initially and may be removed or included with flags as previously mentioned. When not all of the finger constraints may be met, as is usually be the case with low degree of freedom virtual mechanisms, the best “constraint-energy” solution results from our optimization method [95].
7.4 Alpha_1 CAD Specification of Virtual Mechanisms

The Alpha_1 modeling environment [197] has been extended to interface assemblies with the high performance design variation framework. Joint features are defined as local body frame vectors in the Alpha_1 SCL modeling language. For example, the quarter-torus in Fig.7.2 has two revolute joint features, located by $\vec{u}_1$ and $\vec{u}_2$, with directions $\vec{v}_1$ and $\vec{v}_2$ in the local frame of the part. A body, or graph node, associates the model geometry *surfaces* with the node in the SCL construct

\[ b : \text{body(surfaces)}; \]

Parts are assembled with connections, or graph edges indicating joint constraints, between bodies (graph nodes), such as

\[ c : \text{connection}(b1,b2,\text{array}(b1_u1,b1_v1), \text{array}(b2_u1,b2_v1),’\text{REVOLUTE’}); \]

The joint feature vectors can be constants or may be derived through inheritance or instantiation like any SCL modeling variable.

\[ \text{Figure 7.2. Quarter torus local body geometry. Two revolute joints are attachment features of this part. The assembled mechanism, or satisfaction of the geometric constraints is also shown. The ends of the linkage are attached, forming one closed loop.} \]
The entire assembly is defined with a list of connections

\[ a : \text{assembly}(c1,c2,c3,\ldots); \]

Special finger (manipulation) objectives or constraints are also added with connections in a similar way. The interaction thread spawns from the Alpha_1 modeling process to run the high performance haptics design variation. Alpha_1 models are updated with the results of this interactive optimization.

### 7.5 Mechanical Design Objectives

Objective functionals are used to determine degrees of freedom that remain after the constraints are satisfied. Minimal strain energy and dexterity measures have been encoded mathematically within our framework. Flexible body coordinates \( q_f \), shape functions \( S(q_f) \), and stiffness matrices \( K_{ff} \) can be introduced to model flexible part deformations. The strain energy to be minimized is the scalar \( q_f^T K_{ff} q_f \). The manipulability ellipsoid defined by the eigenvectors of the manipulator Jacobian \( J \) is the measure of the dexterity of a mechanism or a character’s hands. One manipulability measure is given by [26]

\[
f(q) = \sqrt{\det(J(q)J^T(q))}. \tag{7.4}\]

where \( J \) is the geometric velocity manipulator Jacobian.

### 7.6 Surface Constraints

The surface constraints from the velocity formulation and optimal surface placement fit into the design variation framework. Equations 5.1,5.2 and the derivatives of the equations can be used as elements of \( C \) and \( C_q \).
CHAPTER 8

ANIMATING NONHOLONOMIC ROLLING CONTACTS

Spatial rolling contact is a nonholonomic condition, or constraint. That is, the nonslipping condition can be written only in terms of the relative velocities between the bodies in contact. Except for special cases, the rolling condition is nonholonomic and is specified by the constraint that the relative velocities at the contact points are zero.

The underlying physics of rolling contact, or contact without slipping, has been studied for many years. New developments in surface representations and surface kinematics in the last two decades have allowed a more general analysis of this surface interaction.

The generation of kinematically admissible rolling contact motions is important in animation and mechanical analysis. We investigate a special case of animating rolling contacts for which the motion is prescribed by kinematic means alone. The case of pure rolling (without slip or twist) for one convex surface $f$ following a curve embedded in another surface $g$, without losing contact, for a prescribed velocity, can be determined by purely kinematic analysis. The kinematic constraints for this special case remove:

- One DOF for a twisting constraint about the contact normal
- One DOF for the contact constraint
- Two DOFs for nonslip in the contact plane

Suppose we have surfaces $f$ and $g$ in contact as defined in Section 2.2. Considering the motion of $f$ with respect to $g$'s coordinate frame, so that $g$ appears to be immobile, we see that 2 DOFs remain for the motion of $f$. 
The position and orientation of \( f \) to move so that the \( u \) contact coordinates can be found in closed form to follow a curve embedded in \( g \) at a rate \( m_C \). Since this is a unique solution, \( f \)’s motion can be determined by purely kinematic analysis. We call this special case *prescribed rolling motion* (Fig. 8.1) since the speed is given as \( m_C \) as input, the need to evaluate dynamics and inertias is removed.

Given an initial admissible contact configuration for \( f \) and \( g \), and a velocity \( m_C \) to follow the embedded curve \( c(t) \), a single solution exists for \( f \) to follow \( g(c(t)) \), \( t \in [t_0, t_n] \). An outline of the method, using the forward and inverse differential surface kinematics relations \( \dot{q} = \tilde{D}\dot{u} \) and \( \dot{u} = D\dot{q} \) from Chapter 3, \( D^{-1} \neq \tilde{D} \), is developed here.

### 8.1 Updating Parametric Surface Configurations During Rolling Contact Motion

The following sequence of surface operations computes the rolling contact motion. The curve is embedded on surface \( g \). The second surface rolls with a velocity relative to surface \( f \)’s contact frame \( (R_{wc}) \) so that in that frame, surface \( g \) is not moving. The generating curve is a mapping from \( R \) to \( R^2 \), from the curve’s parametric space to the parametric space of \( g \).

**Step 1: Set parametric and Cartesian speed on surface \( g \).** It is necessary to find the Cartesian velocities that correspond to the parametric velocities of the

![Figure 8.1](image)  
*Figure 8.1.* Tick marks to show nonslipping between a convex, bumpy spheroid and a rounded surface. The distance between corresponding ticks on each surface is the same up to the order of accuracy of the integration method.
generating curve. The rate at which to follow the curve is given as an argument, \( m_C \).

In our notation, the parametric input velocity at which the curve \( C \) (embedded on \( g \)) should be followed is given by

\[
\dot{u}_g = \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \dot{C}(t) \quad (8.1)
\]

The relative linear velocity on \( g \) for this parametric contact velocity is given by multiplying the desired speed constant \( m_C \) with the unit length velocity at the surface contact point

\[
v_c = m_C \frac{d}{dt} g(C(t)) = m_C \frac{\partial g}{\partial C} \frac{\partial C}{\partial t} = m_C \frac{g_u C_t}{||g_u C_t||} \quad (8.2)
\]

The distance from the world frame (denoted by subscript \( w \)) to the contact frame (subscript \( c \)) is given by

\[
L_w = (q_{tr,g} - f(u_f)), \quad (8.3)
\]

or in the local frame, by

\[
L_c = R_{wc} L_w. \quad (8.4)
\]

**Step 2: Equalize the speed on \( f \).** The surface kinematics mapping from relative contact velocities to parametric space contact velocities is used here. We identify what parametric contact velocities \( \dot{u} \) correspond to the goal of zero relative velocities, \( v = 0 \), with a contact velocity \( v_C \),

\[
v = 0 = v_c + \begin{bmatrix} 0 \\ 0 \\ L_c \end{bmatrix} \times \begin{bmatrix} \omega_{g_x} \\ \omega_{g_y} \\ 0 \end{bmatrix}. \quad (8.5)
\]

(Fig. 8.2). Therefore, \( \omega_g = [-v_{cy}/L_{cz} \ v_{cz}/L_{cz} \ 0]^T \), and

\[
\dot{u}_{4 \times 1} = A_{4 \times 4} \begin{bmatrix} 0_{2 \times 1} \\ \omega_{g_{x,y}} \end{bmatrix}. \quad (8.6)
\]

\[
\begin{bmatrix} 0_{2 \times 1} \\ \omega_{g_{x,y}} \end{bmatrix}
\]
has \( 0_{2 \times 1} \) since the relative velocity must be zero if the generated motion is to be kinematically admissible for nonslip.
Step 3: Compute the Cartesian velocity of f for nonslip. The second surface can move to produce the parametric contact velocities $\dot{\mathbf{u}}$ though linear and angular velocity relations (forward surface kinematics)

$$
\begin{bmatrix}
\mathbf{v}_{c,f} \\
\omega_{c,f}
\end{bmatrix} = \mathbf{A}^{-1}_{4\times4} \mathbf{u}_{4\times1}.
$$

(8.7)

Now update the surface positions and orientations by numerically integrating Eqns. 8.6, 8.8, 8.9.

$$
\dot{\mathbf{q}}_{t,f} = \mathbf{R}^T_{wc} \mathbf{v}_{c,f} - \omega_{w,f} \times \mathbf{L}_w, \\
\dot{\mathbf{q}}_{r,f} = \frac{1}{2} \mathbf{G}(\mathbf{q}_{r,f})^T \mathbf{R}_2^T \omega_{w,f}
$$

(8.8)  (8.9)

where, as shown in Fig. 8.2,

$$
\omega_{w,f} = \mathbf{R}^T_{wc} \omega_{c,f}
$$

(8.10)

$$
\mathbf{R}_2 = q2rot(\mathbf{q}_{r,f})
$$

(8.11)

The $\mathbf{G}(\mathbf{q})$ operator maps angular velocity $\omega$ to quaternion velocity $\dot{\mathbf{q}}$ as given in Section 2.10.1 [28, 158]. $\dot{\mathbf{q}}$ can be integrated and used for surface position and orientation coordinates. $q2rot$ is an operator that maps quaternion orientation into a $3 \times 3$ rotation matrix.

Figure 8.2. Angular velocity of the second surface is the cross product of the local contact velocity and the distance vector from the second surface body frame to the contact point.
CHAPTER 9

MULTIVARIATE SURFACE DERIVATIVES
IN DYNAMICS

Accurate dynamical simulation is becoming increasingly important in computer graphics, and geometric and functional modeling, as well as, more traditionally, mechanical engineering. For models that may have joints and other constraints, we present a novel technique to simulate collisions between models and track potential collisions accurately. It operates directly on the parametric algebraic surface representation and so improves methods that relied on numerical and polygonal approximations. The analytic formulation for the multivariate surface derivatives presented here supports dynamic animation of mechanical systems with colliding (nonresting) bodies, while a local, algebraic minimal distance update method supports the collision detection infrastructure. We are able to integrate the differential, constrained equations of motion over the duration of the impact, important capability in producing accurate physically-based simulation of collisions between bodies subject to constraints. Previous works derive velocities before and after the impact as a result of the impulses transferred [191]. These relations for collisions have not been derived for arbitrary parametric surface geometry. To combat common instabilities in integrating the differential equations, we use a matrix partitioning algorithm to obtain the system in terms of independent coordinates. We present a timing analysis demonstrating the interactive rates attained by our methods.

Computational algorithms and approaches have been derived for the physics of impact that assume that the surface constraint derivatives and contact points can be evaluated at arbitrary configurations. However, implementations have needed to compute numerical or polygonal approximations [94]. Our work, in concert with
previous dynamical formulations [191], allows computations directly on parametric surfaces such as NURBS and subdivision surfaces because our formulation requires only first and second surface derivatives along the parametric directions.

The goal of this dynamics formulation is to integrate arbitrary parametric surface contact constraints with Lagrange multiplier dynamics to analyze surface impacts. Very general, complicated joints or constraints can be described with Lagrange multiplier dynamics. The multiplier method uses implicit equations, which for this research have been implemented as modular constraint software objects, to define joint geometry and the derivatives of the geometry. In our algorithm, surface contact can be represented as a unilateral “joint,” and how the derivatives of this contact with respect to body coordinates can be derived with only two surface evaluations. An algebraic, instantaneously exact solution is obtained. Once formulated, the surface constraint can be used with any surface contact in the our modifications to the Lagrange multiplier algorithm. The additions to the dynamics algorithm admit a straightforward implementation since the surface constraint objects and surface impact velocity response follow directly from the analytic equations to follow.

The research presented here requires the surface floating contact Jacobian as extracted in Section 3.2. This development relates the fields of surface kinematics and surface impact physics. Then the surface contact Jacobian is used in a computation of the closed form solution to the change in momentum in such a way that the resulting velocities are valid over the entire constraint manifold. Solving for the change in momentum during collision events effectively engages the entire system of surface and joint Jacobians in a single, modular computation for the constrained system. We model systems that have both sculpted surface and lower pair joint constraints and consider all the bodies affecting the dynamics of the impact. The colliding surfaces are allowed to have kinematic attachments (such as joints or gear transmissions) to other parts of the mechanism. The surface and lower pair constraints (simple joints) are evaluated as algebraic constraints.

A recent result [191] for impact analysis of constrained systems for planar curve-
curve contact and simple joint constraints (i.e., spherical, revolute joints) was developed for mechanical optimization. This method uses planar contact constraints and simple joint constraints to analyze planar impacts. We extend this result to solve the spatial surface-surface collision case using differential surface kinematics.

Other recent research in impacts between free bodies has focused on the impulse method [94]. With impulse-based systems, collisions are used to represent all contact interactions between objects. For example, surfaces in resting contact are modeled as having many tiny, rapid collisions between the surfaces. More advanced research into impulse methods has focused on incorporating algebraic constraints [94]. This impulse is transferred to a kinematic chain through the use of an open loop manipulator Jacobian. However, systems with large number of kinematic loops and constraints are alternatively formulated in terms of the constrained equations of motion. We have chosen to formulate the entire dynamical system with algebraic constraints so that the surface kinematics relations can be leveraged. The differential surface relations, coupled with [191], allow efficient, analytic simulation of the system. Since surface impacts are affected by the accuracy of the surface normal and tangent, collision detection directly on the smooth parametric surface representation is an advantage of our method over the tessellation approximation used in [94].

This work does not focus on the resting contact case where broad areas of contact are maintained for a long duration of time. The noncolliding case is addressed in numerous formulations in the linear complementarity problem [11], quadratic programming, and a contact-force computation [17].

## 9.1 Dynamical Simulation

To solve for the accelerations of a dynamical system, two derivatives of the constraint (or kinematics) equations $C(q,t) = 0$ are coupled with the kinetic equations of motion. By letting the mass tensor be $M(q)$, undetermined multipliers $\lambda$, external forces $Q_e$, and forces quadratic in velocity $Q_v$, the equations can be written [148, 3]
\[ C_q(q, t)\ddot{q} = -(C_q\dot{q})_q\dot{q} - 2C_{qq}\dot{q} - C_{qt} \quad (9.1) \]
\[ M(q)\ddot{q} + C_q^T\lambda = Q_e(q, \dot{q}, t) + Q_v(q, \dot{q}, t). \quad (9.2) \]

Accelerations \( \ddot{q} \) are obtained by following the linear-time algorithm in [3] except for the collision response step. From mechanics, the Lagrange multipliers projected onto the constraint Jacobian, \( C_q^T\lambda \), give the vector of joint constraint force. When collisions occur, the multipliers corresponding to the surface constraints must be integrated over the duration of the impact [191].

### 9.2 Collisions and Momentum

Since most surface contact conditions transfer only compressive forces, we model surface contacts as unilateral (one-sided) constraints. Suppose a bilateral, non-penetration constraint between surfaces \( f \) and \( g \) is present, such as the floating (frictionless) contact constraint,
\[
C_{srf}(q) = (f - g)^T(f_u \times f_v) = 0. \quad (9.3)
\]

Then the bilateral constraints can be transformed to unilateral constraints through Eqns. 9.5-9.7, which split the constraint into admissible and inadmissible components after the impact event, as derived in [191]. The notation defines \( L = M^{-1} \), while all \( k \) surface constraint Jacobians are incorporated into the matrix \( W_k^N = (C_{qf,k}^T)^T \) and the other constraint Jacobians are \( W = C_q^T \).

When impacts occur, the Lagrange multipliers and accelerations are integrated over the collision to produce a relation of velocities before and after the collision. The Lagrange multipliers are computed from the rules derived in [191],
\[
\lambda_N = -H_N^{-1}(1 + \epsilon)W_N^T L r. \quad (9.4)
\]

where \( \epsilon \) is the coefficient of restitution, and \( H_N \) and \( r \) are given by
\[
H_N = W_N^T L_0 W_N, \quad (9.5) \\
\alpha = 1 - L W H^{-1} W^T, \quad (9.6)
\]
\[ \mathbf{r}^+ = \mathbf{r}^- - \alpha^T \mathbf{W}_N \lambda_N. \]  

(9.7)

The \( ^+ \) and \( ^- \) superscripts refer to the time after and before, respectively, the impact event. \( \mathbf{1} \) is the identity matrix, and \( \alpha \) is measured at the time of impact. This formulation produces a change of velocities that is kinematically admissible; i.e., the velocity satisfies the assembly condition \( \frac{d}{dt}(\mathbf{C}(\mathbf{q}^+, t) = \mathbf{0}) \rightarrow (\mathbf{C}_q \mathbf{q}^+ = -\mathbf{C}_t) \) after the collision. Integration over the differential equations of motion now must reflect the change in \( \mathbf{r} \) given by \( \mathbf{r}^+ \).

From Eqns. 9.5-9.7, it is clear that having an accurate and fast computation of \( \mathbf{W}_N \) is central to the impact analysis. A closed form solution for these Jacobians can be obtained using the formulations presented in Section 9.3.1 on derivative evaluations, an improvement over previous works that hold only for specialized cases or for planar (curve) contact.

### 9.3 Jacobian Computations

Computing \( \mathbf{W}_N \) is the central analysis for impact simulation in our work; it indicates the direction of collision response that is valid for the assembly velocity constraints. The dynamic simulation implemented here modifies and adapts previously developed algorithms [3, 28] to be valid for collision events over the constraint manifold.

#### 9.3.1 Surface Constraint Jacobian Extraction

The surface Jacobian was derived in Chapter 5 for use in optimization. The Jacobian we use here is similar, except that we are taking the partial derivative of Eqn. 9.3. The Jacobian is obtained, of size \( 1 \times 14 \), as

\[
\mathbf{W}_N^T(\mathbf{q}) = \mathbf{C}_{\mathbf{q}}^{\text{surf}}(\mathbf{q})
= \partial \left( (\mathbf{f} - \mathbf{g})^T(\mathbf{f}_u \times \mathbf{f}_v) \right)/\partial \mathbf{q}
= (\mathbf{f}_u \times \mathbf{f}_v)^T(\mathbf{f}_q - \mathbf{g}_q) + (\mathbf{f} - \mathbf{g})^T(\mathbf{f}_u \times \mathbf{f}_v)\mathbf{q}
= (\mathbf{f}_u \times \mathbf{f}_v)^T(\mathbf{f}_q - \mathbf{g}_q).
\]

\( (\mathbf{f} - \mathbf{g})^T \) tends to the zero vector when proper backtracking to the collision is implemented. (Note that we cannot use the surface constraint \( \mathbf{C}_{\mathbf{q}}^{\text{surf}}(\mathbf{q}) = ||\mathbf{f} - \mathbf{g}|| \)
since the Jacobian would be $C_{\text{srf}} = \|f - g\|^{-1}(f - g)^T(f_q - g_q)$, requiring a division by 0. $C_{\text{srf}}(q) = \|f - g\|^2$ has a similar problem.)

9.4 Surface-Curve Constraint Jacobian Extraction

The surface-curve contact constraint Jacobians can be defined from the surface-curve version of $\dot{\mathbf{u}} = D\dot{\mathbf{q}}$ from Chapter 3, where one curve contact coordinate and two surface contact coordinates indicate that $\mathbf{u}$ is $3 \times 1$ and $D$ is $3 \times 14$ or $3 \times 12$ for quaternion and Euler angle coordinates, respectively. The surface-curve contact case frequently arises when a surface hits an edge of another object. The edge can occur as the result of a trimming or boolean operation between NURBS models. The edge is often defined by a parametric curve embedded in the trimmed surface. These cases use the same methods as the surface-surface contact condition. Additional logic is required to detect whether the contact condition is between surfaces or edges. This is done through additional surface-curve tracking and management, presented in Section 6.

9.5 Partitioning: DAE $\rightarrow$ ODE

One criticism of the Lagrange multiplier method has been the unstable nature of the system in Eqns. 9.1-9.2. Integration of $\ddot{\mathbf{q}}$ over time to obtain $\mathbf{q}$ and of $\dot{\mathbf{q}}$ to obtain $\mathbf{q}$ does not satisfy $C(q, t) = 0$ at subsequent timesteps, independent of whether the Lagrange multipliers are eliminated first. This is because the system of equations is differential-algebraic equations (DAEs), with the unknown differential term $\ddot{\mathbf{q}}$ and unknown algebraic term $\lambda$. The graphics literature has previously used stabilization techniques, such as the method in [14] to combat the instability problem, but this method does not reduce the system to an ordinary differential equation (ODE) [30]. ODEs are desirable because the assembly geometry is not violated due to integration errors.

A more stable method is to find the independent and dependent coordinates of the redundantly parameterized system $C(q, t) = 0$. It is possible to systematically reorganize the system of equations in terms of only the independent equations in an easily integrated ODE [28], while maintaining the ability to specify the system
in redundant coordinates so that surface contacts can be handled through the Lagrange multiplier method. To do this, we partition the constraint Jacobian into columns corresponding to dependent and independent coordinates \( q = [\xi \mid \zeta]^T \) through Gauss-Jordan elimination,

\[
\frac{\partial C}{\partial q} = C_q = [C_\xi \mid C_\zeta]. \tag{9.8}
\]

It has been shown [248] that the Lagrange multipliers can then be determined uniquely as a function of \( \zeta \), and the system of equations reduces to the ODE

\[
\hat{M}\ddot{\zeta} = \hat{Q}(q, \dot{q}, t) \tag{9.9}
\]

where

\[
\hat{M} = M^{\xi\xi} - M^{\xi\zeta}C_\zeta^{-1}C_\xi - C_\zeta^TC_\zeta^{-1}T[M^{\xi\xi} - M^{\xi\zeta}C_\zeta^{-1}C_\xi],
\]

\[
\hat{Q} = Q^\xi - M^{\xi\zeta}C_\zeta^{-1}\tau - C_\zeta^TC_\zeta^{-1}T[Q^\xi - M^{\xi\zeta}C_\zeta^{-1}\tau],
\]

\( \tau \) is the right hand side of Eqn. 9.1, and \( Q = Q_e + Q_v \). A nonsingular matrix \( C_\xi \) is always obtained by Gauss-Jordan elimination for mechanical systems with nonredundant constraints (a nonredundant system can always be found for mechanical systems). Suppose the Gauss-Jordan elimination returns a permutation matrix selecting independent columns \( P^i \) and dependent columns \( P^d \). \( M^{\xi\xi} = P^iMP^i, \)

\( M^{\xi\zeta} = P^iMP^d, \) \( C_\zeta = C_qP^i \) and analogous notation for the other terms \( C_\xi, Q^\xi, \) \( Q^\zeta \).

The method carries cubic cost in the number of dependent constraints to evaluate \( C_\xi^{-1} \). The dependent variables \( \xi \) must be found after each timestep from \( \zeta \). We may fix \( \zeta = \zeta^* \) and apply numerical iteration to solve \( C(\xi, \zeta^*) = 0 \) for \( \xi \). These steps are computationally expensive for large systems since they have cubic cost. However, large systems and many surface impact systems can exhibit unstable behavior, so we judge the cost tradeoff worthwhile.

### 9.6 Surface Self-Assembly

The dynamic constraints from Eqns. 9.1,9.2 and those including surface constraints must be satisfied for each step of the simulation, i.e., \( C(q, t) = 0, \forall t, \)
so that the assembly remains assembled. Finding an initially valid configuration $q$ may be simple for mechanisms with simple joints, but complications can arise with kinematics loops or surface configurations. It may be desirable to determine a “best” surface placement through mathematical programming [95].

Our approach has been to use an augmented Lagrangian optimization to find $C(q, t) = 0$. Many optimization-based solvers and interval analysis methods use some variant of Newton iteration or derivative information to improve performance [270]. Our root finder is two orders of magnitude faster when gradient information $u_q$ and $S_q$ from Section 9.3.1 is available for surface placement objectives and constraints. The implementation of the root finder through optimization is presented in Chapter 5.

The initial velocities used in the dynamics computation must also be kinematically valid. If the desired independent velocities $\zeta$ are known, then the dependent velocities can be obtained from $C_\xi \dot{\xi} + C_\zeta \dot{\zeta} = 0$, since $C_\xi$ is invertible.

9.7 Local Collision Detection and Extremal Distance Updates

As introduced in Chapter 3, the velocity formulation $\dot{u} = D\dot{q}$ can be used to update distance and collision checks when the distance function between the two surfaces under consideration is continuous, i.e., where the curve $u$ embedded simultaneously in $f$ and $g$ is continuous. This method does not require convex surfaces. However, the local method does not handle the case of discontinuous change in minimal distance. This limitation can be removed by using the global minimal distance management algorithm from Chapter 6. When two surfaces penetrate, the distance returned is the locally maximal penetrating distance. Multiple potential contact points can be tracked with this management algorithm and updated quickly through the velocity formulation for the local collision case.

9.7.1 Backtracking in Time

When impacts are detected by the local distance algorithm, as shown in Fig. 9.1, some amount of penetration usually has occurred unless one is fortunate to en-
Figure 9.1. The teapots approach and penetrate in multiple places, followed by backtrack to first contact. Local collision updates with the velocity formulation return maximal penetrating distance after impact occurs for algebraic collision analysis for smooth surfaces and better backtracking. An alternative is to use the minimal distance to predict time of impact, but penetration can still occur no matter how small the timestep.

counter the exact instant when the two models just touch but do not interpenetrate. (We do not consider the problem of missed collisions between polygons for initialization due to discrete time updates, as previously addressed in [21].)

For an accurate dynamical simulation, one must either advance to an impending collision or backtrack the interpenetrating models to the point at which they make first contact. Numerical methods usually get stuck at extraneous solutions upon penetration [174, 231], although a gradient descent method [201] has been proposed to more reliably find the maximal penetrating distance for typical surface contacts. The velocity formulation always returns the (locally) maximal penetrating distance
since it is an (instantaneously) exact computation and not an optimization method.
The velocity formulation holds for the extremal distance case, for minimal distance
or maximal penetrating distance.

With the computation of the maximal penetrating distance, we can measure
the distance before the collision and the amount of penetration after the collision.
This distance is used to interpolate model positions, i.e., reposition the model and
step backwards in time, to obtain a better estimate on when the exact moment
of impact occurred. The process of backtracking, moving forward until impact,
and interpolation may be repeated several times to obtain a very accurate time
of impact. Two iterations have been sufficient to obtain an error of $10^{-8}$ units of
distance in common examples on the order of 1 cubic unit in volume, such as the
model in Fig. 9.2.

9.8 Inverse Dynamics: Applying the Surface
Jacobian to Rendering Virtual
Contact Force in Haptics

Previous augmented reality displays have rendered crisp virtual walls and virtual
objects with finger-surface reactions through haptics feedback devices. The widely
used stiffness (or impedance) methods penalize a user’s intrusion into a virtual
surface with a force response dependent on the amount of penetration along the
contact normal at the virtual proxy location (see Chapter 4). We present an
alternative approach for computing the surface impact response that multiplies
Lagrange multipliers associated with a unilateral surface constraint onto a mul-
tivariate surface constraint Jacobian for the case where the two objects (finger
and model) do not stick together. Advantages of our method are that the surface
Jacobian is already being computed from the local collision update algorithm, and
that the force response is accurate to numerical precision; no polygonal or numerical
approximations are employed.
Figure 9.2. Surface constraint and joint constraint Jacobians and Hessians allow impacts between surfaces to be analyzed while respecting the algebraic joint constraints.

9.8.1 Forward and Inverse Dynamics

Stiffness methods for virtual surface response are similar to inverse dynamics methods (the Newton-Euler method \[159\], for example) in that they measure positions and velocities, and return or render forces. The goal of dynamical simulation such as the impulse dynamics work \[94\] is very different; external forces like gravity are applied to objects, and accelerations are computed to animate the objects. Our system requires both forward and inverse dynamics computations. The virtual mechanism is affected by gravity and impulsive forces from the virtual finger. The accelerations of the virtual mechanism are computed so that the object positions can be updated in the graphics display. However, the acceleration of the user’s finger surface is known because it is measured from our haptics
position sensors. We return force, or inverse dynamics, to the user by leveraging a by-product of the Lagrange multiplier method: the Lagrange multipliers are the magnitude of the generalized constraint forces for all the joints in the assembly. In particular, the surface constraint force is computed from the evaluation of the multipliers.

9.8.2 Computing the Surface Impact Response Force

From the principle of virtual work, it is known that the Lagrange Multipliers $\lambda$ associated with an algebraic constraint $C$ multiplied by the Jacobian of the constraint $C_q$ are the vector of generalized joint (constraint) reaction force, i.e., is $C^T_q \lambda$ [28].

A finger surface, $f$, and another surface, $g$, are constrained to be nonpenetrating, using Eqn. 9.3 for the finger $f$ and mechanical model $g$ for surfaces parameterized to have an inward normal convention (otherwise, $(f_v \times f_u)$ can be used for the normal). Upon impact, the first half of the vector $C^N_q T \lambda_N$ can be rendered to the user as generalized impact force due to the collision (where $q = [q_f^T \ q_g^T]^T$).

The generalized force of constraint between the user’s finger and mechanical model, $W = C^N_q \lambda_N$, can be rendered to body space force and torque through standard kinematic mappings $G(q)$ as given in [28, 278]. The force returned to the user is evaluated from $T = GW$. For our implementation with the PHANToM haptics device, the first three components of $T$ are used. Higher DOF devices such as the SARCOS Dextrous Arm$^{TM}$ or large PHANToM can make use of all six components of the wrench $T$.

9.8.3 Surface Impact Feedback Results

The penetration depth in the surface Jacobian method is an artifact caused by the maximum sampling frequency of 1000 Hz from the PHANToM haptics device drivers. In practice, the amount of penetration was on the order of $10^{-4}$-$10^{-3}$ millimeters for typical user hand motions. The stiffness method produced greater penetrations, and for multiple cycles, from $10^{-3}$-$10^{-1}$ millimeters for the same mechanism, same collision detection algorithm, and similar collisions. The penetration
results are similar to the point-surface penetrations given in [175].

The entire constraint manifold is analyzed at impact as shown in Figs. 9.3 and 9.4.
Figure 9.3. Impact sequence between a virtual finger (the Utah Teapot) and the virtual assembly.

Figure 9.4. Rendered, separate frame sequence.
CHAPTER 10

RESEARCH DISCUSSION AND RESULTS

The differential surface kinematics relations and Jacobian extraction from Chapter 3 follow (and are proven) from the steps of the derivation. This Jacobian formulation exhibits high update rates and good numerical stability, which allows our surface-surface interaction applications to attain force-interactive rates.

10.1 Timing Complexity of Surface Kinematics

The Jacobian resulting from that derivation and an expensive high order numerical approximation of the Jacobian for numerous sample points in several data sets has been carried out for timing and accuracy comparisons. The timing results show several orders of magnitude speedup of the closed form approach. The difference is due to the expense involved in the numerical differentiation on each column of the Jacobian rather than obtaining all of the columns in a single computation as is done with the analytical method. The cost of the analytical approach has been shown to be well within the update rates of that required for haptics force feedback interactions, on the order of several thousand updates per second. Even when divided by the cost of evaluating several instances of surface contact, this rate is sufficient for manipulation in a haptics framework.

The differential surface relations can be evaluated with only two surface evaluations, where a surface evaluation consists of the point, partial derivatives, and second partial derivatives. Using Alpha matrix refinement operations [197], a surface evaluation can be done in excess of 30,000 times per second on an R10000 SGI Onyx 2. Other computational terms become important since the surface derivatives are no longer the dominant cost. Our implementation allows the surface
constraint objects and the Jacobians and Hessian information to be evaluated at 4,000 times per second.

The sizes of the control meshes of the surfaces being considered do not slow down the local constraint analysis since the surface evaluations are done with Alpha refinement operations. The global constraint tracking depends on the complexity of the surfaces being analyzed, but a hierarchical, polygonal global management scheme is employed that reduces this cost so that it is not the bottleneck in the parallel management implementation.

A high-order finite differencing scheme \[120\] has been used to obtain the same result as the analytic solution to 11-12 digits of accuracy, depending on the step size \(h\). It is notable that making \(h\) too small causes instability in some cases, thus limiting the accuracy of the numerical differentiation. An even higher order method is possible to implement, such as an eighth-order formulation, but more samples are required, making the timing complexity even worse for the numerical approach.

10.2 Timing Complexity of the Surface Interaction Applications

10.2.1 Surface-Surface Tracing

The surface tracing algorithm is the same as the surface kinematics algorithm. The global management algorithm runs only periodically (at 50 Hertz) to track potential global state changes and is therefore not a direct factor in the surface-surface tracing complexity. The global management algorithm can also run in a separate process. We depend on the ability to track collisions with minimal distance before objects actually collide, such as when a user’s finger comes into proximity with a surface, and do not consider the class of collisions from \[21\]. The complexity of global polygonal distance algorithms depends on the distance between models, curvature of the model geometry, and number of geometry polygons, so a straightforward explanation of the complexity bound is not available besides a loose upper bound of quadratic cost in the number of polygons. When model distances are small, the cost increases. In this case, the algorithm depends on local update methods and does not require input from the global algorithm.
10.2.2 Optimization of Surface-Surface Contact Configurations

A surface contact constraint can be evaluated in constant time, as in the surface-surface tracing algorithm. A single processor can evaluate \( n \) surface constraints and other joint constraints in linear time. The constraints do not depend on each other within an iteration, so they could, in theory, be evaluated in constant time in theory. The sparse matrix algorithm used in the augmented Lagrangian optimization can evaluate an iteration in constant time given enough processors as well. In our implementation, both constraints and optimization are done on a single processor with linear cost. The total cost is therefore the linear cost multiplied by the number of iterations required for convergence. The augmented Lagrangian implementation follows that in [187]. The convergence of this method is super-linear, i.e., between linear and quadratic [270]. Alternatives to our solution approach include a “quadratically” converging solution [252] that proposes a matrix inversion algorithm to achieve linear time computation, but uses approximations that degrade the convergence speed of their method. An implementation with alternative approaches to optimization is a subject for future work.

10.2.3 Surface Impact Dynamics

The Lagrange multiplier method in our system allows closed loops in both the Matlab and the C++ implementation. This method is cubic in the cost of the dimension of the constraint manifold (constraint vector) plus a linear term in the number of generalized coordinates that describe the system. A linear cost version from [3] for open loop systems has also been implemented in Matlab. Most of our Alpha_1 models contain closed loops, so the linear method [3] is not used since it carries cubic cost in that case as well. The same surface and simple joint constraints as in the optimal surface configuration work is used, although in terms of a different set of generalized coordinates. These constraints can be evaluated in linear single CPU time or constant parallel time for one dynamics timestep. Thus, the cubic cost of obtaining the Lagrange multipliers associated with the constraints is the dominant cost.

The five body, five constraint system shown in Fig. 10.1 required five milliseconds
Figure 10.1. An initial angular velocity is given for the Utah teapot as it drops onto the Lego assembly.

to compute the impulse response during a collision and 1.8 milliseconds to compute the Lagrange multipliers when collisions are not occurring. These costs are much more than the cost to evaluate the surface constraint derivative information. The 5 body example can therefore be updated interactively, or at marginally interactive rates with respect to force control updates for the integration with an external haptics device.

When a dynamics system exhibits instability, the Matlab implementation repartitions the constraint manifold to obtain an independent set of generalized coordinates, as described in Section 9.5. The integration of the resulting equations is more stable because an ODE rather than a DAE results, but the repartitioning algorithm cost is cubic in the size of the constraint vector.
We have experimented with a number of ODE integration methods, including Euler’s method, Runge-Kutta and adaptive methods. Numerical integration is not the focus of this work, but we note that the simple stabilization schemes did not work as well as the partitioning algorithm from Section 9.5. The norms of the constraint and velocity violation $||C||$ and $||C_q\dot{q}||$ were measured for comparison. $||C||$, the error due to numerical integration of the dynamic equations of motion, was less than $10^{-5}$ for mechanisms about 1 cubic unit of volume for real-time interaction timesteps $dt = 0.01$ (.01 second integration timesteps). For the impulse response, the velocity error $||C_q\dot{q}||$ at a collision event for constraints that are exactly satisfied before the collision are on the order of $10^{-14}$, or near the limit of double floating point accuracy. Ignoring the backtracking step (allowing some penetration) causes significantly more error in most cases, from a factor of two up to several orders of magnitude.

10.3 Global Concerns

While global convergence for large systems is a hard problem in general, a number of global management measures have been introduced to reduce the complexity of the problem. Previous work in global collision management is leveraged [185, 199], accurate extremal distance updates are used, and checks for constraint addition and deletion have been employed. In addition, the collision management scheme also allows the trimming curve extremal distance updates to be maintained.

The norm of the constraint manifold, a constraint violation measure, is maintained to less than $10^{-5}$ for simple mechanisms on the scale of 1 unit in size. This measure is generally considered to be converged in the calibration literature [88].

10.3.1 Conditioning

Empirical measures of the condition number of the parametric contact Jacobian and the surface Jacobian are very close to 1 for a typical set of measurements of model-model contact conditions; values less than 1.001 are observed regularly. This conditioning gives rise to good local behavior for our surface-surface interaction applications. Analytic evaluation of the condition number is a hard problem and
not investigated here. The values observed to date in the nonsingular configuration points generally range from 1 to 1.1. Condition numbers up to 200 are generally considered acceptable during optimization in the calibration literature [88].

Special cases where an infinite number of extremal distance points exist cause a singularity in the velocity formulation. Since these cases are at discrete configurations, the problem does not degrade the utility of our system. An example of this isolated singularity is a surface-torus configuration that has the axis of the torus pointing to the sphere center. Another case is where the radius of curvature is identical at the minimal distance points between the two surfaces. The only case where broad regions of singularity occur is for identical radius of curvatures, most commonly, two flat, parallel surfaces. A simple constraint, such as that given in the polyhedral manipulation work from [253], can solve this degenerate case.

### 10.3.2 Convergence Rates

A number of measures have been introduced in this work to allow multiple, highly nonlinear surface and trimming constraints to be satisfied while maximizing mechanical design objectives. This convergence is an intractable problem in general (note that it is easy to write a system of constraints that cannot be satisfied). However, our monitoring mechanism simplifies the problem by keeping track of global collision events. This simplification allows the local extremal distance contact computations to be maintained, without problems that other numerical methods suffer. The collision management scheme also allows the curve extremal distance to be maintained. For surfaces that do not have tiny, sharp changes as described in our section on “spheres-of-neglect,” the proper extremal distance can be maintained.

### 10.3.3 Class of Mechanisms

The fast surface Jacobian framework has demonstrated that surface constraint evaluation can be done in excess of haptics interactive (force control) rates. From the timing analysis, we can determine a class of mechanisms that are handled by our system in the steady (during-manipulation) state.

Since the optimization iterations run efficiently in the steady state, local case,
we look at what class of mechanisms converge locally. Convergence in our optimization algorithm takes advantage of inter-iteration coherence of the solution. When mechanisms are updated at several hundred times per second, assuming that the mechanisms are initialized with an admissible (assembled) configuration, the optimization characteristics are improved dramatically. For the class of mechanisms that converge within a single iteration given high update rates, our algorithm can accommodate medium and large sized systems. For mechanisms without surface constraints,

\[(50,000 \text{ simple constraint evaluations / second}) \times (1 \text{ mechanism update / } n \text{ constraint evaluations}) = 200 \text{ updates/second}\]

required implies that \(n\) is on the order of 250 constraints. Since the number of constraints (joints) is usually linearly related with the number of bodies in mechanisms that make sense, several hundred bodies with several hundred joints can be handled for simple mechanical constraints. This analysis holds for mechanisms that maintain convergence at 200 updates/second given an initially valid configuration.

For mechanisms with surface constraints, with both initially valid configurations and initially valid minimal distance state, the number of surface constraints and bodies is on the order of 20 (replace 50,000 by 4,000). Therefore, medium and large sized systems with a dozen or two surface contacts are the class of mechanisms allowed with our current implementation and current hardware. Parallelism within the constraint evaluation process is not fully exploited by our optimization implementation, indicating that significant improvement of these results is straightforward, but outside of the research issues in this work.

While the global initialization of the surface kinematics and admissible assembly configuration is a more costly problem, it does not degrade the utility of our system. The global computations are run only at start-up and periodically during manipulation to manage the local computations. The periodic management is also in a separate process. In this way, we have hidden the expensive, semi-interactive operations from the user. Our focus is then on the steady state, local case that can take advantage of interframe coherence with sufficient convergence characteristics.
to be used in a variety of virtual environment and CAD applications.

For the interactive dynamics application, all the minimal distance initialization and update problems are the same as in the optimal surface configuration application, but dynamics computations are the dominant cost rather than optimization. The constraint evaluation is not the dominant cost in this implementation since the cubic cost characteristics of the closed loop dynamics evaluation are the largest term. The dynamics analysis can evaluate any sized system, with only memory and surface minimal distance initialization as limitations. Small and medium sized systems, such as the example given above, can achieve marginally force-interactive rates of 100 Hertz.
CHAPTER 11

DISSERTATION CONCLUSION

The aim of this work was to develop methods for integrating interactive surface constraints with a CAD and virtual prototyping system. We have derived efficient relations that enabled our experimental testbed to run a class of mechanisms at rates sufficient to allow force feedback as well as graphical feedback.

The surface kinematics computations have been generalized, and from the resulting linear equations, an exact surface constraint Jacobian and Hessian has been derived. This work has utilized our fast differential tools for use in Lagrange multiplier surface impact dynamics, interactive model configuration optimization, and for local collision detection updates. Our methods handle a range of real mechanisms well. The conclusion of this work is the methods scale to practical applications that run at interactive rates.

11.1 Future Work

The differential relations can be applied to the interval methods for optimal surface placement because an algebraic solution to our surface Jacobian exists. The interval analog to this solution can be used to compute a Jacobian for use in interval Newton iteration. Because our methods are efficient, the interval version is also likely to be efficient, and is likely to speed up the slow interval methods previously applied to these problems.

Flexible body coordinates can be added to our vector of generalized coordinates to allow deformation in our assembly optimization and surface tracing applications. The derivatives of the surface with respect to the deformation coordinates are required. The derivatives of the control mesh for small deformations can be evaluated as given in [158], making the extension of our methods to flexible coordinates
straightforward. Preliminary experiments into the effect of additional deformation coordinates has already been carried out in our experiments in Matlab with good convergence results.

A great deal of parallelization is available in our constraint framework that could improve performance. The constraints within a dynamics timestep or an optimization iteration can both benefit from parallel evaluation. Further research in parallelizing the optimization matrix operations or dynamics equations of motion is another avenue for increasing algorithm speeds.

The surface-surface floating contact interactions in the optimal surface placement application allow surfaces to slide due to their “floating” characteristics. Because any realistic surface sliding has a great deal of friction, the inverse dynamics and constraint force feedback should incorporate friction forces. There are several avenues to incorporating these forces into our interaction computations:

- **External forces.** We can use Coulomb friction or more advanced stick-slip friction [59] together with friction cone considerations to apply external forces to the mechanical parts undergoing sliding contact. The friction forces would be applied in the tangent plane at the surface contact.
- **Linear complementarity computations.** A great deal of research has been done in the area of force computations at unilateral surface contact points [83, 17]. While surface dynamics between polyhedral objects, restricted to the case where the relative velocity between the objects is small, as in previous works, is not the focus of this dissertation, these techniques could be used to find the frictional contact forces. A hybrid forward / inverse dynamics solution could be employed. However, the positions, velocities, and accelerations are already being computed by the optimization algorithm, making a friction force profile based on object velocity, as in the external forces approach, the best alternative.

The friction forces between surfaces in the optimization application could be reduced to prevent large, confusing forces to be rendered to a user. One reason
that the full amount should be reduced is that the surface contacts in this application are bilateral, while real surface contacts, except for magnetic surfaces, are unilateral.
APPENDIX

ADDITIONAL PROOFS AND DERIVATIONS

A.1 Matrix Relating Differentials: Jacobian Extraction

In Chapter 3, we extract the parametric floating contact Jacobian matrix from the comparison of the equation of the form $\dot{u} = D\dot{q}$ with the chain rule $\dot{u} = \partial u/\partial q \dot{q}$. This comparison holds based on the argument that both equations are true for any sample $\dot{q}$, or a nonsingular matrix of samples $\dot{Q}$, which produce a matrix of results $\dot{U}$. We have

$$\dot{U} = D\dot{Q}$$ \hspace{1cm} (A.1)

and

$$\dot{U} = \frac{\partial u}{\partial q} \dot{Q}.$$ \hspace{1cm} (A.2)

Therefore, we have

$$D\dot{Q} = \frac{\partial u}{\partial q} \dot{Q}.$$ \hspace{1cm} (A.3)

Since we can choose any set of linearly dependent samples $\dot{Q} = \{\dot{q}^k\}_{k=1..n}$, where $n$ is the number of generalized coordinates for 2 surfaces (12 for Euler angle velocities and exponential map velocities, 14 for kinematically valid quaternion velocities), $Q$ is invertible. Therefore, we have

$$D = \frac{\partial u}{\partial q},$$ \hspace{1cm} (A.4)

which shows that the Jacobian is the matrix $D$. 
A.2 Fourth-Order Numerical Second Derivative Formulation

Finite differences for first and second derivatives are sometimes required when an analytical solution is intractable or too expensive. High order methods for doing this for greater accuracy are desirable. Although the focus of this work is on analytic solutions to derivatives, empirical comparisons with the high order numerical means of computing derivatives is often useful.

A typical $O(h^2)$ formula for the second derivative is given by [120],

$$\frac{\partial^2 f}{\partial x^2} \approx D_{2h}(f(x_1)) = \frac{f(x_2) - 2f(x_1) + f(x_0)}{h^2}$$ (A.5)

where $x_j = x_0 + (j - 1)h$ for some step size $h$.

We derive an $O(h^4)$ second derivative formula, the derivation and final result presented below. (To get more stability without more samples or significant computational cost, filtering would be the best way to get a more accurate result for small $h$.)

The way to get higher order accuracy is to use more samples and therefore more information to solve the problem. The samples are often expensive to evaluate. If a fourth order method takes twice as many samples as a second order method, this is a win since more than twice as much accuracy is obtained, assuming $h$ is somewhat small ($<< 1$) and the fourth order method is not greatly more sensitive to noise (so that it is still robust).

Richardson extrapolation can be used to obtain a fourth order ($h^4$) first derivative, and then obtain the second derivative by performing the higher order operation twice, or composing this operator:

$$fp(x) = \phi(cos, x, h/2) + 1/3[\phi(cos, x, h/2) - \phi(x, h)]$$ (A.6)

$$= \frac{4}{3}\phi(cos, x, h/2) - 1/3\phi(cos, x, h)$$ (A.7)

where

$$\phi(f, x, h) = 1/(2h)(f(x + h) - f(x - h))$$ (A.8)
Composing, or performing the operation again, with \( fp \),

\[
f'' = \frac{4}{3} \phi(fp, x, \frac{h}{2}) - 1/3 \phi(fp, x, h) \tag{A.9}
\]

Hypothesis: With the Taylor series terms, the denominator is only \( O(h^2) \) in error magnification, and agrees with the true \( f'' \) up to \( h^4 f''''(\zeta) \).

**Proof**

\[
D_{4h} = \frac{4}{3} \phi([\frac{1}{3} \phi(f, x, \frac{h}{2}) - \frac{1}{3} \phi(f, x, h)], x, \frac{h}{2})
\]

\[
-\frac{1}{9} \phi([\frac{1}{3} \phi(f, x, \frac{h}{2}) - \frac{1}{3} \phi(f, x, h)], x, h)
\]

\[
= \frac{4}{3} \phi([\frac{4}{(6h)}](f(x + \frac{h}{2}) - f(x - \frac{h}{2})) - \frac{1}{6h} (f(x + h) - f(x - h)], x, \frac{h}{2})
\]

\[
-\frac{1}{9} \phi([\frac{4}{(6h)}](f(x + \frac{h}{2}) - f(x - \frac{h}{2})) - \frac{1}{6h} (f(x + h) - f(x - h)], x, h)
\]

\[
= \frac{4}{3} \phi([\frac{4}{(6h)}](f(x + \frac{h}{2}) - f(x - \frac{h}{2})) - \frac{1}{6h} (f(x + h) - f(x - h)], x, \frac{h}{2})
\]

\[
-\frac{1}{9} \phi([\frac{4}{(6h)}](f(x + \frac{h}{2}) - f(x - \frac{h}{2})) - \frac{1}{6h} (f(x + h) - f(x - h)], x, h)
\]

\[
= \frac{4}{3} \phi([\frac{4}{(6h)}](f(x + \frac{h}{2}) - f(x - \frac{h}{2})) - \frac{1}{6h} (f(x + h) - f(x - h)], x, \frac{h}{2})
\]

\[
-\frac{1}{9} \phi([\frac{4}{(6h)}](f(x + \frac{h}{2}) - f(x - \frac{h}{2})) - \frac{1}{6h} (f(x + h) - f(x - h)], x, h)
\]

Continuing,

\[
\left(\frac{4}{3h} (f(x + h) - f(x)) - \frac{1}{6h} (f(x + 3h/2) - f(x - h/2))\right)
\]

\[
-\left(\frac{4}{3h} (f(x) - f(x - h)) - \frac{1}{6h} (f(x + h/2) - f(x - 3h/2))\right)
\]
\[ \frac{-1}{3} \cdot \frac{1}{(2h)} \cdot ( \frac{4}{3h}(f(x + \frac{3h}{2}) - f(x + \frac{h}{2})) - \frac{1}{6h}(f(x + 2h) - f(x))) \\
-(\frac{4}{3h}(f(x - h) - f(x - \frac{3h}{2})) - \frac{1}{6h}(f(x) - f(x - 2h)))) \\
= \frac{4}{3h}( \frac{4}{3h}(f(x + h) - f(x)) - \frac{1}{6h}(f(x + \frac{3h}{2}) - f(x + \frac{h}{2}))) \\
-(\frac{4}{3h}(f(x) - f(x - h)) - \frac{1}{6h}(f(x + \frac{h}{2}) - f(x - \frac{3h}{2})))) \\
= \frac{1}{(h^2)} \cdot [1/36f(x + 2h) + 1/36f(x - 2h) \\
-4/9f(x + \frac{3h}{2}) - 4/9f(x - \frac{3h}{2}) \\
+16/9f(x + h) + 16/9f(x - h) \\
+(-2 \cdot 16/9 - 2 \cdot 1/36)f(x) \\
+4/9f(x + \frac{h}{2}) + 4/9f(x - \frac{h}{2})]. \\
\]

The denominator is \( O(h^2) \); we are no worse off on the noise magnification issue.

To find the agreement with \( f'' \), that is, the order of accuracy of this method, we substitute

\[
\begin{align*}
  f(x + 2h) & = f(x) + 2hf' + 1/2 \cdot 4h^2f'' + 8/6h^3f''' + 16/24h^4f'''' + ... \\
  f(x - 2h) & = f(x) - 2hf' + 1/2 \cdot 4h^2f'' - 8/6h^3f''' + 16/24h^4f'''' + ... \\
  f(x + \frac{3h}{2}) & = f(x) + \frac{3h}{2}f'(x) + 9/8h^2f''(x) + 27/48h^3f'''(x) + 81/(24 \cdot 16)h^4f'''' \\
  f(x - \frac{3h}{2}) & = f(x) - \frac{3h}{2}f'(x) + 9/8h^2f''(x) - 27/48h^3f'''(x) + 81/(24 \cdot 16)h^4f'''' \\
  f(x + h) & = f(x) + hf'(x) + 1/2h^2f''(x) + 1/6h^3f''' + 1/24h^4f'''' + ... \\
\end{align*}
\]
\[
f(x - h) = f(x) - hf'(x) + 1/2h^2f''(x) - 1/6h^3f''' + 1/24h^4f'''' + ...
\]
\[
f(x + \frac{h}{2}) = f(x) + \frac{h}{2}f'(x) + 1/8h^2f''(x) + 1/48h^3f''' + 1/384h^4 + ...
\]
\[
f(x - \frac{h}{2}) = f(x) - \frac{h}{2}f'(x) + 1/8h^2f''(x) - 1/48h^3f''' + 1/384h^4 + ...
\]

We have

\[
D_{4h} = \frac{1}{h^2} [ \\
1/36[f(x) + 2hf' + 1/2 * 4h^2f'' + 8/6h^3f''' + 16/24h^4f'''' + ... \\
+1/36[f(x) - 2hf' + 1/2 * 4h^2f'' - 8/6h^3f''' + 16/24h^4f'''' + ... \\
-4/9[f(x) + \frac{3h}{2}f'(x) + 9/8h^2f''(x) + 27/48h^3f''' + 81/(24 * 16)h^4f'''' \\
-4/9[f(x) - \frac{3h}{2}f'(x) + 9/8h^2f''(x) - 27/48h^3f''' + 81/(24 * 16)h^4f'''' \\
+16/9[f(x) + hf'(x) + 1/2h^2f''(x) + 1/6h^3f''' + 1/24h^4f'''' + ... \\
+16/9[f(x) - hf'(x) + 1/2h^2f''(x) - 1/6h^3f''' + 1/24h^4f'''' + ... \\
+(-2 * 16/9 - 2 * 1/36)f(x) \\
+4/9[f(x) + \frac{h}{2}f'(x) + 1/8h^2f''(x) + 1/48h^3f''' + 1/384h^4 + ... \\
+4/9[f(x) - \frac{h}{2}f'(x) + 1/8h^2f''(x) - 1/48h^3f''' + 1/384h^4 + ... \\
] \\
= \frac{1}{h^2} [ \\
1/36[2hf' + 1/2 * 4h^2f'' + 8/6h^3f''' + 16/24h^4f'''' + ... \\
+1/36[-2hf' + 1/2 * 4h^2f'' - 8/6h^3f''' + 16/24h^4f'''' + ... \\
-4/9[\frac{3h}{2}f'(x) + 9/8h^2f''(x) + 27/48h^3f''' + 81/(24 * 16)h^4f'''' \\
-4/9[-\frac{3h}{2}f'(x) + 9/8h^2f''(x) - 27/48h^3f''' + 81/(24 * 16)h^4f'''' \\
+16/9[hf'(x) + 1/2h^2f''(x) + 1/6h^3f''' + 1/24h^4f'''' + ... \\
+16/9[-hf'(x) + 1/2h^2f''(x) - 1/6h^3f''' + 1/24h^4f'''' + ... 
\]
}
\[ +\frac{4}{9} \left( \frac{h}{2} \right) f'(x) + \frac{1}{8} h^2 f''(x) + \frac{1}{48} h^3 f''' + \frac{1}{384} h^4 + \ldots \]

\[ +\frac{4}{9} \left( -\frac{h}{2} \right) f'(x) + \frac{1}{8} h^2 f''(x) - \frac{1}{48} h^3 f''' + \frac{1}{384} h^4 + \ldots \]

Now some easy simplifications present themselves,

\[ = \frac{1}{h^2} \left[ \frac{1}{36} \left( \frac{1}{2} \right) 4 h^3 f'' + 16/24 h^4 f''' + \ldots + h^6 O(h^6) \right. \]

\[ + \frac{1}{36} \left( \frac{1}{2} \right) 4 h^3 f'' + 16/24 h^4 f''' + \ldots + h^6 O(h^6) \]

\[ - \frac{4}{9} \left( \frac{9}{8} h^2 f''(x) + 81/(24 \times 16) h^4 f''' + \ldots + h^6 O(h^6) \right) \]

\[ - \frac{4}{9} \left( \frac{9}{8} h^2 f''(x) + 81/(24 \times 16) h^4 f''' + \ldots + h^6 O(h^6) \right) \]

\[ + 16/9 \left( \frac{1}{2} h^2 f''(x) + 1/24 h^4 f''' + \ldots + h^6 O(h^6) \right) \]

\[ + 16/9 \left( \frac{1}{2} h^2 f''(x) + 1/24 h^4 f''' + \ldots + h^6 O(h^6) \right) \]

\[ + \frac{4}{9} \left( \frac{1}{8} h^2 f''(x) + 1/384 h^4 + \ldots + h^6 O(h^6) \right) \]

\[ + \frac{4}{9} \left( \frac{1}{8} h^2 f''(x) + 1/384 h^4 + \ldots + h^6 O(h^6) \right) \]

Since

\[ 1 = 1/36 \times \frac{4}{2} + 1/36 \times \frac{4}{2} - 4/9 \times \frac{9}{8} - 4/9 \times \frac{9}{8} + 16/9 \times \frac{1}{2} + 16/9 \times \frac{1}{2} + 4/9 \times \frac{1}{2} + 4/9 \times \frac{1}{2}, \]

and

\[ 0 == 2 \times 1/36 \times 16/24 - 2 \times 4/9 \times 81/(24 \times 16) + 2 \times 16/9 \times 1/24 + 2 \times 4/9 \times 1/384, \]

we are left with

\[ D_{4h} = \frac{1}{(h^2)} (h^2 f'' + O(h^6)) f'''. \quad \text{(A.10)} \]

\( D_{4h} \) is now \( f'' \) up to a term \( O(h^6/h^2) = O(h^4) \).

\textbf{End of Proof.}
The error therefore has been established to be at least on the order of $O(h^4)$. Since more precision is not seen in the original fourth-order differencing method, we stop writing algebra for cancellation of possible higher terms. This is better than $O(h^2)$ from the standard 2nd derivative central finite differences.

It is probable that even higher order Richardson extrapolation can be used, since higher order first derivatives can be extracted, order without limit. Applying the very high order first derivative formula again, as we did here, should give very high order 2nd derivatives. The number of samples approximately doubles at each Richardson application, so the cost is not worthwhile for more than $h^4$ or $h^8$ methods for our engineering applications.

### A.3 Simple Joint Constraint

We derive the constraint equation for a spherical and universal joint for reference here. The universal joint included here incorporates a spherical joint constraint for the first three constraints and a perpendicular constraint for the fourth equation.

The universal joint features of the two bodies must meet (which is the definition of a spherical joint), and is given by

$$C^{\text{sph}}(q) = \left[ R(q^i_{\text{rot}})k_j + q_{i,\text{disp}} - R(q^j_{\text{rot}})k_j - q^j_{i,\text{tr}} \right]_{3 \times 1}$$  \hspace{1cm} (A.11)

$R$ is a rotation matrix, $q^i_{i,\text{tr}}$ is a translation vector, $q^i_{\text{rot}}$ are the rotation coordinates, and $k$ is a constant local body frame vector indicating the position of the universal joint feature, as shown in Fig. A.1.

![Figure A.1](image-url)  

**Figure A.1.** A universal joint is a spherical joint that can rotate in any way as long as both axes are perpendicular.
The perpendicular condition enforces the constraint that the local body axes, $\vec{h}_i$, $\vec{h}_j$ of each piece remain at 90 degrees, and the resulting universal joint removes four degrees of freedom.

$$C_{\text{perp}}(q) = \left[(R_i \vec{h}_i)^T (R_j \vec{h}_j)\right]_{1x1}.$$  \hspace{1cm} (A.12)

$$C_{\text{univ}}(q) = \begin{bmatrix} C_{\text{sph}} \end{bmatrix}_{4x1}.$$  \hspace{1cm} (A.13)

### A.4 Quaternion Differential Algebra

The partial derivative of the rotation of constant vectors with respect to the generalized orientation coordinates is used in the derivatives of the surface constraint and the simple joint constraints. The local surface control mesh is rotated by the generalized coordinates, as are local vectors defining spherical, universal, and other joints. The rotation operator with quaternion arguments $q_{\text{rot}}$ rotates a constant (i.e., local frame) vector $\mathbf{v}$ using quaternion multiplication, given by:

$$q_{\text{rot}} \mathbf{v} q_{\text{rot}}^* = (q_0^2 - q_{\text{rot}} \cdot q_{\text{rot}}) \mathbf{v} + 2q_0 q_{\text{rot}} \times \mathbf{v} + 2q_{\text{rot}} (q_{\text{rot}} \cdot \mathbf{v})$$

$q_{\text{rot}}^* = q_0 - q_{\text{rot}}$ is the inverse of the quaternion.

The important Jacobian operation of the expression $q_{\text{rot}} \mathbf{v} q_{\text{rot}}^*$ with respect to the rotation $q_{\text{rot}}$ used in simple joints and more complicated surface constraints is now given by

$$\frac{\partial(q_{\text{rot}} \mathbf{v} q_{\text{rot}}^*)}{\partial q_{\text{rot}}} = \begin{bmatrix}
q_y v_z - q_z v_y & q_x v_x + q_y v_y + q_z v_z \\
q_0 v_y + q_z v_x - q_x v_z & -q_z v_y - q_0 v_z + q_y v_x \\
q_0 v_z + q_x v_y - q_y v_x & q_0 v_y + q_z v_x - q_x v_z \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
\end{bmatrix}.$$  \hspace{1cm} (A.14)

The term $(C_q \dot{q})_q$ is the time derivative of the Jacobians of the simple joint constraints. It is required in Eqn. 9.1. The derivative of the surface constraints
is not required, although we could compute it in closed form from the differential surface kinematics formulation. The time derivative of Jacobian of the rotation matrix times a constant vector, $R\pi$, can be computed symbolically from

$$\frac{d}{dt}(R\pi)_q = (R\pi_q\dot{q})_q.$$  \hspace{1cm} (A.15)
REFERENCES


[71] Omelyan, I., “On the Numerical Integration of Motion for Rigid Polyatomics:


[83] Laursen, T., Govindjee, S., “Note on the Treatment of Frictionless Contact Between Non-Smooth Surfaces in Fully Non-Linear Problems,” in *Communications in Numerical Methods in Engineering*, vol. 10, no. 11, Nov. 1994, John Wiley & Sons Ltd.


[140] Fang, Y., Hsieh, C., Kim, M., Changh, J., Woo, T., “Real-time Motion


Nelson, E., Cohen, E., “User Interaction with CAD Models with Nonholo-


Hollerbach, J., Calibration Class Notes, University of Utah, 1997.


Tolani, D., Badler, N., “Real-Time Inverse Kinematics of the Human Arm,”
in *Presence*, vol. 5, no. 4, Fall 1996, pp. 393-401.


