Accepted for final publication, January 2018

6-DoF Haptic Rendering of Static Coulomb Friction Using Linear Programming

Danyong Zhao, *IEEE Member,* Yijing Li, *IEEE Member,* and Jernej Barbič, *IEEE Senior Member*

Abstract—Simulating frictional contact between objects with complex geometry is important for 6-DoF haptic rendering applications. For example, friction determines whether components can be navigated past narrow clearances in virtual assembly. State-of-the-art haptic rendering of frictional contact either augments penalty contact with frictional penalty springs, which do not prevent sliding and cannot render correct static friction, or uses constraint-based methods that have difficulties in meeting the stringent haptic loop computation time requirements for complex geometry. We give a 6-DoF Coulomb friction haptic rendering algorithm for distributed contact between rigid objects with complex geometry. Our algorithm is compatible with the fast point vs implicit function penalty-based contact method such as the Voxmap-PointShell method. Our algorithm incorporates the maximal dissipation principle and produces correct static friction, all the while inheriting the speed of penalty-based methods. We demonstrate our algorithm on several challenging 6-DoF haptic rendering examples.

Index Terms—haptics, penalty contact, friction, feasibility test

1 Introduction

VIRTUAL assembly and simulation of real-world engineering tasks using haptics is challenging due to the stability, accuracy and fast computation requirements. A key bottleneck in realistic haptic rendering of complex geometry is accurate and efficient collision modeling. Recent methods in haptics have achieved stable haptic rendering with complex collision environments. Among them, penalty contact forces [1], [2], [3], [4] and static virtual coupling [5] are two commonly used techniques that can provide fast computation with acceptable physical realism. However, Coulomb friction whereby objects do not slip unless friction cone thresholds are exceeded has remained too computationally intensive for complex geometry. Friction is an important component of the haptic experience and plays a crucial role in determining whether manufactured parts can be assembled successfully or not.

Two common approaches to modeling distributed contact are the penalty-based and constraint-based methods. Distributed contact here refers to general, complex contact between two rigid objects with complex geometry, varying spatially and temporally, and usually consisting of several collision sites with varying sizes and orientations. The penalty method computes collision forces based on penetration depths or volumes, whereas the constraint method applies equality or inequality constraints to remove penetrations, usually performed by solving linear complementarity problems (LCP). Constrained-based modeling gives correct contact results, but is generally slow for complex geometry. The penalty method is easy to implement and scales well with complex scenes at haptic rates, at the cost of reduced accuracy. The majority of the friction literature adopts Coulomb's law to model the complex physical frictional interaction. One popular approach extends the normal penalty contact forces by adding anchors and penalty friction springs [6]. It inherits the simplicity and speed of penalty-based methods, but since it uses a spring to model static friction, correct static friction cones cannot be simulated; objects will slide even under very small forces.

In contact mechanics literature, it is generally accepted that an exact approach for friction is to model both the normal contact and friction as constraints, resulting in a nonlinear complementarity problem (NCP). Although the approximation of the friction cone using inscribed friction polygons [7] reduces the complexity to a LCP, the constraint framework is generally computationally expensive for complex geometry, limiting its distributed contact haptic applications. So, the question that we aim to answer in our paper is, "How do we simplify/approximate exact friction methods, so that we still obtain plausible static Coulomb friction, and are able to scale to haptic rates with complex geometry?" Given the success of frictionless penalty-based methods for complex geometry at haptic rates such as the VPS method [1], [2] and general point-vs-implicit function 6-DoF haptic rendering [3], [4], it is natural to wonder if proper Coulomb friction can be somehow added to such methods. In this work, we demonstrate such an approach. In order to accelerate LCP methods, one has to somehow remove the complementarity. For complementarity of the normal contact force and normal velocity, we do this by using the penalty forces (call this ingredient A). For complementarity between friction forces and tangential velocities, we achieve this using our linear programming approach (call this ingredient B). We would like to emphasize that both A and B are necessary in order to avoid the complementarity nature of the optimization problem to solve at each timestep. While we do not claim to be as accurate as the friction methods of Stewart and Trinkle [8], Kaufman [9] or Duriez [10], our method is an improvement over friction methods that were previously available for haptic rates with complex geometry (Yamane's method [6]).

We give a new friction algorithm that can compute correct static Coulomb friction and that runs at haptic rates even for complex geometry and more than one hundred contacts. Our algorithm uses the penalty-based method for normal contact forces and can be easily added to standard state-of-the-art penalty-based

• *All authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, 90089.*
E-mail: danyongz@usc.edu, yijingl@usc.edu, jnb@usc.edu.

methods for 6-DoF haptic rendering of complex geometry, such as the VPS method [1], [2], [3], [4]. Because the manipulated object dynamics is not of primary concern in virtual assembly, we use static virtual coupling [5]. Static virtual coupling does not model rigid body dynamics. It models the velocity as always being zero, and always places the object in static equilibrium under the virtual coupling, normal contact and frictional contact forces.

Our work demonstrates how to combine the rapid penalty-based contact forces with Coulomb frictional inequality constraints and the maximal dissipation principle. At first glance, Coulomb friction seems incompatible with both the penalty method and fast computation, as it is not obvious how to combine penalty forces in any meaningful way with constraint-based contact, and because Coulomb friction leads to quadratic optimization problems (QP) or linear complementarity problems (LCPs) that are slow to solve for complex geometry. We demonstrate how to use penalty forces as normal contact forces in a stable way (thereby removing them from optimization, boosting speed) and how to replace the quadratic static friction optimization program for a *linear program* (LP). The linear program is much faster to solve, which enables good scaling of our method under an increasing number of contacts. Our algorithm runs faster than 1 msec with about 100 contact points (Figure 1), which is about 50× faster than the state-of-the-art constraint-based friction approaches (Figure 15). Because we can achieve haptic rates (1,000 Hz) directly, we do not need to use multirate simulation. We prove that for static friction, the solution to our linear program is equivalent to the original quadratic program. We are not aware of this observation being made previously anywhere in the contact literature. Using Coulomb static friction, we augment virtual assembly with the ability to discover that certain insertion paths are impossible under high friction, but possible with small (or zero) friction. We experimentally demonstrate that such effects cannot be achieved with Yamane's penalty-based friction [6]. When our linear program is infeasible, this provably means that the static friction is too weak to keep the object motionless. In such configurations, we then use static damping [11] to simulate dynamic friction-like effects and improve haptic simulation stability.

We do not model dynamic friction in our work. Our goal is to provide plausible static Coulomb friction for complex geometry at haptic rates. Constraint-based methods do not run at haptic rates for complex geometry. The existing state of the art for friction for complex geometry at haptic rates is the Yamane's method [6]. Although Yamane's friction is very fast, it has a substantial limitation: the objects do not stay still, but slide already under arbitrarily small forces. Our work improves upon this: in our work, if the object is undergoing static friction at all contacts, it will not slide. We note that we define the precise meaning of undergoing static friction at all contacts in Section 4; and this definition matches the familiar expected intuition. However, if the friction coefficient(s) at some contact(s) is/are too low for static friction to occur at all contacts, then in the real-world, the object will slide at those contacts. This is where our method differs from the ground truth: in our method, we slide at all contacts, not just the ones where the friction was inadequate. However, this is still a substantial improvement over the Yamane's method, which always slides, even under tiny forces when the object is undergoing static friction. In Section 5.6, we perform an extensive comparison to Yamane's method. We observe that there is a substantial difference in allowing penalty-based compliance in both normal and tangential directions (as in Yamane's work [6]), versus allow it only in the normal direction (as in our work). In the real world, normal contact forces originate from normal compliance, namely deformations of the contact site in the normal contact direction. In contrast, friction forces do not originate from any such tangential compliance or tangential deformation. Instead, they originate from many micro-contacts on the serrated contact site geometry, which collectively add to completely block the motion, without any deformations, unless the friction cone constraints are exceeded. Therefore, there is a substantial difference in the nature of normal and tangential compliances, which warrants a different treatment for the two directions.

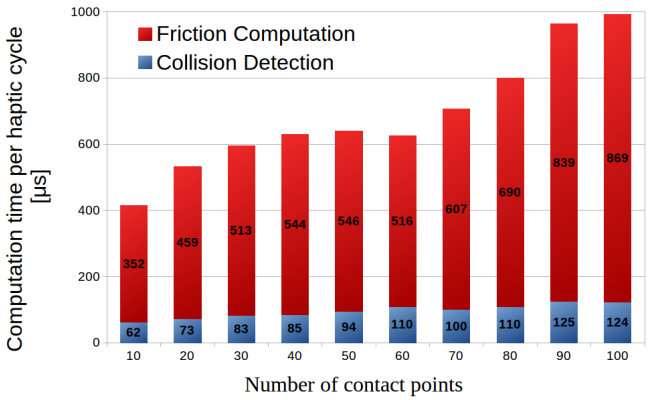


Fig. 1: **Scalability of our friction algorithm.** The total time spent on collision detection and static Coulomb friction computation is under than 1 msec for 100 contact points. This figure corresponds to the cube ($256 \times 256 \times 256$ signed distance field) vs 90-degree corner (7,772 points, 8-level nested point shell) example (Section 5.4).

2 Related Work

A survey of haptic rendering can be found in [12] and we also refer the reader to [13] for a good introduction of several commonly used haptic rendering algorithms. Many techniques exist for efficient haptic rendering of contact. To achieve fast collision detection, McNeely [1] introduced the Voxmap PointShell method (VPS), followed by several VPS improvements [14], [15]. Barbič and James [3] proposed using signed distance fields and nested point trees for improved collision quality and speed. To lessen the problems caused by too many contacts, contact clustering [16], [17], [18], [19] and adaptive contact stiffness scaling [4] can be applied, making the contact more tractable and robust. Virtual coupling [2], [5] is often used to stabilize haptic rendering and reduce penetration. Direct force rendering is also used in some publications [16], [20]. Most of the 6-DoF haptic rendering methods apply to contact between rigid objects, whereas some work studied deformable objects [10]. Otaduy [21] simulated collisions between a rigid tool and a deformable environment. Barbič and James [3] and Kaufman [9] simulated deformable objects in the model-reduced space. Tournier [22] developed a stable method to allow some compliance for constraint-based contact.

The penalty method has a long history in physically based simulation [23]. Due to their simplicity, penalty forces [2], [3], [4] are commonly used to resolve collisions at haptic rates. Several recent publications [24], [25] on penalty forces stabilized their collision response. Yamane [6] demonstrated how to add friction

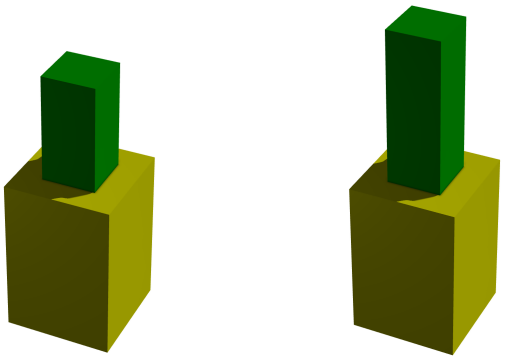
to penalty-based forces, by modeling friction as penalty forces as well. After normal penalty forces are computed, Yamane's friction forces are generated from virtual springs for static friction, or according to Coulomb's law under dynamic friction. Yamane's method is very fast, but cannot model correct Coulomb static friction: virtual frictional springs cannot keep the object static under static friction because the object must move a bit to stretch the virtual friction spring so that a friction force can appear. Constraint-based methods are more involved. Based on Coulomb's friction law, Baraff [26] developed a rigid body contact and friction algorithm that is based on the contact point accelerations. Mirtich [27] observed that the previous acceleration-level formulations do not always lead to a unique solution and proposed an impulse-based method to address this. Impulse-based methods have been widely used [28], [29]. When no friction is modeled, the constraint problem can be stated as an LCP. It can be extended to model friction via the Coulomb's friction law. This can be performed at the position or velocity level [30], as done in the popular Stewart and Trinkle velocity-based formulation familiar to the audience in interactive simulation [8] (see also [31]). The friction cone approximation is used to linearize the constraints, at the cost of introducing additional degrees of freedom to the LCP. We note that LCPs are in general not equivalent to linear programs (LP). The LCP-LP equivalence is known for a special class of matrices, called "Z-matrices" [32], but the LCP for a friction problem does not lead to "Z-matrices" and hence it cannot be solved using such techniques. Duriez [10] proposed an iterative Gauss-Seidel-like algorithm to solve the nonlinear friction constraints directly. Because the Signorini's friction law is not perfect and may give indeterminate solutions, Kaufman [9] and Drumwright [33] used the principle of maximum dissipation to compute a well-defined and unique friction. They cast the friction problem as a quadratic programming (QP) problem. Kaufman solved it via staggered projections and a convex QP solver, using the QL library [34]. Constraint-based methods are usually slower than penalty-based methods, so a multirate framework can be used to allow the contact force computation to run at a lower rate than the haptic computation [21], [35]. Multirate simulation comes at a cost of decreased haptic fidelity. Different from all these prior methods, we do not need to use multirate simulation. We utilize the penalty method for normal contact forces, but use a constraint method for static friction. Our novel combination greatly accelerates static Coulomb friction.

3 Background: Penalty-Based Contact and Static Virtual Coupling

Our penalty contact model uses a nested point shell and a signed distance field to perform collision detection between two rigid bodies [3]. During collision detection, we check points from the point shell against the distance field. When the distance value of a point is negative, we label it in contact and apply a penalty normal force and torque to it,

$$F_n = -k_n d N, \quad \tau_n = r \times F_n, \tag{1}$$

where k_n is the normal contact stiffness, $d < 0$ is the penetration depth, and $N, r \in \mathbb{R}^3$ are the contact normal and the torque handle. The penetration depth d is obtained from a precomputed signed distance field [36]. We use precomputed (rigidly rotated with the object) contact normals on the point shell. We utilize the adaptive stiffness method [4], which improves stability of contact rendering. The penalty model requires less time to resolve contact than constraint-based methods. The gradients of the normal contact force and torque have been given in [25],



(a) Using penalty forces (b) Using constraint forces

Fig. 2: **Flexibility and tolerance of penalty-based contact vs constraint-based contact.** When the peg size is slightly larger (1%) than the hole size, the constraint-based method prevents the peg from entering into the hole, due to the strict non-deformable and non-penetration property. However, much like in many real structures, the penalty method is able to simulate the assembly process by modeling a small (adjustable) compliance.

$$\frac{\partial F_n}{\partial x} = -k_n N N^T, \qquad \frac{\partial F_n}{\partial \omega} = -\frac{\partial F_n}{\partial x} \cdot [r]_\times, \tag{2}$$

$$\frac{\partial \tau_n}{\partial x} = (\frac{\partial F_n}{\partial \omega})^T + [F_n]_\times, \qquad \frac{\partial \tau_n}{\partial \omega} = [r]_\times \cdot \frac{\partial F_n}{\partial \omega}, \tag{3}$$

where $[r]_\times$ represents a skew-symmetric matrix

$$[r]_\times = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}. \tag{4}$$

We note that in constraint-based methods, friction does not affect the insertability of the object into complex geometry. In other words, for constraint-based methods, the object is either insertable for all coefficients of friction (including infinite), or not insertable at all. This is because the constraint-based methods do not permit any penetration. Suppose that an insertion with a constraint-based method is possible, but involves contact (call it "original simulation"). Then, the user can always perform exactly the same insertion trajectory with zero contact forces and torques, by navigating the manipulandum so that the change in the virtual coupling forces and torques relative to the original simulation equals the contact forces and torques in the original simulation. Therefore, the value of the friction coefficient does not change insertability with constraint-based methods. For both position-based and velocity-based rigid body constraint-based methods, the insertion simulations are hampered because the tiny contact deformations cannot be mimicked by non-penetrable constraints. In contrast, our penalty-model can simulate a certain level of compliance, mimicking the real-world effect that objects are not rigid in contact, but bend and dent somewhat to accommodate insertion. We know from practical every day experience that friction coefficient matters for insertability into tight spaces. Our penalty-based approach gives the simulation such compliance (Figure 2), and makes insertability dependent on the friction coefficient.

We now briefly explain static virtual coupling, which we adopt because it decreases penetration and stabilizes simulation. There are two copies of the manipulated object, the manipulandum object and the simulation object. By moving the arm of a haptic device, the user can move and rotate the manipulandum object. Rigid body simulation is performed on the simulation object, and collision detection is performed between the simulation object and the rigid environment. Three types of forces are applied to the simulation object: the virtual coupling force F_{vc}, the normal contact force F_n and the friction force F_f. Accordingly, there are three types of torques, the virtual coupling torque τ_{vc}, the normal contact torque τ_n and the friction torque τ_f. We use impedance control in this work: at each haptic cycle, we read positions and orientations from the device, and calculate and set the forces and torques. The position and orientation of the manipulandum object are read from the haptic device, and then the virtual coupling force F_{vc} and torque τ_{vc} are computed by

$$F_{vc} = k_{vc}(x_m - x_s), \tag{5}$$

$$\tau_{vc} = k_{vc,tor}(\omega_m - \omega_s), \tag{6}$$

where x_m and x_s are the translations of the manipulandum and simulation object, respectively, and equivalently for ω_m and ω_s for rotations. Parameters k_{vc} and $k_{vc,tor}$ are the virtual coupling force and torque stiffness. One then performs collision detection to calculate the normal contact forces and torques, and their gradients. The standard way of using static virtual coupling [3], [5] is to then compute the change of translation Δx and rotation $\Delta\omega$ of the simulation object, by solving the 6×6 linear system of equations,

$$F + \frac{\partial F}{\partial x}\Delta x + \frac{\partial F}{\partial \omega}\Delta\omega + F_f = 0, \tag{7}$$

$$\tau + \frac{\partial \tau}{\partial x}\Delta x + \frac{\partial \tau}{\partial \omega}\Delta\omega + \tau_f = 0, \tag{8}$$

$$F = f_{vc} + \sum_i F_n^i, \qquad \tau = \tau_{vc} + \sum_i \tau_n^i. \tag{9}$$

Force F is the sum of the virtual coupling force and all the normal contact forces gathered at all contact points, and equivalently for the torque τ. Finally, one completes the haptic cycle by updating the position and the rotation of the simulation object by $x = x + \Delta x, \omega = \omega + \Delta\omega$. The above 6×6 linear system of Equations 7-8 postulates that the friction forces F_f and torques T_f are known. Prior to our work, this has typically been done by computing them using Yamane's penalty-based friction method [6]. In our work, we do not solve the 6×6 linear system of Equations 7-8. Instead, we modify the static friction formulation to support Coulomb static friction (Section 4). We do so by defining and solving a linear program that is a generalization of the linear system of Equations 7-8, and that becomes equivalent to Equations 7-8 when the friction coefficient is zero. Our linear program simultaneously solves for the friction forces and the change of translation Δx and rotation $\Delta\omega$.

4 Static Friction via Linear Programming

For static friction, Coulomb's law states

$$F_f \perp F_n, \tag{10}$$

$$\|F_f\|_2 \le \mu\|F_n\|_2, \tag{11}$$

where F_n and F_f are the normal contact force and friction force, respectively, $\mu \ge 0$ is the friction coefficient, and $\|\cdot\|_2$ is the L^2-norm. We linearize the friction inequality constraint (Equation 11) by approximating the friction cone with a ℓ-sided pyramid (Figure 3), for some integer $\ell \ge 3$,

$$F_f = \beta_u T_u + \beta_v T_v = T\beta, \tag{12}$$

$$\beta_u \cos\theta_j + \beta_v \sin\theta_j \le \mu\|F_n\|, \ j = 0,1,\ldots,\ell-1, \tag{13}$$

where $\theta_j = (2j+1)\pi/\ell$, vectors $T_u, T_v \in \mathbb{R}^3$ are the two tangential directions at contact, $T = (T_u \ T_v) \in \mathbb{R}^{3\times 2}$ and $\beta = (\beta_u \ \beta_v)^T \in \mathbb{R}^2$. Equation 13 requires the projection of F_f on direction $(\cos\theta_j, \sin\theta_j)$ to be smaller or equal to $\mu\|F_n\|$. Different from previous friction constraints which are written using ℓ unknowns (impulse magnitudes), ℓ inequalities that constrain the impulse magnitudes to be non-negative and one "general" inequality constraint (the friction cone constraint) [9], we use two unknowns β_u, β_v (friction force components on two tangential directions) and ℓ "general" inequality constraints. We note that, geometrically in terms of permitted friction forces, our discretized friction cone is the same as in [9], scaled by the constant factor $\cos\theta_0$, which we prove in the theorem below. Note that as $\ell \to \infty$, $\cos\theta_0 \to 1$, i.e., the two friction cones converge to each other. Compared to [9], our friction cone formulation decreases the number of unknowns, at the expense of making the inequalities general linear inequalities (i.e., of the form $\sum a_i x_i \ge 0$), as opposed to coordinate inequalities (i.e., of the form $x_i \ge 0$). In our work, this results in faster solves because the computation time of our linear programming problem is more sensitive to the number of unknowns than the number of constraints.

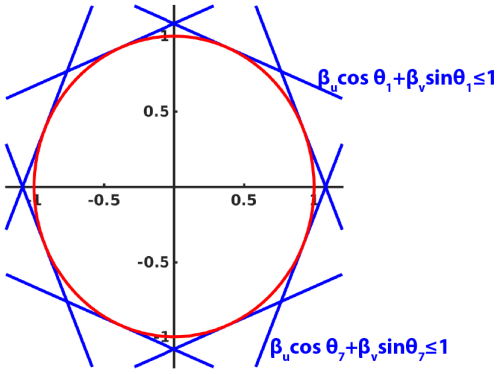


Fig. 3: **8-sided ($\ell = 8$) friction pyramid.** The red circle represents the original friction cone of exact Coulomb friction, whereas the 8-sided polygon bounded by the 8 blue straight lines represents the approximated friction cone. We also label the linear equations on two of the straight lines.

Theorem: Our approximate friction cone defined by Equations 12-13 is identical to the friction cone of [9], geometrically scaled by $\cos\theta_0$, i.e., defined by the following equations,

$$F_f = \sum_{i=0}^{\ell-1} \beta_i T_i, \tag{14}$$

$$T_i = (\cos\varphi_i)T_u + (\sin\varphi_i)T_v, \ i = 0,1,\ldots,\ell-1, \tag{15}$$

$$\varphi_i = 2\pi i/\ell, \tag{16}$$

$$\sum_{i=0}^{\ell-1} \beta_i \le \mu\|F_n\|/\cos\theta_0, \tag{17}$$

$$\beta_i \ge 0, \ i = 0,1,\ldots,\ell-1. \tag{18}$$

Proof: Suppose F_f satisfies Equations 12-13. Because T_i are sorted counter-clockwise, there exists $i \in [0,\ell)$ such that F_f "lies between" consecutive vectors T_i and $T_{(i+1)\%\ell}$, defined mathematically as

$$(F_f \times T_i)\cdot N \leq 0 \leq (F_f \times T_{(i+1)\%\ell})\cdot N, \tag{19}$$

where $\times$ and $\cdot$ denote cross and dot products, respectively, and $\%$ denotes the integer remainder, and N is the contact normal. We now prove that the following solution satisfies Equations 14-18,

$$\beta_i = (F_f \times T_{(i+1)\%\ell})\cdot N/\sin\varphi_1 \tag{20}$$
$$\beta_{(i+1)\%\ell} = -(F_f \times T_i)\cdot N/\sin\varphi_1, \tag{21}$$
$$\beta_j = 0, \text{for } j \notin \{i, (i+1)\%\ell\}, \tag{22}$$

where i comes from Equation 19. It is easy to verify that β_i satisfy Equations 14-15. Because of Equation 19, β_i also satisfy Equation 18. Finally, we verify Equation 17:

$$\begin{aligned}
&\sum_{i=0}^{\ell-1}\beta_i = \beta_i + \beta_{(i+1)\%\ell} = F_f \times (T_{(i+1)\%\ell} - T_i)\cdot N/\sin\varphi_1 \\
&= F_f \times \Big((\cos\varphi_{i+1} - \cos\varphi_i)T_u + (\sin\varphi_{i+1} - \sin\varphi_i)T_v\Big)\cdot N/\sin\varphi_1 \\
&= (\beta_u T_u + \beta_v T_v) \times (-2\sin\theta_i \sin\theta_0 T_u + 2\cos\theta_i \sin\theta_0 T_v)\cdot N/\sin\varphi_1 \\
&= (2\cos\theta_i \sin\theta_0 \beta_u N + 2\sin\theta_i \sin\theta_0 \beta_v N)\cdot N/\sin\varphi_1 \\
&= 2\sin\theta_0(\cos\theta_i\beta_u + \sin\theta_i\beta_v)/(2\cos\theta_0\sin\theta_0) \\
&\leq \mu\|F_n\|/\cos\theta_0.
\end{aligned} \tag{23}$$

Conversely, for any F_f that satisfies Equations 14-18, for any $j \in [0,\ell)$, we have

$$F_f = \sum_{i=0}^{\ell-1}\beta_i T_i = \sum_{i=0}^{\ell-1}\beta_i\cos\varphi_i T_u + \sum_{i=0}^{\ell-1}\beta_i\sin\varphi_i T_v, \tag{24}$$
$$\beta_u = \sum_{i=0}^{\ell-1}\beta_i\cos\varphi_i, \quad \beta_v = \sum_{i=0}^{\ell-1}\beta_i\sin\varphi_i, \tag{25}$$
$$\begin{aligned}
\mu\|F_n\| &\geq \sum_{i=0}^{\ell-1}\beta_i\cos\theta_0 \geq \sum_{i=0}^{\ell-1}\beta_i\cos\theta_{j-i} \\
&= \sum_{i=0}^{\ell-1}\beta_i(\cos\varphi_i\cos\theta_j + \sin\varphi_i\sin\theta_j) \\
&= \beta_u\cos\theta_j + \beta_v\sin\theta_i.
\end{aligned} \tag{26}$$

Therefore, F_f also satisfies Equations 12-13. ■

The maximal dissipation principle [37] states that friction maximizes the rate of negative work of all forces, or, equivalently, minimizes the kinetic energy at the end of the timestep,

$$E_{kinetic} = \frac{1}{2\Delta t^2}(\Delta x^T M \Delta x + \Delta\omega^T I \Delta\omega), \tag{27}$$

where $M, I \in \mathbb{R}^{3\times3}$ are the mass matrix and inertia tensor. Because our goal is to augment penalty-based methods with static Coulomb friction, we assume that the normal contact forces are already determined in the usual way (penalty method), and are not themselves subject to optimization. We determine static friction forces by combining Equations 7, 8, 12, 13 and 27, which yields the following quadratic program (Equations 28-31)

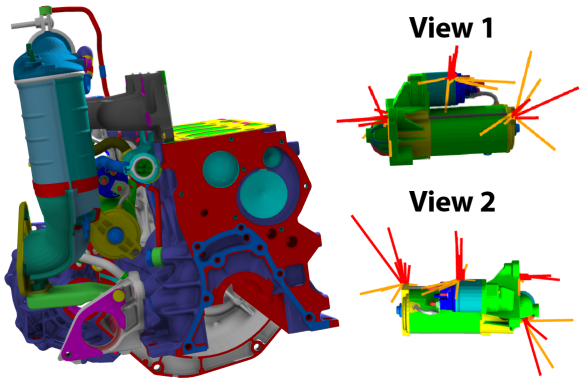


Fig. 4: **Virtual assembly with friction.** The left picture shows the car engine virtual assembling setup, with the car starter engine (highlighted) inserted in the engine. The right pictures show the contact forces on the starter motor from two different views. The red and orange lines denote normal and frictional contact forces, respectively.

$$\min_{\beta,\Delta x,\Delta\omega} \frac{1}{2}\big(\Delta x^T M \Delta x + \Delta\omega^T I \Delta\omega\big), \quad \text{subject to} \tag{28}$$
$$F + \frac{\partial F}{\partial x}\Delta x + \frac{\partial F}{\partial \omega}\Delta\omega + T\beta = 0, \tag{29}$$
$$\tau + \frac{\partial \tau}{\partial x}\Delta x + \frac{\partial \tau}{\partial \omega}\Delta\omega + R\beta = 0, \tag{30}$$
$$\beta_u^i\cos\theta_j + \beta_v^i\sin\theta_j \leq \mu\|F_n^i\| \quad \text{for all } i, j, \tag{31}$$

where i runs over all contacts $i = 0,\ldots,k-1$ ($k =$ #contacts), and $j = 0,\ldots,\ell-1$ runs over all sides of the friction pyramid. Force F is the sum of virtual coupling and normal contact forces, whereas torque τ gives the equivalent summation for torques. We denoted $T = (T^0,\ldots,T^{k-1}) \in \mathbb{R}^{3\times2k}$, $R = (R^0,\ldots,R^{k-1}) \in \mathbb{R}^{3\times2k}$, $\beta = (\beta_u^0,\beta_v^0,\ldots,\beta_u^{k-1},\beta_v^{k-1})^T \in \mathbb{R}^{2k}$, where $R^i = [r^i \times T_u^i \quad r^i \times T_v^i]$. Note that the constraints given by Equations 29-31 are feasible, as $\beta = 0$ always satisfies them; hence the given quadratic optimization problem is feasible. The case $\beta = 0$ corresponds to not applying any friction. In this case, Equations 29 and 30 uniquely determine Δx and $\Delta\omega$. This special case corresponds to standard frictionless static virtual coupling [5].

In this work, we are not interested in dynamic friction, as we are primarily concerned with virtual assembly haptic rendering applications (Figure 4). Our manipulated object is always in static equilibrium under the virtual coupling, normal and friction contact forces. We use static virtual coupling and do not simulate dynamic effects. Therefore, we designed our friction algorithm so that it detects whether all contacts are undergoing static friction; if not, this means that the object is undergoing dynamic friction and therefore we exclude friction from the static equilibrium computation. Intuitively, an object is undergoing static friction if the friction cone threshold is not activated at any contact, i.e., the inequality constraints of Equation 31 are not "active".

Definition: Mathematically, we define the object to be in static friction if there exists a solution $(\beta,\Delta x,\Delta\omega)$ to the optimization problem of Equations 28-31 with the property that $(\beta,\Delta x,\Delta\omega)$ also solves the optimization problem defined by Equations 28-30,

i.e., the solution does not change if we remove Equation 31. We can now state and prove our main result.

Theorem: An object is in static friction if and only if the following linear program for the unknown variables $(\beta, \Delta x, \Delta\omega, \lambda_1, \lambda_2)$ is feasible (Equations 32-37):

$$M\Delta x + \left(\frac{\partial F}{\partial x}\right)^T \lambda_1 + \left(\frac{\partial \tau}{\partial x}\right)^T \lambda_2 = 0, \tag{32}$$

$$I\Delta\omega + \left(\frac{\partial F}{\partial \omega}\right)^T \lambda_1 + \left(\frac{\partial \tau}{\partial \omega}\right)^T \lambda_2 = 0, \tag{33}$$

$$T^T \lambda_1 + R^T \lambda_2 = 0, \tag{34}$$

$$F + \frac{\partial F}{\partial x}\Delta x + \frac{\partial F}{\partial \omega}\Delta\omega + T\beta = 0, \tag{35}$$

$$\tau + \frac{\partial \tau}{\partial x}\Delta x + \frac{\partial \tau}{\partial \omega}\Delta\omega + R\beta = 0, \tag{36}$$

$$\beta_u^i \cos\theta_j + \beta_v^i \sin\theta_j \le \mu \|F_n^i\| \quad \text{for all } i, j. \tag{37}$$

Furthermore, if the linear program is feasible, then for any feasible solution $(\beta, \Delta x, \Delta\omega, \lambda_1, \lambda_2)$, the triple $(\beta, \Delta x, \Delta\omega)$ minimizes the quadratic program of Equations 28-31. We note that for feasibility, the specific objective function of the LP is not important; hence, we do not state it.

Proof: Suppose the object is in static friction. Then, by definition, there exists a solution $(\beta, \Delta x, \Delta\omega)$ to the optimization problem of Equations 28-30 that also satisfies Equation 31. By using the technique of Lagrange multipliers for the optimization problem of Equations 28-30, it follows that there exist Lagrange multipliers $\lambda_1, \lambda_2 \in \mathbb{R}^3$ such that Equations 32-34 are satisfied. Because Equations 31 and 37 are the same, it follows that $(\beta, \Delta x, \Delta\omega, \lambda_1, \lambda_2)$ is a feasible solution of the linear program of Equations 32-37. Conversely, suppose $(\beta, \Delta x, \Delta\omega, \lambda_1, \lambda_2)$ is a feasible solution of the linear program of Equations 32-37. Then, $(\beta, \Delta x, \Delta\omega)$ minimizes the optimization problem of Equations 28-30. Because Equations 31 and 37 are the same, $(\beta, \Delta x, \Delta\omega)$ also satisfies Equation 31. This means that $(\beta, \Delta x, \Delta\omega)$ also minimizes the quadratic optimization problem of Equations 28-31, i.e., object is in static friction. ■

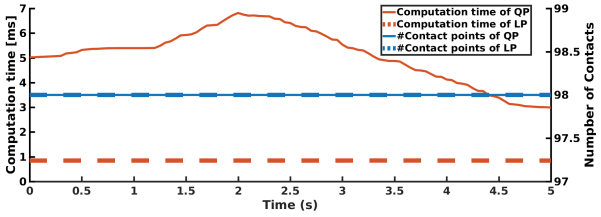


Fig. 5: **Comparison of the computational cost of LP vs QP.** The blue line shows that the numbers of contacts is 98 for both methods. The red line shows that on average, QP (5.15ms) is 6.1× slower than LP (0.85ms). Cube vs 90-deg corner example.

In each haptic cycle, we first read the position and orientation of the device and set this as the manipulandum object's position. Then, we perform discrete collision detection between the movable simulation object and the fixed environment. This gives us the contacts and the penalty normal contact forces. Then, we use our linear programming model to compute the friction forces and the new static equilibrium position of the simulation object. We do so by solving the linear program of Equations 32-37. We note that the constraint matrix formed by expressing Equations 32-37 in a matrix form is sparse. This is because the inequalities of Equation 37 only have two non-zero entries for each row of Equation 37. The number of non-zero entries is linear in k. This speeds up the SNOPT linear program solver. If there is a feasible solution, we displace the virtual object according to Δx and $\Delta\omega$. Otherwise, we apply no friction and calculate Δx and $\Delta\omega$ via static equilibrium, defined by Equations 29 and 30. After updating the position of the simulation object, we send the computed virtual coupling force and torque to the device. Since the Lagrange multipliers are only applied to the six-dimensional static equilibrium constraints (Equations 32, 33), but not the friction cone constraints, the size of our LP does not grow much compared to the original QP. Our LP method achieves a 6x speedup over QP (Figure 5).

4.1 Avoiding Stickiness

In some configurations (Figure 6), the static friction could cause spurious sticking forces, acting against separation of contact. This is due to our hybrid algorithm where the normal contact forces are determined using penalty forces and are not optimized, whereas the friction is optimized. For example, when the user tries to lift the box in Figure 6, the non-zero penalty forces at contact points create non-zero friction forces which counter-balance the upward virtual coupling force. To avoid such cases, we amend the friction model by removing friction forces at contact points that are separating while the normal contact force is preserved. To determine whether a contact is separating or not, we first pretend that there is no friction and compute the predicted displacement of all the contact points under the static virtual coupling and normal contact forces, using Equation 29 and 30 where we set $\beta = 0$. If the angle between the predicted contact point displacement and the outward contact normal is smaller than 90 degrees, and at the same time the angle between the predicted contact point displacement and the direction of the virtual coupling force is less than 90 degrees, we label this contact as separating. Only the non-separating contacts enter the linear program of Equations 32-37; we apply no friction at the separating contacts.

4.2 Static Damping

Physically, when static friction is not strong enough to keep the object motionless, dynamic friction should be applied to reduce the kinetic energy. However, since our model does not model dynamics or dynamic friction, the sliding object may jump to the static equilibrium target location rapidly from one haptic cycle to another. The motion can be made smoother by adding *static damping* to the static virtual coupling model. In each haptic cycle with collisions, after computing Δx and $\Delta\omega$ as described in Section 4, we use $(1-\alpha)\Delta x$ and $(1-\alpha)\Delta\omega$ as our final displacements. The parameter $\alpha \in [0, 1)$ controls the amount of static damping. The higher the α, the higher the "dynamic friction" felt by the user. With static damping, the object will slide more slowly and converge to the static friction equilibrium (defined by $\alpha = 0$). In our experiments, we found $\alpha = 0.6$ to be a good value that keeps the simulation stable and the virtual coupling force smooth. For each haptic cycle without collisions, we do not apply any static damping, and directly use the translation and rotation results from the static virtual coupling equilibrium. We note that static damping could be applied also to cycles without collisions, but then it just adds damping to free-space motion; we did not see stability changes if it was added, hence we omit it. We analyze static damping in Figure 7.

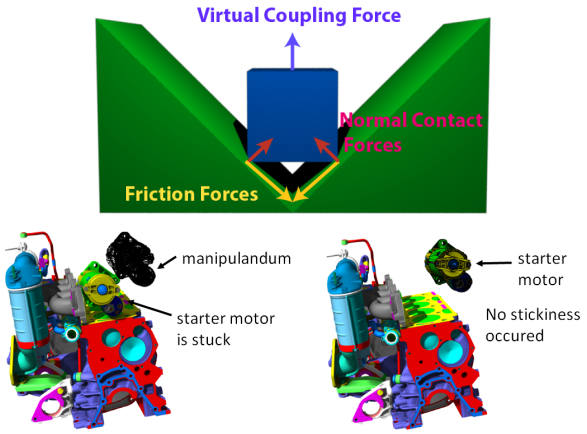
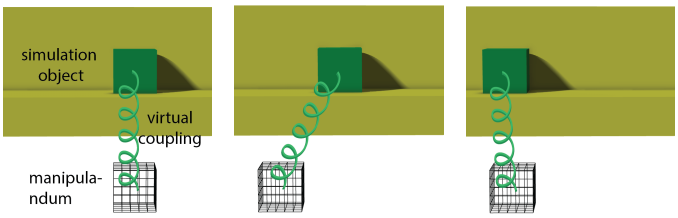


Fig. 6: **Addressing artificial stickiness.** Top: When the cube is lifted, static friction forces may point downhill, which may artificially prevent the cube from being lifted. Our method of Section 4.1 addresses this by removing such contacts from the optimization problem. Bottom-left: Without the method of Section 4.1, the starter motor sticks to the engine even when the manipulandum (shown in wireframe) is pulling it out of contact. Bottom-right: With our method, no stickiness occurs even with high friction ($\mu = 10$).



(a) Initial frame (b) After one frame; static damping is on (c) After one frame; static damping is off

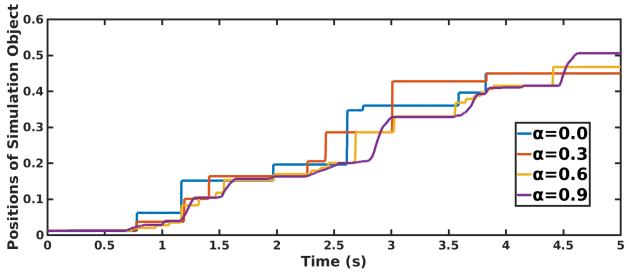


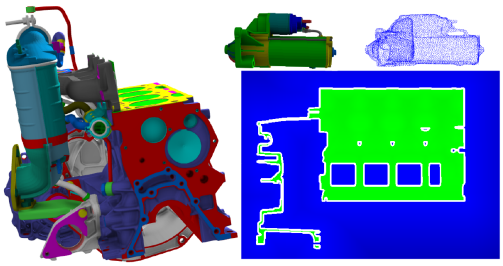
Fig. 7: **Static damping:** The top figures show how the static damping ($\alpha = 0.6$) influences the simulation. The wireframe cube indicates the haptic manipulandum position. From the initial position shown in the top-left figure, the user moves the manipulandum to the left. In the top-middle figure, due to static damping, the simulated cube stops halfway to the position above the manipulandum. Without static damping, the cube can move very far in one cycle (top-right). The bottom figure shows the equivalent simulation result for the starter motor sliding on the car engine model. Larger static damping coefficients α produce smoother simulation object motion.

5 Results

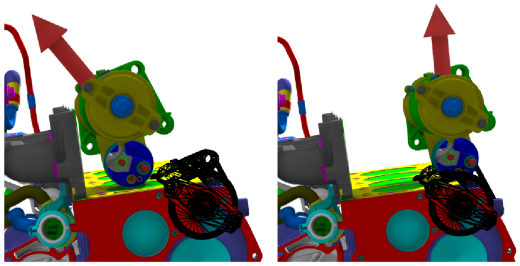
We conducted a number of experiments to verify the performance and robustness of our friction method. We render our forces and torques using the 6-DoF Haption Virtuose 6D haptic device. We use the SNOPT Library [38] to solve all linear programming problems. All examples were computed on a 3.00GHz Intel Xeon i7 CPU E5-2687W v4 processor using a maximum of 7.4 GB of memory and a Quadro P5000 graphics card with 16 GB of RAM. We used a single thread (the haptics servo thread generated by the Virtuose API) to solve the linear programs to compute the friction force. All examples run at 1,000 Hz haptic rates.

5.1 Car Engine

In our first experiment of haptic virtual assembly, we manipulate a car starter motor into its target location in a car engine. In this example, the manipulated motor is modeled by a point shell (8,517 points, 8-level nested point tree), and the fixed car engine is modeled by a $1024 \times 1024 \times 1024$ signed distance field (Figure 8a). We use a 8-regular pyramid to approximate the friction cone.



(a) Car engine and starter motor geometry



(b) The contact force on the simulation object

Fig. 8: **Static Coulomb friction.** The right-top image depicts the point shell of the starter motor. The blue points indicate the points. We show one slice of the signed distance field of the car engine, where green represents interior and blue represents exterior. In the bottom, the black wireframe shows the manipulandum position. The red arrow represents the total contact force on the object, which is the sum of the normal contact forces and the friction forces, and is the force felt by the user via the haptic device. Our method correctly simulates the friction force (bottom-left). Note that the contact force in the frictionless case is always normal to the contact (bottom-right), whereas our contact force has a tangential frictional component (bottom-left).

Our static friction prevents the starter motor from sliding on the surface of the car engine (Figure 8b). We also analyze the time complexity of computing the friction (Figure 9). Our model can achieve the 1,000 Hz haptic rate with over 100 contact points.

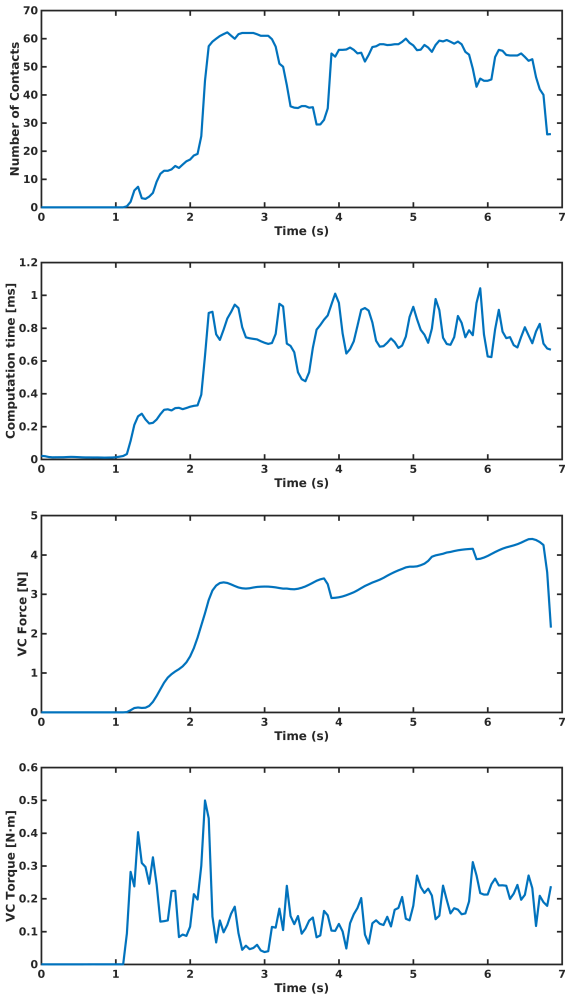
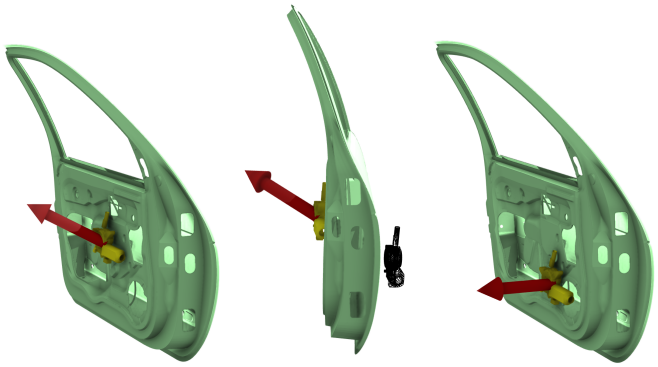


Fig. 9: **Performance and force rendering of the car engine.** Here, the starter motor is sliding on the engine surface as shown in Figure 8b. The number of contacts in this experiment is approximately 50 (top). The second figure shows that it costs around 0.76 msec to compute the friction forces. This time budget is sufficient to meet the 1,000 Hz haptic simulation requirement. The third and forth figure show the magnitude of the total contact forces and torques.

5.2 Car Door

Our second experiment is to manipulate a window motor into a car door. The window motor is modeled by a $256 \times 256 \times 256$ signed distance field and the fixed car door is model by a point shell (30,017 points, 8-level nested point tree). An 8-regular pyramid is used to approximate the friction cone. When the motor collides with the surface of the car door, static friction, acting in a direction parallel to the contact surface, is able to prevent the sliding.

In Figure 10, parts (a) and (b) refer to a simulation with friction. We can see that the total contact force, represented by the red arrow, is not perpendicular to the surface, due to the presence of friction. For experiment, we also slide the manipulated motor tangentially, and track the magnitudes of all forces (Figure 11). It can be seen that while the virtual coupling force is smoothly changing, the normal contact force may change sharply due to the change in the number of contact points. However, the static friction is always able to respond immediately to compensate the sharp normal contact force changes, keeping the object in place.



(a) With friction (b) With friction (c) Frictionless

Fig. 10: **Car door static friction.** The red arrow represents the contact force on the object. (a) Our method can simulate static friction. (b) Another view, also showing the manipulandum position (black wireframe). (c) Comparison to frictionless contact.

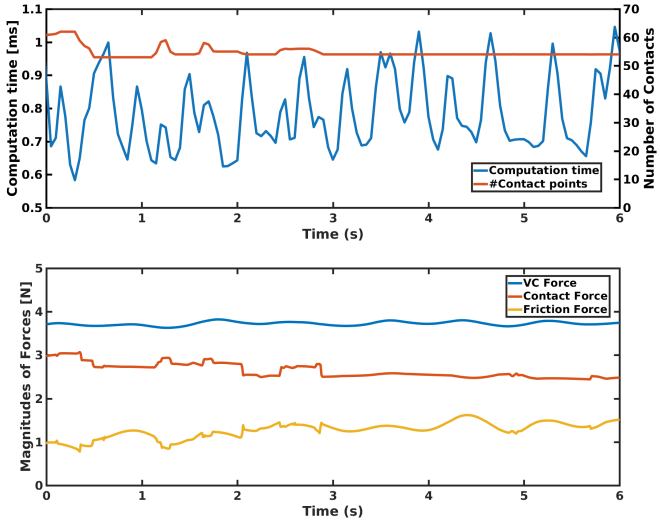
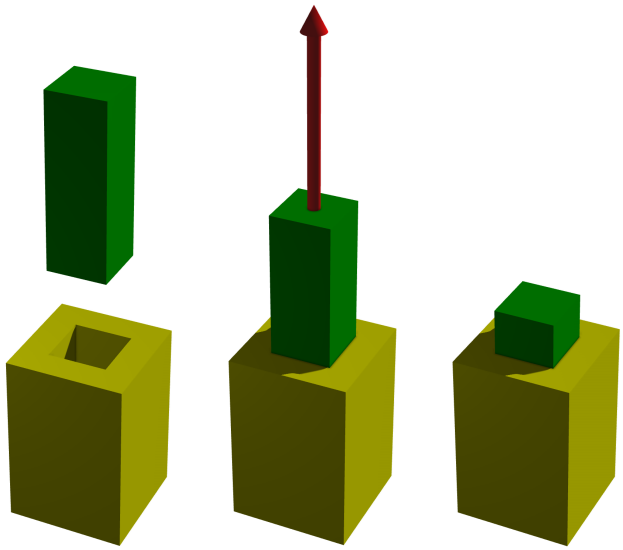


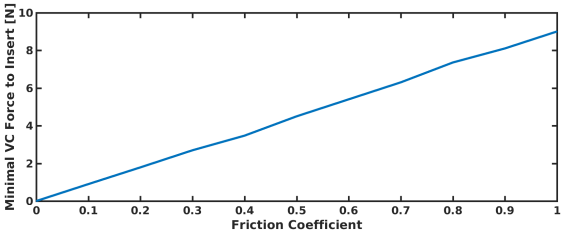
Fig. 11: **Performance and results of forces on the car door.** The top figure shows that the number of contacts is approximately 55. It costs approximately 0.86 msec to compute the friction contact forces on average, which is sufficient to maintain the 1,000 Hz haptic update rate. The bottom figure shows the magnitudes of the virtual coupling force, the contact force and the friction force.

5.3 Peg Insertion

In our third example, we insert a peg into a hole. The size of the hole is 1% smaller than the size of the peg. The peg is modeled by a $128 \times 128 \times 128$ signed distance field whereas the peg is modeled by a point shell with 672 points and a nested point tree with 6 levels. We use an 4-regular pyramid to approximate the friction cone. Due to the difference in size, there is always a slight penetration and the penalty normal contact force always exists when the peg is inserted. In such configurations, friction is unavoidable and it may block the insertion process if the friction coefficient is sufficiently high (Figure 12).



(a) Before insertion (b) With friction (c) Frictionless



(d) Minimal force to insert

Fig. 12: **Hole-peg insertion under varying friction coefficients.** A peg is manipulated into a slightly smaller hole. Without friction, the insertion can be done easily and with a very small virtual coupling force. With friction, the insertion requires a larger virtual coupling force. The bottom figure shows the relation between the minimal virtual coupling force to insert the peg and the friction coefficient. The observed linear relation confirms the correctness of our Coulomb's static friction cone. The stiffness and stiffness scaling [4] is the same in all experiments.

5.4 90-Degree Corner and Cube

Our fourth example is to slide a cube against a 90-degree corner shape (Figure 13). The yellow corner is modeled by a point shell with 7,772 points and a nested point tree with 8 levels, whereas the green cube is modeled by a $256 \times 256 \times 256$ signed distance field. We use a 4-regular pyramid to approximate the friction cone. We set a high friction coefficient to measure the quality of static friction. Our static friction prevents the cube from sliding.

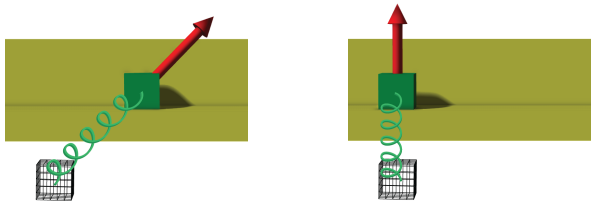


Fig. 13: **Cube manipulated against a 90-degree corner.** The red arrow gives the total contact force on the cube.

We also did this experiment with a high static damping coefficient ($\alpha = 0.9$), and moved the manipulandum object left-and-right very quickly. Under such high static damping, when the static friction is not able to keep the cube motionless, the cube will slowly slide towards to manipulandum position until the static friction is strong enough to keep the cube motionless again. When the cube is sliding, the linear programming solver will detect that no feasible static friction is possible. Figure 14 analyzes the performance and transitions between static friction and sliding.

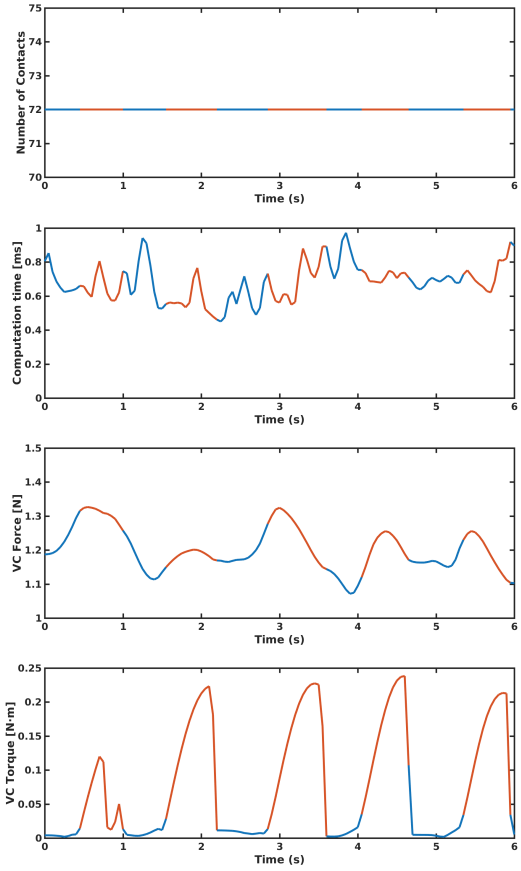


Fig. 14: **Simulation performance of the 90-degree corner and cube example.** In these figures, the blue lines indicate that the object is in static friction, whereas the orange lines indicate that the object is sliding. The top figure shows that the number of contacts in this experiment is approximately 70. The second figure indicates that the cost of each haptic cycle is approximately 0.72 msec, which is sufficient to maintain the 1,000 Hz haptic simulation rates.

5.5 Comparison with constraint-based methods

We have compared our method with two constraint-based methods: Staggered Projections by Kaufman [9] and Gauss-Seidel-like method by Duriez [10]. We performed our comparison experiment on the 90-degree corner model. When sliding the cube, the number of contact points is approximately 72. The computation time (Figure 15) shows that our method can maintain the 1,000 Hz haptic rate under a moderate number of contact points. The computation rates of the two compared constraint-based methods fall below 100 Hz.

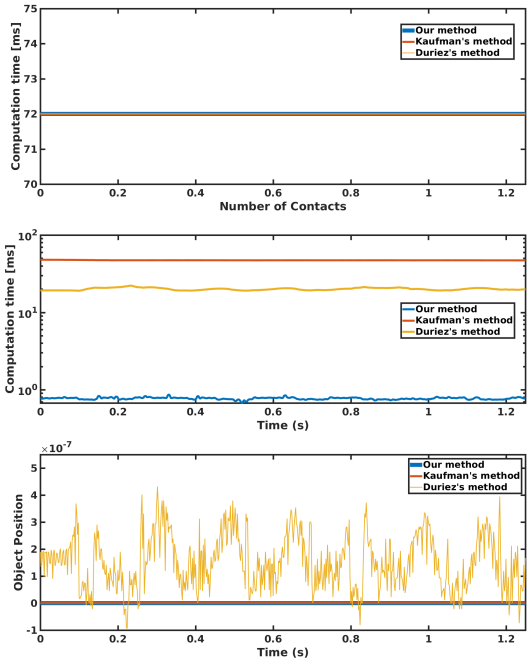


Fig. 15: **Time cost and static friction quality of our method and constraint-based methods.** It can be seen that our method is about 50× faster than the two constraint-based methods. The quality of the friction computed by our method is as good as in Kaufman's staggered projections [9] (both methods keep zero positions at all times), and better than using the Gauss-Seidel-like method of Duriez [10] which permits a small (but practically negligible) offset.

5.6 Comparison to Yamane's frictional springs

We also compare our method with the penalty-based frictional springs of Yamane [6]. We use implicit integration of Yamane's springs. Otherwise, if the stiffness of the friction spring is set high to reduce the sliding, the virtual coupling system easily becomes unstable under explicit Yamane's springs. Even with implicit springs, we observe a stability stiffness limit in practice. Because of its simplicity, Yamane's method can compute friction about 6× faster than our linear programming method. However, we will now demonstrate in three separate experiments that our method produces better static friction quality than Yamane's friction. First, our static friction can stop the cube without any residual motion (Figure 16). In Yamane's method, the static friction originates from stretching the friction springs, which permits the object to always incorrectly slide when it should be motionless.

Second, we observe that Yamane's friction has the following flaw: it is possible to "cheat" Yamane friction via "snake"-like motion (Figure 17). In this experiment, we set a very high friction coefficient ($\mu = 100$). For our method, we experimentally verified that our friction correctly prevents the "snake" peg insertion (Figure 17, a). However, with Yamane's springs, if we manipulate the peg in left-right "snake" motion when inserting the peg, we will cause the contacts on the left side of the peg to appear and disappear, and same for the right side. Therefore, the Yamane friction anchors will keep being removed and re-formed as contacts appear and disappear. In each cycle the anchor position moves slightly down. As a result, the peg will be slowly but steadily incorrectly inserted. This does not happen in our method.

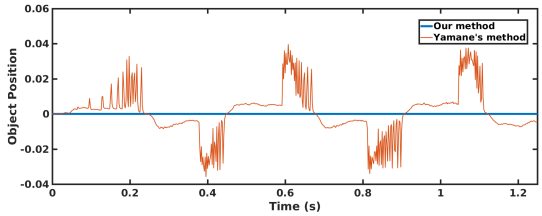


Fig. 16: **Static friction quality of our method vs Yamane's frictional springs.** Our static friction keeps the position of the simulation object fixed, regardless of manipulandum motion. However, with Yamane's friction, the object will translate (creep) to follow the manipulandum object.

We note that the non-stickiness technique of Section 4.1 was enabled in all of our examples. In our method, because the size of the peg is slightly larger than the hole, the peg will always be in contact with at least one side of the hole. Therefore, in practice, our LP formulation keeps the object still, preventing the snake maneuver, unless the user applies a large force that exceeds the friction cone limit. Note that for the technique of Section 4.1 to activate, one required condition is that the virtual coupling force must have a positive dot product with the contact normal. If one attempts to perform the "snake" motion using our method, in order to exploit the presence of the technique of Section 4.1, one has to start from a state where the virtual coupling is pushing the peg deep into contact, so that the opposite side of the peg is contact free. Then, one would have to abruptly move the manipulandum a large distance out of contact in one haptic cycle, so that the positive dot product condition is satisfied. Note that if done more slowly, the contacts on the other side of peg will activate during subsequent haptic cycles, blocking the maneuver. Such abrupt manipulandum motion is not possible in practice due to the human limits on the manipulandum speed. In contrast, with Yamane's method the snake maneuver is possible with arbitrarily smooth user manipulandum motion, because the Yamane's method always permits sliding.

We note that the virtual coupling stiffness does not actually affect the feasibility of the snake-like motion. If virtual coupling stiffness was made smaller and smaller, then the user can still break the contacts to perform the snake motion in Yamane's method; she will just need to move the manipulandum deeper into contact. Conversely, if virtual coupling stiffness was made larger and larger, then the Yamane contacts will break already for a small manipulandum motion into the contact direction. In either case, they will break, permitting the snake motion in Yamane's method.

We also compare our method to Yamane's friction in the car engine example (Figure 18). Under a friction coefficient of $\mu = 0.8$, the starter motor cannot be inserted using our method (and also not using Duriez's and Kaufman's methods), even if very large virtual coupling forces are used. With Yamane's method, the motor can be spuriously inserted at $\mu = 0.8$ all the while the virtual coupling forces are smaller than in our method. Furthermore, Yamane's method permits a spurious motor insertion even under much larger friction coefficients; the largest we tried was $\mu = 1000$.

6 Conclusion

We presented a stable algorithm to compute static Coulomb friction for 6-DoF distributed haptic contact between rigid objects with complex geometry. Our method can be combined with the existing penalty-based 6-DoF haptic rendering methods to provide stable static friction rendering. By using our novel combination of penalty normal contact forces and frictional constraints, we are able to achieve haptic-rate performance for complex scenes and more than 100 frictional contacts, without the need for multirate simulation.

7 Future Work

In the future, we would like to incorporate correct dynamic friction into our model, as well as study the applications of our method to computer graphics and animation. Our friction makes use of static virtual coupling to stabilize haptic rendering. We do not simulate dynamic friction and hence the dynamics under sliding is not physically accurate. We use static damping as our "quasi-dynamics". In the future, we would like to combine dynamic virtual coupling and friction, by properly incorporating the velocity of the simulated object into the friction model. As we focus on quasi-static modeling without dynamics, our fast linear programming method is not designed to handle configurations where both static and dynamic friction exist simultaneously on the object. Currently, if one wants to simulate both static and dynamic friction, the user must specify what contacts are expected in static friction and what contacts are expected in dynamic friction. By removing variables β corresponding to dynamic friction contacts from Equations 32- 37, our model can either compute correct static friction for the rest of the contacts, or determine that the user's specification is incorrect. Also the system may take a guess about what contacts are likely in static friction by the size of each friction cone.

In the future, one may attempt to handle such hybrid configurations by investigating heuristics to determine the contacts where the static friction cone constraints may be satisfied. As demonstrated by [22], it is possible to make constraint-based methods compliant; it would be interesting to apply our LP friction ideas to such methods. Similarly, one could attempt to combine our LP friction with velocity-based formulation for static equilibria. For simplicity, we did not use multirate simulation in this paper, although in principle our work can be integrated into a multirate system as well. Often, insertion in the real-world is made possible by geometrically enlarging narrow passages via plastic deformations, either of the environment or the manipulated object. Modeling of complex plastic deformations would be interesting future work.

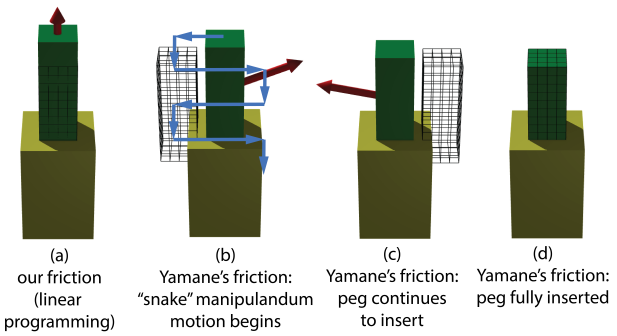


Fig. 17: **Spurious insertion via "snake" motion in Yamane's friction.** The red arrow gives the total contact force on the peg. The wireframe gives the manipulandum location. Part (a) shows that the peg correctly cannot be inserted under our friction. Parts (b,c,d) show Yamane's friction. The user moves the manipulandum in a snake-like motion (indicated in (b)). This causes the peg to slowly sink into the hole because the contacts keep appearing and disappearing in alternation on the left and right wall. Consequently, the Yamane friction anchors keep breaking and reappearing at a lower and lower position. Part (d) shows that the peg eventually becomes fully inserted via the snake motion.

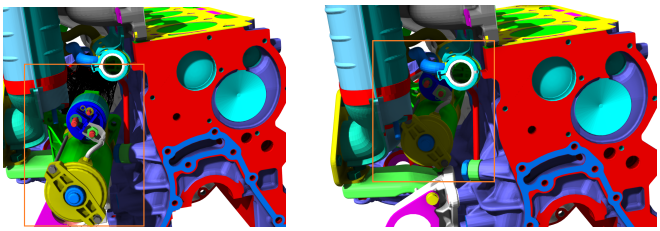


Fig. 18: **Comparison to Yamane's friction on the car engine model.** Left: the motor cannot be inserted with our method ($\mu = 0.8$). Right: with Yamane's method, the motor can be spuriously inserted into the car engine, despite a much larger friction coefficient ($\mu = 1000$), and a smaller virtual coupling force.

Acknowledgments

This research was sponsored in part by the National Science Foundation (CAREER-1055035, IIS-1422869), the Sloan Foundation, the Okawa Foundation, and USC Annenberg Graduate Fellowships to Danyong Zhao and Yijing Li.

References

[1] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," in *Proc. of ACM SIGGRAPH 99*. ACM, 1999, pp. 401–408.

[2] ——, "Six degree-of-freedom haptic rendering using voxel sampling," in *ACM SIGGRAPH 2005 Courses*. ACM, 2005, p. 42.

[3] J. Barbič and D. L. James, "Six-dof haptic rendering of contact between geometrically complex reduced deformable models," *IEEE Transactions on Haptics*, vol. 1, no. 1, pp. 39–52, 2008.

[4] H. Xu and J. Barbič, "Adaptive 6-dof haptic contact stiffness using the gauss map," *IEEE Transactions on Haptics*, vol. 9, no. 3, pp. 323–332, 2016.

[5] M. Wan and W. A. McNeely, "Quasi-static approximation for 6 degrees-of-freedom haptic rendering," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. IEEE Computer Society, 2003, p. 34.

[6] K. Yamane and Y. Nakamura, "Stable penalty-based model of frictional contacts," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1904–1909.

[7] D. E. Stewart, "Rigid-body dynamics with friction and impact," *SIAM review*, vol. 42, no. 1, pp. 3–39, 2000.

[8] D. Stewart and J. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," vol. 39, pp. 2673–2691, 1996.

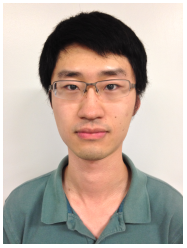
[9] D. M. Kaufman, S. Sueda, D. L. James, and D. K. Pai, "Staggered projections for frictional contact in multibody systems," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5. ACM, 2008, p. 164.

[10] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic Haptic Rendering of Interacting Deformable Objects in Virtual Environments," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 12, no. 1, pp. 36–47, 2006.

[11] J. Barbič, "Real-time reduced large-deformation models and distributed contact for computer graphics and haptics," Ph.D. dissertation, Carnegie Mellon University, Aug. 2007.

[12] S. D. Laycock and A. Day, "A survey of haptic rendering techniques," in *Computer Graphics Forum*, vol. 26, no. 1. Wiley Online Library, 2007, pp. 50–65.
[13] M. C. Lin and M. Otaduy, *Haptic rendering: foundations, algorithms, and applications*. CRC Press, 2008.
[14] M. Renz, C. Preusche, M. Pötke, H.-P. Kriegel, and G. Hirzinger, "Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm," in *Proc. of Eurohaptics*, 2001, pp. 149–154.
[15] M. Sagardia, T. Hulin, C. Preusche, and G. Hirzinger, "Improvements of the voxmap-pointshell algorithm-fast generation of haptic data-structures," in *53rd IWK-Internationales Wissenschaftliches Kolloquium, Ilmenau, Germany*, 2008.
[16] Y. J. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha, "Six-degree-of-freedom haptic rendering using incremental and localized computations," *Presence: Teleoperators and Virtual Environments*, vol. 12, no. 3, pp. 277–295, 2003.
[17] Q. Luo and J. Xiao, "Physically accurate haptic rendering with dynamic effects," *IEEE Computer Graphics and Applications*, vol. 24, no. 6, pp. 60–69, 2004.
[18] M. A. Otaduy and M. C. Lin, "A modular haptic rendering algorithm for stable and transparent 6-dof manipulation," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 751–762, 2006.
[19] L. Glondu, S. C. Schvartzman, M. Marchal, G. Dumont, and M. A. Otaduy, "Fast collision detection for fracturing rigid bodies," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 1, pp. 30–41, 2014.
[20] D. D. Nelson, D. E. Johnson, and E. Cohen, "Haptic rendering of surface-to-surface sculpted model interaction," in *ACM SIGGRAPH 2005 Courses*. ACM, 2005, p. 97.
[21] M. A. Otaduy and M. Gross, "Transparent Rendering of Tool Contact with Compliant Environments," in *Proc. of the World Haptics Conference*, 2007, pp. 225–230.
[22] M. Tournier, M. Nesme, B. Gilles, and F. Faure, "Stable constrained dynamics," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 132, 2015.
[23] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," in *ACM Siggraph Computer Graphics*, vol. 22, no. 4. ACM, 1988, pp. 289–298.
[24] E. Drumwright, "A fast and stable penalty method for rigid body simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 231–240, 2008.
[25] H. Xu, Y. Zhao, and J. Barbič, "Implicit multibody penalty-based distributed contact," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 9, pp. 1266–1279, 2014.
[26] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, 1994, pp. 23–34.
[27] B. V. Mirtich, "Impulse-based dynamic simulation of rigid body systems," Ph.D. dissertation, University of California at Berkeley, 1996.
[28] R. Bridson, R. Fedkiw, and J. Anderson, "Robust Treatment of Collisions, Contact, and Friction for Cloth Animation," *ACM Trans. on Graphics*, vol. 21, no. 3, pp. 594–603, 2002.
[29] D. Harmon, E. Vouga, R. Tamstorf, and E. Grinspun, "Robust treatment of simultaneous collisions," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3. ACM, 2008, p. 23.
[30] B. Brogliato and V. Acary, "Numerical methods for nonsmooth dynamical systems," *Lecture Notes in Applied and Computational Mechanics*, vol. 35, 2008.
[31] S. Niebe and K. Erleben, "Numerical Methods for Linear Complementarity Problems in Physics-Based Animation," *Synthesis Lectures on Computer Graphics and Animation*, vol. 7, no. 1, pp. 1–159, 2015.
[32] R. Cottle, J.-S. Pang, and R. E. Stone, *The linear complementarity problem*. Society for Industrial and Applied Mathematics, 2009.
[33] E. Drumwright and D. A. Shell, "Modeling contact friction and joint friction in dynamic robotic simulation using the principle of maximum dissipation," in *Algorithmic Foundations of Robotics IX*. Springer, 2010, pp. 249–266.
[34] K. Schittkowski, "Ql: A fortran code for convex quadratic programming-users guide," *Report, Department of Mathematics, University of Bayreuth*, pp. 64–71, 2003.
[35] F. Barbagli, D. Prattichizzo, and K. Salisbury, "A multirate approach to haptic interaction with deformable objects single and multipoint contacts," *The International Journal of Robotics Research*, vol. 24, no. 9, pp. 703–715, 2005.
[36] H. Xu and J. Barbič, "Signed distance fields for polygon soup meshes," in *Proc. of the Graphics Interface Conf.*, 2014, pp. 35–41.
[37] J. J. Moreau, "On unilateral constraints, friction and plasticity," in *New variational techniques in mathematical physics*. Springer, 2011, pp. 171–322.
[38] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

Danyong Zhao is a PhD student in computer science at the University of Southern California. He obtained his BS degree from Tsinghua University. His research interests are in computer graphics, physically based animation, contact and haptics.



Yijing Li is a PhD student in the Department of Computer Science, Viterbi School of Engineering, University of Southern California. He received his undergraduate degree from Tsinghua University. His research interests are in computer graphics, physically based animation, contact and interactive physics.

Jernej Barbič is associate professor of computer science at USC. In 2011, MIT Technology Review named him one of the Top 35 Innovators under the age of 35 in the world (TR35). Jernej's research interests include nonlinear solid deformation modeling, model reduction, collision detection and contact, and interactive design of deformations and animations. He is the author of Vega FEM, an efficient free C/C++ software physics library for deformable object simulation. Jernej is a Sloan Fellow (2014).