Master's thesis

Rendering 6-DOF Object-to-Object Interaction with 3-DOF Haptic Interfaces

T. C. Lee

Delft, 22nd of October 2014

Student No.: 4246748

Report No.:

Department of Biomechanical Engineering

Faculty 3mE

Delft University of Technology

Rendering 6-DOF Object-to-Object Interaction

with 3-DOF Haptic Interfaces

TsungChi Lee

Preface

I always like to picture that someday in the future, what the world would look like if people are used to interacting with the multimedia devices with realistic haptic or tactile feedback in their daily life. As far as I am concerned, the development of haptic devices could even bring a larger impact to our life than the visual displays due to its bidirectional-interaction nature. Via a haptic feedback device, we will be able to extract and manipulate the desired multi-dimensional information actively and intuitively. However, there are still many obstacles on the way of developing haptic interfaces and displays.

In this project, my primary goal is to solve a long existing rendering problem concerning the stability in the haptic feedback. The major concepts of my Master's thesis are presented in a form of research paper in order to introduce the essential ideas in a compact way. In the appendices, all the major concepts are introduced with more details, explanations, and examples.

In Appendix A, the simulations built in MATLAB for testing algorithms are presented. Appendix B shows how the original rendering algorithm works, and provides a problem analysis to explain why it failed in certain scenario. Appendix C gives a detailed introduction of the proposed haptic rendering algorithm, which is the major contribution of this work. In Appendix D, a comparison between the proposed and original rendering algorithm is provided with the result of two simple man-machine experiments. Appendix E provides more data figures about the validation test used in the paper form thesis. Last but not least, Appendix F gives several ideas and concepts I have developed other than the primary rendering algorithm during this project. These ideas are rather immature and not included in the main frame of the thesis. However, I think it would still be interesting to those who would like to work in the same field in the future.

Finally, I would like to thank all who support and help me throughout the whole project. I would like to thank my supervisors David and Piet for their efforts in reviewing all my immature ideas and guiding me into right direction. I would also like to thank Mark, Eyal and Niels, who are the engineers at Moog Inc., for their professional advices and patience. I would like to thank my two kind "colleagues", Jasmien and Siddhi, for their full encouragement while I was stressed in the end phase of my Master thesis. Special thanks to Steven Liu, for all the discussions we had throughout my master work. Finally, I would like to thank my family and my girlfriend for their kind support from Taiwan in the last two years.

Rendering 6-DOF Object-to-Object Interaction with 3-DOF Haptic Interfaces

Tsung-Chi Lee, Piet Lammertse, and David A. Abbink

Abstract— Three degree-of-freedom (3-DOF) tool-based haptic interfaces are widely used in virtual environments as an affordable way to train operators, as well as for virtual prototyping and design. In some special cases with higher requirements on haptic fidelity, the tool needs to be modeled as an object with real volume rather than a single point. However, such object-to-object interaction will inherently involve reaction torques, which 3-DOF haptic interfaces are incapable of rendering. As a result, whenever reaction torques are induced, undesired system behavior will occur, such as rendering errors, vibrations or even instability. Six degree-of-freedom (6-DOF) interfaces with torque feedback would be a solution of this problem, but these devices are costly, hard to maintain, and would require more computational power to determine the feedback. This paper presents a penalty-based algorithm to realize stable yet convincing object-to-object interaction with 3-DOF haptic interfaces. The major contribution of this work is the regulation of excessive directional combined stiffness when multiple contact points are considered in the calculation of rendered force. In contrast to other 3-DOF rendering methods, this approach is to generate translational movement to resemble the dynamics of end-effector during torque-involved interaction. A virtual peg-in-hole task was conducted to evaluate the performance of the proposed algorithm. We used the geometrical constraints to calculate an ideal trajectory of the end-effector as a function of the peg's orientation. The result shows that the end-effector's trajectory resembled the ideal one when the virtual tool was rotated in the hole. We also showed that the regulated combined stiffness converged to a desired value so that the system stayed stable throughout the whole interaction.

Index Terms—Haptic Rendering, Distributed Contact, Combined Stiffness, Penetration Depth

1 Introduction

Three degree-of-freedom (3-DOF) tool-based haptic interfaces are widely used in virtual environments as an affordable method to train operators [1][2][3][4][5][6], and for virtual prototyping and design [7][8][9]. For example, it can mimic the reaction force between the scalpel and the soft tissues, giving the surgeon an immersive environment for practicing delicate operations; it can also be used in virtual assembly task, in which the rendered force gives the user an intuitive impression about the interaction between the virtual objects.

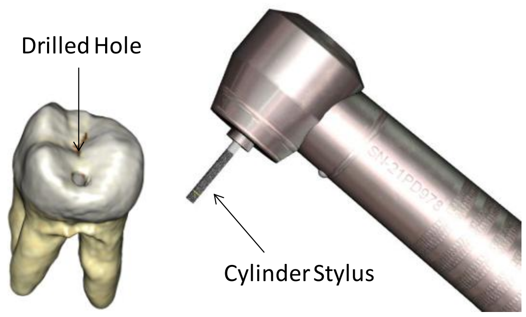


Figure 1. In dental operations, the dentist sometimes has to remove the corrosive area of the tooth with milling or drilling tools. The drilled hole and the tool formed a typical peg-in-hole configuration, which would involve reaction torque during the interaction. (Moog Inc.)

Typically, for 3-DOF tool-based haptic interfaces a hand-held tool replica is attached to a series of mechanical linkages driven by three actuators, e.g. [10][11]. Often, both position and orientation of the tool replica are measured and treated as the reference of the virtual tool's configuration. In other words, on such interfaces, there are more sensors than actuators, resulting in an under-actuated system [12].

Conventionally, the operated tool is modeled in the virtual environment as a single point. This inherently prevents virtual torques, and only translational feedback forces have to be rendered, e.g. [13][14]. The point represents the position of the end-effector in the virtual world and can be moved by the human operator via the haptic interface. When the end-effector penetrates the boundary of the virtual environment, the system will generate a reaction force to stop the end-effector from moving further.

In some special 3-DOF rendering cases with higher requirements on haptic fidelity, the tool needs to be modeled as an object with real volume [2][15][16]. For example, in the dental preparation process, the dentist often has to remove the corrosive area with various types of milling tools, and thus the geometric of the tooltip must be taken into account when calculating the feedback force.

There are two major advantages of using object-like end-effectors in 3-DOF rendering. First, it allows the operator to interact with the virtual environment with any part of the tool: the tip, the side, or even the end. Second, the resultant force will be dependent on the geometric of the tool. These features largely increase the fidelity of the haptic feedback and provide a visually and haptically

- *Tsung-Chi Lee, David A. Abbink are with the Delft Haptic Lab, Delft University of Technology, Delft, the Netherlands. E-mail: D.A.Abbink@tudelft.nl*
- *Piet Lammertse is with the Robotics Division, Moog Inc., Nieuw-Vennep, the Netherlands. plammertse@moog.com*

immersive environment for tool-based simulations.

However, such object-to-object interaction (e.g. the peg-in-hole task) will inherently involve torque (Fig. 1), which per definition cannot be realized by 3-DOF haptic interfaces. On current platform, we observed that applying conventional 3-DOF rendering algorithm to the rendering of tasks that involved multiple contact sites (as known as the distributed contact, [17]) would result in undesired behavior such as vibrations, sudden repulsive forces or easy penetrations, etc. To the author's knowledge, this issue was not addressed in previous 3-DOF rendering studies, presumably due to the simplified tasking environment used in their validation tasks.

Indeed, a straightforward solution to the haptic rendering of object-to-object interaction is to use a 6-DOF haptic interface that could perform torque feedbacks. However, in contrast to 3-DOF devices, 6-DOF interfaces are costly, hard to maintain, and require more computational power for calculating the feedback commands.

In this paper, we present a penalty-based method that allows for stable yet convincing haptic rendering of object-to-object interaction with 3-DOF haptic interfaces. We particularly focused on the regulation the combined stiffness in all directions to ensure the stability of the system. Moreover, our method could generate a pure translational movement to compensate for the lack of torque feedback. To evaluate the performance of the proposed method, we chose the peg-in-hole interaction as the standard testing task. We expect that this work to be an alternative solution when:

- tasks require the simulation of 6 DOF object-to-object interaction with affordable commercially available 3 DOF interfaces
- the fidelity of the torque feedback has a minor influence on the task performance

1.1 Overview

1.1.1 Peg-in-hole Interaction

Peg-in-hole is one of the most challenging scenarios that involves distributed contact, and is often used as a benchmark of the performance of 6-DOF rendering algorithm [17].

In applications such as dental operation or virtual sculpting, the operator sometimes has to explore a narrow drilled hole with thin hand tool. This forms a typical peg-in-hole task: when the human operator rotates a tool in the hole (Fig. 2a), ideally, the reaction force from the corner and the side wall should induce a reaction torque that stops the rotational movement. Without the actual torque feedback, as what would happen in most 3-DOF haptic interfaces, the tool would penetrate the object quickly whereas the operator only perceive a small reaction force due to the mutual cancelling of the reaction forces along the horizontal direction.

Large penetration depth caused by distributed contact can result in several problems. First, it dramatically decreases the fidelity of both visual and haptic feedback. Second, by using penalty-based rendering methods, it can cause the well-known pop-through effect [13]. Last but not least, most of the existing methods for rendering distributed contact with 3-DOF interface will suffer from undesired vibration due to the improperly estimated direction of the force feedback.

1.1.2 Combined Stiffness

Often, the rendered force has to be determined by combining individual penetration depth of forces when multiple surface constraints are applied to the end-effector simultaneously. This, from the aspect of stiffness, can be regarded as combining multiple individual springs with identical stiffness. In typical 3-DOF rendering where the end-effector is modeled as a point, the combined stiffness could deviate from the desired value, depending on the angle between surface constraints [13]. In the extreme case, the combined stiffness can be **n** times larger than the default value when the **n** surface constraints are nearly parallel to each other, resulting in an excessive stiffness that cannot be rendered by the haptic interface. As a result, the excessive stiffness could lead to unstable system behavior, which is also known as the "energy leak" [18].

Similar problem was observed in both 3-DOF and 6-DOF haptic rendering of object-to-object interaction. Common solutions are taking average among the individual forces, or using the virtual coupling to bound the combined stiffness. But none of these methods can prevent the easy-penetration problem observed in the peg-in-hole task.

1.1.3 Combined Damping

Similar to the combined stiffness, the damping can easily accumulate when multiple constraints are applied on the end-effector simultaneously. More importantly, the combined damping can change the relative damping ratio of the overall system, making the system over-damped or under-damped (see Appendix C for more detail). Both cases could cause system instability in a discrete system. In contrast to the attention the combined stiffness has received, combined damping was hardly discussed in the previous studies.

1.2 Related Work

Three degree-of-freedom (3-DOF) haptic rendering of object-to-object interaction allows for higher fidelity force feedback in applications that the end-effector must be modeled as a tool with real volume, such as dental operations [15][3], and bone drilling operations [2][6].

Given the lack of torque feedback, the major challenge of the rendering of object-to-object interaction is to determine the rendered force under distributed contact configuration [17] while maintain the system stability and feedback fidelity.

Excessive combined stiffness is a common issue in penalty-based haptic rendering when there are more than one contact points between two colliding objects (Fig. 2a). It arises naturally when the rendered force is determined by summing the sub-forces corresponding to all the colliding points.

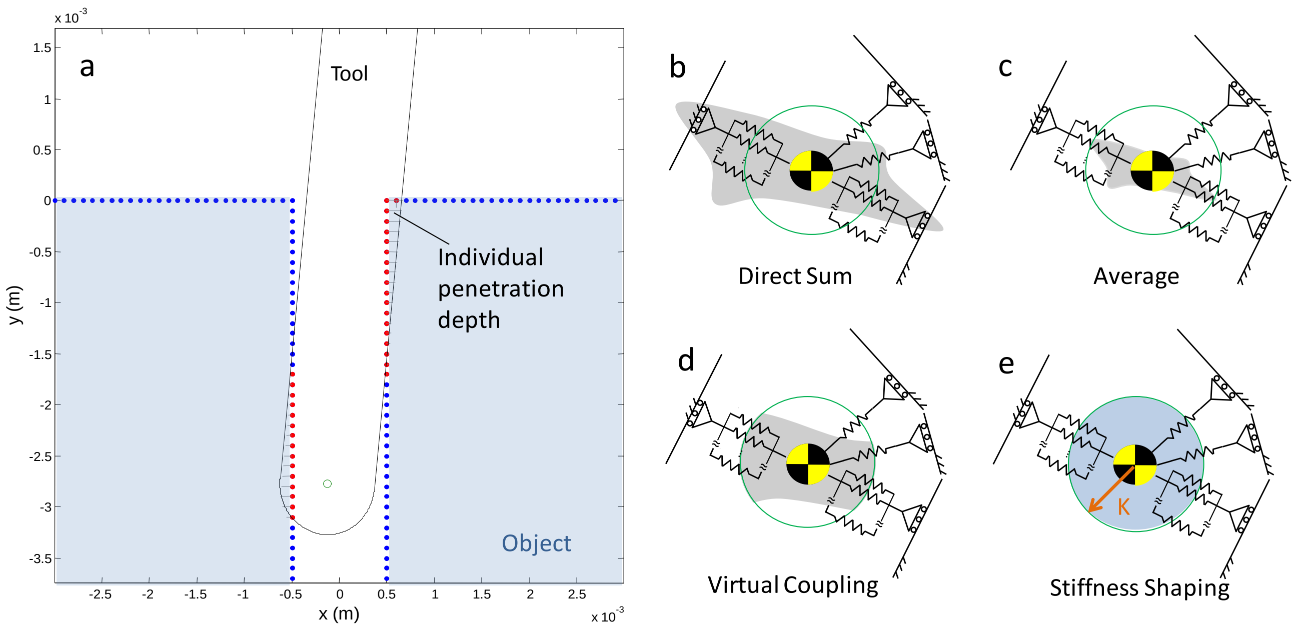


Figure 2. (a) 2D representation of the problem of excessive stiffness in peg-in-hole interaction: when there are multiple colliding points (red dots) during the collision, the combined stiffness could exceed the maximal stiffness that can be rendered by the hardware if the rendered penetration depth was evaluated by simply summing all the individual penetration depths (b). Alternative solutions are shown in (c~e). The shaded areas represent the visualized stiffness field, while the green circle represents the boundary of maximal renderable stiffness. By taking the average of all individual penetration depths (c), it is possible to bound the maximal combined stiffness, but the contribution of penetrations in some directions will be underestimated. Attaching a virtual coupling (d) between the virtual world and the controller can also bound the combined stiffness. However, the rendered force will be coupled to the local point density on a directional basis, indicating that the reaction force in some direction would be underestimated. For example, if a tool is rotated in a hole as (a), the reaction force would be dominated by points on the side walls, causing a horizontal motion and an easy penetration along y-axis.. The proposed solution (e) is to realize equal combined stiffness along the gradient of all colliding points while decoupling the rendered penetration depth from the local point density.

There existed several ways to "merge" the individual penetration depths, but none of them can be applied to our case directly. By summing all the individual penetration depth vector directly [2][6], the combined stiffness could rapidly build up when there are many colliding points (Fig. 2b), and lead to system instability [18]. Taking average of the individual penetration depth vectors, on the other hand, can solve the problem of excessive stiffness [19].

However, this method would still deteriorate the haptic feedback when distributed contact is involved. If most of the colliding points have the same or parallel gradients in their penetration depth vectors, the rendered force will be dominated by these points while the contribution of other colliding points is averaged out (Fig. 2c). This is not desired since the feedback would be coupled to the point density again.

Virtual coupling [18] was a common way to limit the excessive stiffness when there are multiple contact sites and multiple colliding points during the interaction [20]. However, virtual coupling only bounds the combined stiffness after the rendered penetration depth was determined, indicating that it does not decouple the rendered force from the density of point. As a result, if there are more colliding points that have similar or parallel gradients in their penetration depth vectors, they will dominate the render force (Fig. 2d).

The domination of the rendered force's direction by points that has similar gradients could lead to serious problem in interactions that involves multiple contact sites. Take the peg-in-hole task for an example, when the peg is rotated, since most colliding points would be on the side wall of the hole, the reaction force will mainly pointing horizontally, and will thus let the peg easily penetrate the upper corner of the hole (Fig. 2a).

In the field of computer graphic rendering, the simulation of distributed contact between objects is also a major challenge. Computer graphical rendering based method, e.g. [21], can theoretically resolve the problem of combined stiffness. In [21], the idea of "translational penetration depth" was proposed that the shortest distance for separating two overlapping objects was computed by taking all geometric features of both the end-effector and the environment into account. However, visual rendering techniques such like the translational penetration depth is computationally expensive, and thus has limited usefulness in the field of haptic rendering given the limitation of hardware capability.

So far, the methods mentioned above are all penalty-based, in which the end-effector is allowed for penetrating the target object. Another common approach in haptic rendering is the constraint-based method, in which a proxy of the end-effector is defined to be always constrained to the surface of the object while the actual position of the end-effector is inside the target object [13][17][22]. The force rendered to the user is then determined based on the distance between the position of the proxy and that of the actual end-effector serves.

Since the proxy is not a single point in our task, 3-DOF constraint-based is not applicable. Moreover, the use of 6-DOF constraint-based methods, such as [17][20], is not preferred. In fact, given the lack of torque feedback, applying a 6-DOF constraint-based method to our case is similar to finding the translational penetration depth [21], which would create a huge burden in terms of computational efficiency.

Here, we proposed a penalty-based method that could regulate the combined stiffness along the gradient of all penetration depth vectors to a desired value *K*, while decoupling the force's direction from point density (Fig. 2e). This method can also be thought as taking average of the individual penetration depths based on their gradients rather than the total number of colliding points.

2 Rendering Algorithm

In this work, we focused on the calculation of penetration depth and the rendered force in a way that the combined stiffness along all gradients can be regulated to a default stiffness *K*. The proposed method can be applied to different virtual models, such as voxel-based model, point-shell model, polygonal model, as long as the haptic rendering involves multiple colliding points.

In section 2.1 and 2.2, we will give brief introduction of the existing modeling methods on the platform used in this project, the Simodont® dental trainer, as a background. From section 2.3, we will give detailed derivations to illustrate the concept of the proposed rendering algorithm.

2.1 Objects Modeling

On the current platform, the end-effector, i.e. a dental drill, is modeled using a set of polynomial equations, and the surface of the environmental object is modeled with point-shell.

The point-shell was formed by applying marching cubes method [23] to the raw density field. The intersecting points between a surface and a cube are extracted and assigned to the point-shell. Depending on the relationship between the intersected surface and the cube, there could be up to 4 points, i.e. 4 intersecting points, contained by a single grid cell.

2.2 Collision Detection

The collision detection is conducted under a multi-rate paradigm. A bounding box surrounding the stylus of the dental drill was used as the range for searching potential colliding grid cells. In the slow updating loop, the 3D grid cells that locate inside the bounding box will be examined one by one. A grid cell will be added to the contact list as long as one of points it contains is found to be inside the tool's boundary.

In the fast updating loop, the grid cells on the candidate list will be carefully examined. Once a colliding point is detected, the system will calculate its corresponding penetration depth and gradient.

2.3 Individual Penetration Depth Vector

During the collision, every colliding point has a corresponding penetration depth vector. In the rest of this work, we will use $\vec{d}_i$ to represent the individual penetration depth vector that corresponds to the ith colliding point, $p_i,\ i \in N$. The magnitude of $\vec{d}_i$, i.e. the penetration depth, is denoted as d_i and the gradient of $\vec{d}_i$ is denoted as a unit vector, $\hat{u}_i$. We used the term "gradient" to represent the direction of the vector $\vec{d}_i$ since the corresponding reaction force would decrease along this direction.

The penetration depth d_i was determined by calculating the distance from the colliding point to the nearest point on the surface of the end-effector. The gradient $\hat{u}_i$ is defined by the inverse surface normal vector of the *ith* colliding point.

The use of the inverse surface normal as the gradient of penetration depth vector allowed the direction of $\vec{d}_i$ to depend on the surface geometry of the environmental objects, and it also prevented of the "pop through" effect when the colliding point came across the median line of the tool's geometry [3][19][20].

2.4 Combined Stiffness Shaping

Suppose there are **n** colliding points at certain time step, the system can be thought as a mass attached with **n** springs in various directions (Fig. 2). Each spring has a stiffness of *K*. Here we have assumed that angular velocity of the tool is constant throughout the whole time step based on the fact that the rotational movement in dental operation is always slow and gentle, and the interval of a time step on our platform is only 0.5 ms.

To evaluate the combined stiffness along $\hat{u}_i$, i.e. the gradient of the ith point, p_i, we have to calculate the reaction force along $\hat{u}_i$ by applying a infinitesimal displacement along $\hat{u}_i$. The total reaction force can be calculated as follow:

$$\vec{f}_{c,i} = \sum_{j=1}^{n} \vec{f}_j = -K \cdot \Delta x \cdot \sum_{j=1}^{n} \left\langle \hat{u}_i, \hat{u}_j \right\rangle \cdot \hat{u}_j, \quad \Delta x \to 0 \tag{1}$$

Since we are interested in the reaction force along $\hat{u}_i$, the reaction force in Eq. (1) must be projected to $\hat{u}_i$ so that:

$$\vec{f}^{*}_{c,i} = \left\langle \vec{f}_{c,i}, \hat{u}_i \right\rangle \cdot \hat{u}_i = -K \cdot \Delta x \cdot \hat{u}_i \cdot \sum_{j=1}^{n} \left\langle \hat{u}_i, \hat{u}_j \right\rangle^2 \tag{2}$$

By rearranging Eq. (2), we can obtain the combined stiffness along $\hat{u}_i$:

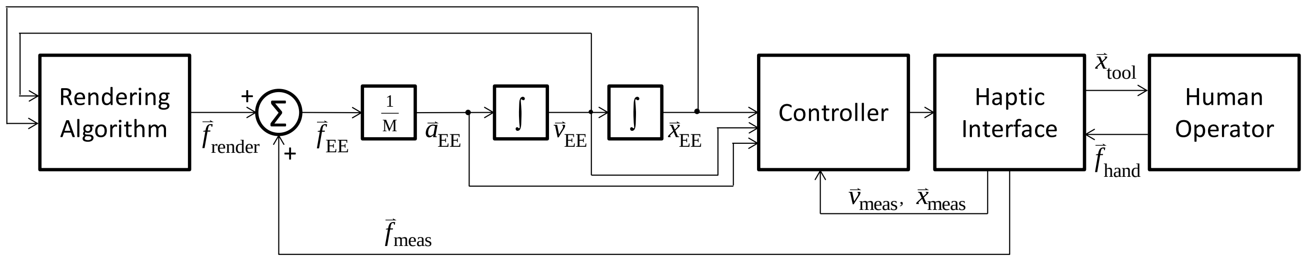


Figure 3. The structure overview of the platform used in this project. It is an admittance control haptic interface. The system measures the force applied by the user, denoted as f_{meas}, and sums this force with the rendered force, f_{render}, calculated in the virtual environment. The summation force, f_{EE}, is then applied to the end-effector to create movements. By limiting the input force with a pre-defined threshold, the end-effector's movement will also be limited. As a result, theoretically, the corresponding displacement of the end-effector will be zero when the input force is higher than the threshold, regardless of the force applied to the interface by the human operator.

$$K_{c,\text{i}} = \frac{\left\|\vec{f}^{\,*}_{c,i}\right\|}{\Delta x} = K \cdot \sum_{j=1}^{n} \left\langle \hat{u}_i, \hat{u}_j \right\rangle^2 = \rho_i \cdot K \tag{3}$$

The coefficient ρ_i is used to represent the ratio of the combined stiffness to the default stiffness, K. According to Eq. (3) , ρ_i will be no less than 1, and can have a maximal value of **n** when all the **n** springs have same gradients. Using the same approach, we can calculate the combined stiffness along all **n** directions. Based on Eq. (3), we can organize all stiffness ratio ρ_i in a matrix form as follow:

$$\boldsymbol{\rho}_{n\times 1} = \boldsymbol{A}_{n\times n} \cdot \mathbf{1}_{n\times 1}$$

$$\boldsymbol{\rho} = \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_n \end{bmatrix} \quad \boldsymbol{A} = \begin{bmatrix} \langle \hat{u}_1, \hat{u}_1 \rangle^2 & \cdots & \langle \hat{u}_1, \hat{u}_n \rangle^2 \\ \vdots & \ddots & \vdots \\ \langle \hat{u}_n, \hat{u}_1 \rangle^2 & \cdots & \langle \hat{u}_n, \hat{u}_n \rangle^2 \end{bmatrix} \quad \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

If the stiffness ratio is so high that the combined stiffness exceeds the maximal stiffness that can be rendered by the hardware, the system would become unstable [18]. In contrast, if ρ_i has a value of 1, the combined stiffness will have a desired value K. To realize this, we introduce a new parameter $\boldsymbol{c}$, called the stiffness scaling coefficient, to manipulate the stiffness of each spring so that the value of all ρ_i becomes 1.That is:

$$\rho_i^* = 1 = \sum_{j=1}^{n} c_j \cdot \left\langle \hat{u}_i, \hat{u}_j \right\rangle^2 \tag{4}$$

With Eq. (4), the matrix form can be rewritten as:

$$\boldsymbol{\rho}^*_{\text{n}\times 1} = \mathbf{1}_{\text{n}\times 1} = \boldsymbol{A}_{\text{n}\times\text{n}} \cdot \boldsymbol{c}_{\text{n}\times 1}$$

$$\boldsymbol{\rho}^* = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad \boldsymbol{A} = \begin{bmatrix} \langle \hat{u}_1, \hat{u}_1 \rangle^2 & \cdots & \langle \hat{u}_1, \hat{u}_n \rangle^2 \\ \vdots & \ddots & \vdots \\ \langle \hat{u}_n, \hat{u}_1 \rangle^2 & \cdots & \langle \hat{u}_n, \hat{u}_n \rangle^2 \end{bmatrix} \quad \boldsymbol{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_i \end{bmatrix}$$

If the matrix $\boldsymbol{A}$ is invertible, then the vector $\boldsymbol{c}$ will be equal to $\boldsymbol{A}^{-1}\mathbf{1}$. However, most of the time, $\boldsymbol{A}$ is non-singular and thus non-invertible. This indicates that $\boldsymbol{c}$ should be solved with linear programming methods, which could compromise the high computational efficiency of penalty-based rendering method. As such, rather than solving the vector $\boldsymbol{c}$ analytically, we proposed an iterative method to make all the stiffness ratio ρ sufficiently close to 1 by approximating the values in $\boldsymbol{c}$.

At each iteration, the stiffness ratio ρ is first evaluated using Eq. (4), and is then used to update the stiffness scaling coefficient c as follow:

$$\rho_i^{m+1} = \sum_{j=1}^{n} \text{c}_j^m \cdot \left\langle \hat{u}_i, \hat{u}_j \right\rangle^2$$

$$c_i^{m+1} = \frac{c_i^m}{\rho_i^{m+1}}$$

where m denotes for the iteration number.

2.5 Combined Damping Normalization

During a collision with multiple contact sites, calculate the damping force using the velocity of the end-effector along the direction of the rendered force is likely to result in an under-damped system (Appendix B.3.2).

In this work, the overall damping force is calculated in two steps. First, we scale the damping coefficient corresponding to each colliding point by the stiffness scaling coefficient, c, acquired in section 2.4. Second, we calculate the damping force of each colliding point, and sum them up to form the rendered damping force.

Here we use B_i to represent the damping coefficient corresponding to the ith colliding point. Given a default damping coefficient $\boldsymbol{B}$, the damping coefficient of the ith colliding point, B_i, would be:

$$B_i = c_\text{i}^* \cdot B \tag{5}$$

An important assumption underlying Eq.(5) is that the ith combined stiffness, $K_{\text{c,i}}$, is sufficiently close to K, i.e., ρ_i is sufficiently close to 1. If $K_{\text{c,i}}$ is way higher or lower than K, further regulation on the damping coefficient must be made to prevent over- and under-damping (see Appendix C.4.3).

2.6 Rendered Force Calculation

The rendered force consists of an elastic force from the virtual springs and a damping force. The elastic force, denoted as $\vec{f}_{\text{spring}}$, is calculate as:

$$\begin{aligned}\vec{f}_{\text{spring}} &= \sum_{i=1}^{n} \vec{f}_{\text{spring,i}} = -\sum_{i=1}^{n} K_{\text{i}} \cdot d_{\text{i}} \cdot \hat{u}_{\text{i}} \\ &= -\sum_{i=1}^{n} \left(c_{\text{i}}^{*} \cdot K\right) \cdot d_{\text{i}} \cdot \hat{u}_{\text{i}} \\ &= -K \cdot \sum_{i=1}^{n} c_{\text{i}}^{*} \cdot d_{\text{i}} \cdot \hat{u}_{\text{i}} = -K \cdot \vec{d}_{\text{render}}\end{aligned} \tag{6}$$

Eq. (6) shows that regulating the stiffness corresponding to each colliding point can also be thought as manipulating the individual penetration depth to realize a desired stiffness K along the gradients of all colliding points.

The damping force can be divided into two parts, including a normal damping force and a tangential damping force, denoted as $\vec{f}_{\text{damping,N}}$ and $\vec{f}_{\text{damping,T}}$, respectively. The calculation of normal damping force is straightforward, as follow:

$$\begin{aligned}\vec{f}_{\text{damping,N}} &= \sum_{i}^{n} \vec{f}_{\text{damping,N,i}} = -\sum_{i}^{n} B_{\text{i}} \cdot \langle \vec{v}_{\text{EE}}, \hat{u}_{\text{i}} \rangle \cdot \hat{u}_{\text{i}} \\ &= -\sum_{i}^{n} \left(c_{\text{i}}^{*} \cdot B\right) \cdot \langle \vec{v}_{\text{EE}}, \hat{u}_{\text{i}} \rangle \cdot \hat{u}_{\text{i}} \\ &= -B \cdot \sum_{i}^{n} c_{\text{i}}^{*} \cdot \langle \vec{v}_{\text{EE}}, \hat{u}_{\text{i}} \rangle \cdot \hat{u}_{\text{i}} = -B \cdot \vec{v}_{\text{normal}}\end{aligned}$$

where $\vec{v}_{EE}$ is the velocity of the end-effector. It is shown that the manipulated individual damping force can be thought as forming a "regulated" normal penetration velocity, $\vec{v}_{\text{normal}}$. With this normal penetration velocity, the tangential penetration velocity and the corresponding tangential penetration depth are calculated as:

$$\begin{aligned}\vec{v}_{\text{tangential}} &= \vec{v}_{\text{EE}} - \vec{v}_{\text{normal}} \\ \vec{f}_{\text{damping, t}} &= -\left(\mu_{\text{s}} + \mu_{\text{f}} \cdot \left\| \vec{f}_{\text{spring}} \right\|\right) \cdot \vec{v}_{\text{tangential}}\end{aligned}$$

where μ_s and μ_f are pre-defined coulomb friction coefficients. Finally, the rendered force can be expressed as the summation of all elements, as follow:

$$\vec{f}_{\text{render}} = \vec{f}_{\text{spring}} + \vec{f}_{\text{damping,N}} + \vec{f}_{\text{damping,T}}$$

2.7 Input Force Limiting

Theoretically, an admittance-control haptic interface should be able to render infinity stiffness since the displacement is controlled by the computer. In reality, the renderable stiffness depends on the hardware limitations such as mechanical compliance, maximal motor torque, etc. Here we propose a simple method to exploit the renderable stiffness of an admittance control system.

Normally, the measured force, $\vec{f}_{meas}$, is applied to the virtual object directly in an admittance control system (Fig. 3). In this work, we imposed a limitation on the magnitude of $\vec{f}_{meas}$ so that:

$$\vec{f}_{\text{meas}}^{\,*} = \begin{cases} \vec{f}_{\text{meas}}, & \left\| \vec{f}_{\text{meas}} \right\|_2 \leq f_{\text{th}} \\ \dfrac{\vec{f}_{\text{meas}}}{\left\| \vec{f}_{\text{meas}} \right\|_2} \cdot f_{\text{th}}, & \text{otherwise} \end{cases}$$

where $\vec{f}_{\text{meas}}^{*}$ is the actual force applied to the virtual object and f_{th} is the force limit.

When the tool and the environmental object are in contact, interpenetration would induce reaction force, $\vec{f}_{\text{render}}$, and would require more input force, $\vec{f}_{\text{meas}}$, to increase the interpenetration. Once the magnitude of $\vec{f}_{\text{meas}}$ reaches the threshold, f_{th}, the dynamics of the end-effector would be limited so that its velocity in the virtual environment can be expressed as:

$$\vec{v}_{EE} = \begin{cases} \vec{v}_{EE}, & \langle \vec{v}_{EE}, \nabla f_{\text{render}} \rangle \leq 0 \\ \vec{v}_{EE} - \langle \vec{v}_{EE}, \nabla f_{\text{render}} \rangle \cdot \dfrac{\nabla f_{\text{render}}}{\left\| \nabla f_{\text{render}} \right\|_2}, & \langle \vec{v}_{EE}, \nabla f_{\text{render}} \rangle > 0 \end{cases}$$

In other words, when the force limit is reached, the human operator can only move the end-effector in a direction that would decrease the reaction force from the object. If the operator keeps applying force toward the direction of ∇f_{render}, the end-effector will not move and thus realize a high perceived stiffness.

2.8 Gradient-based Point Reduction

During the collision, the number of colliding points is coupled to the surface area in contact. On our system, we found that in an extreme case such as peg-in-hole task, the number of colliding points could be up to almost a thousand. This made it impossible to complete the calculation of rendered force in one time step, i.e. 0.5ms. As such, a simple filter was implemented to extract the colliding points that contain most information of the collision.

The idea of this method is similar to direction clustering but is way simpler due to the limited computational power. We first compare the gradient of each colliding point by taking inner product. If there exists two colliding points whose gradients are close enough, i.e. the value of their inner-product is higher than a pre-defined threshold, the system will keep the point that has the larger penetration depth and remove the other one from the list. The default value of the inner-product threshold in this work is 0.995. Detailed steps of this point reduction method are illustrated in Appendix C.3.

3 RESULT

3.1 Hardware

All evaluation tests were conducted on the Simodont® dental trainer. The haptic interface of Simodont® is an admittance control device. It measures the force input by the human subject and render 3-DOF translational displacement feedback (Fig. 3). The default updating rate of the haptic simulation is 2048Hz. The interface is an asymmetric system that it has 6-DOF input (both position

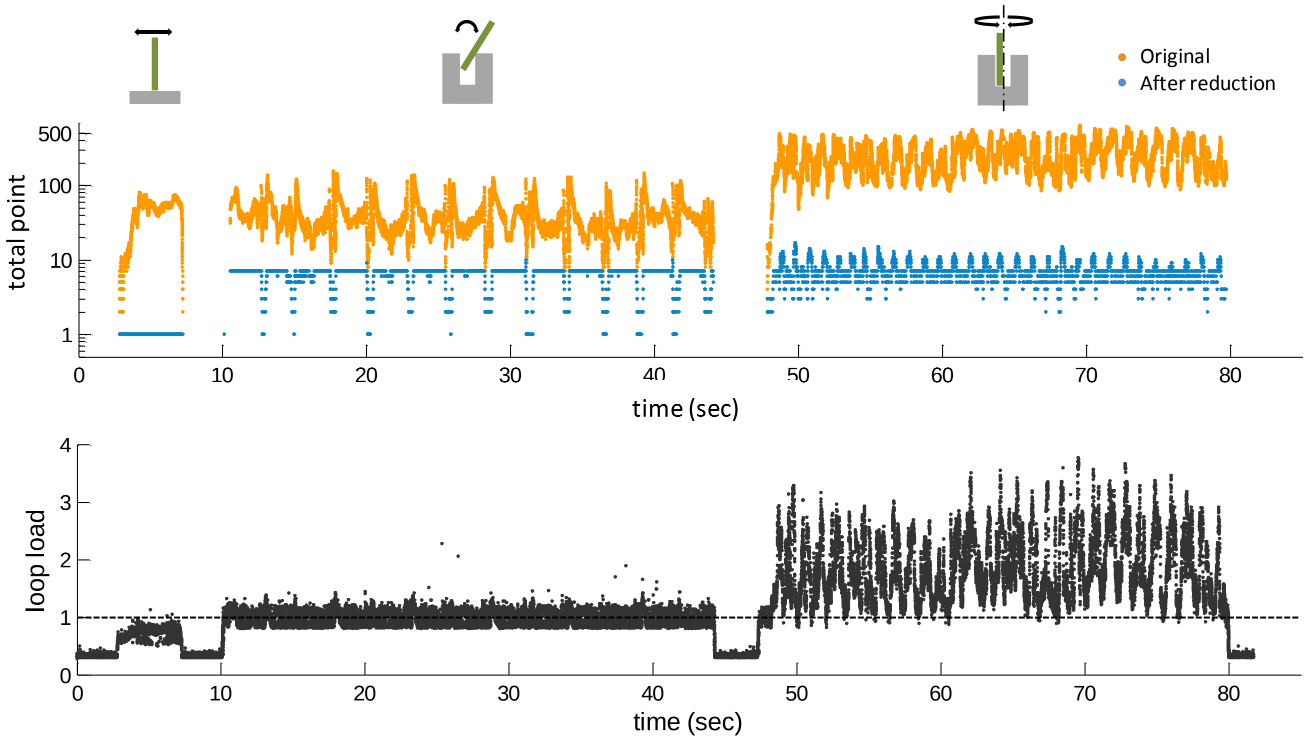


Figure 4. The total colliding point before (orange dots) and after (blue dots) the point reduction process, and the loop load (black dots) logged during the three types of evaluation tasks. In task 1, the human operator poked and dragged the dental drill against a flat surface (between 0~10s). In task 2, the dental drill was rotated with respect to a vector perpendicular to the median line of the hole (between 10~45s). In task 3, the operator performs a spin movement of the dental drill with respect to the median line of the hole (after 45s).

and orientation) and only 3-DOF actuator output. During the test, the human operator held the dummy dental drill as holding a pen to interact with virtual objects.

3.2 Evaluation Tasks

We used the peg-in-hole interaction to evaluate the performance of the proposed algorithm, similar to [17]. A 1.5cm×1.5cm×1cm virtual cuboid with a grid-cell size of 0.1mm×0.1mm×0.1mm was built to be the standard environment. A circular hole with a diameter of 2mm and a depth of 5mm was located at the middle of the cuboid's top surface. The end-effector was modeled as a dental drill, but only the stylus of the drill was haptically rendered. The cylindrical stylus had a diameter of 1mm and a length of 10mm. The evaluation consisted of three subtasks (Fig. 4):

- *Task 1:* Poke and drag the drill on a flat surface for 5s.
- *Task 2:* Rotate the drill with respect to a vector perpendicular to the orientation of the hole. The drill and the hole must be in contact throughout the sub-task, which lasted for 30s.
- *Task 3:* Spin the drill with respect to the orientation of the hole, i.e., the tool's orientation was constant throughout the sub-task, which lasted for 30s.

3.3 Gradient-Based Point Reduction

In task 1, the original number of colliding points (Fig. 4, orange dots) were 44.8 ± 19.9 (MEAN ± STD). After the point reduction, the total number of colliding point (Fig. 4, blue dots) was always equal to 1 since all colliding points had the same gradient (Fig. 3, upper left). In task 2, the original and the reduced number of colliding points were 37.8 ± 17.7 and 6.21 ± 1.7, respectively (Fig. 3, upper middle). In task 3, the original and reduced number of colliding points were 231.5 ± 116.1 and 6.5 ± 2.1, respectively (Fig. 3, upper right).

3.4 Combined Stiffness

In section 2.4, we proposed an iterative approximation method to regulate the combined stiffness along all gradients. We introduced a stiffness ratio ρ_i to represent the ratio of regulated combined stiffness along the ith gradient to the desired stiffness K.

To evaluate the combined stiffness as a function of the iteration number, we logged the maximal and minimal stiffness ratio among all gradients in every updating cycle. Ten sets of stiffness ratios were logged, at the iteration number of 1, 2, 3, 4, 5, 10, 20, 40, 80, and 160, respectively.

The result shows that, both the maximal and minimal stiffness ratio converged quickly to 1 in several iterations (Fig. 5). The iteration had small impact on the loop load since the number of colliding points (and thus the number of stiffness ratios) was below 15 even in extreme contact scenario such as task 3.

3.5 Limited Input Force

In section 2.7, we introduced a method to exploit the high renderable stiffness of an admittance control inter-

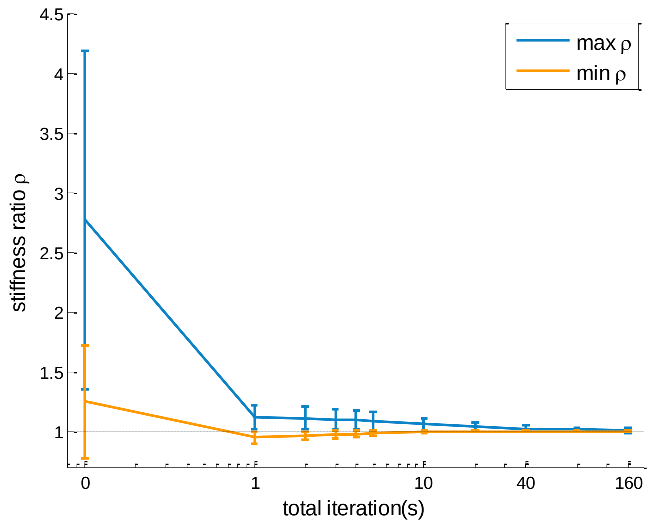


Figure 5. The maximal (blue) and minimal (orange) combined stiffness of all colliding points in each time step was logged in real-time throughout the three tasks. It is clear that the stiffness ratio converged to 1 quickly in several iterations. In this work, we set the default iteration number to be 3.

face, by saturating the input force with a pre-defined threshold, f_{th}. To evaluate its influence on the rendered stiffness, a separate evaluation task was conducted in which the orientation of the dental drill was fixed to be perpendicular to a flat surface. The drill was pressed directly against the flat surface while the corresponding measured force and the model position of the end-effector were logged in real-time.

The result showed that, without the input force limit, the displacement of the end-effector in the virtual world was more or less proportional to the measured force, i.e. the input force (Fig. 6, orange dots). With the force limiting algorithm, the displacement of the end-effector stayed nearly constant regardless of the input force (Fig. 6, blue dots) when the input force was beyond the threshold.

The force threshold was set to be 2N in this demonstration. In Fig. 6, the limitation started from 2.7N since the measured force was a summation of the hand force applied by the human operator and a force offset of 0.7N to compensate for the weight of the dummy drill on the haptic interface.

3.6 Trajectory Evaluation

One of the major goals of this work was to use motion to replace the need of torque during tasks such as peg-in-hole interaction. In reality, if there is no friction on the surface inside the hole and the peg is always in contact with the hole, a rotated tool would slide out of the hole along a certain trajectory (Fig. 7, orange line).

As such, we compared the resultant trajectory of the end-effector in task 2 with the ideal trajectory computed based on the geometrical constraint between the tool and the hole. The ideal trajectory also served as an implicit surface of the collision. Therefore, we could also use the ideal trajectory to evaluate to what extend the interpenetration between the drill and the hole was as the drill was rotated (Fig. 7a).

To evaluate if the measured end-effector's position was correctly determined based on the changing contacting configuration between the tool and the hole due to rotational movement, we analyzed the tool tip's height (with respect to the bottom of the hole) as a function of drill's tilted angle (Fig. 7b), denoted as α (rad). A α with a value of 0 means that the tool's orientation is parallel to that of the hole, while a value of $\pi/2$ means that the tool is perpendicular to the hole. We also analyzed the distance between the end-effector and the median line of the hole as a function of the drill's tilted angle (Fig. 7c). Based on the geometry of the hole and drill, we only analyze the trajectory of the end-effector corresponding to α in a range from 0 to 1.047 since the drill would no longer be in touch with the side wall of the hole with α larger than 1.047.

During dental operations, the movement made by the dentist is gentle and the force applied to the tooth is normally under 2N. Therefore, we categorized the logged position into two parts, one part corresponding to low input force from the human operator (<2N), and one part corresponding to force higher than 2N.

The result shows that, when the force applied by the human operator against the hole was lower than 2N, the trajectory of the end-effector followed the trajectory pretty well (Fig. 7, green dots). With a relative higher input force (> 2N), the position of the end-effector had a higher

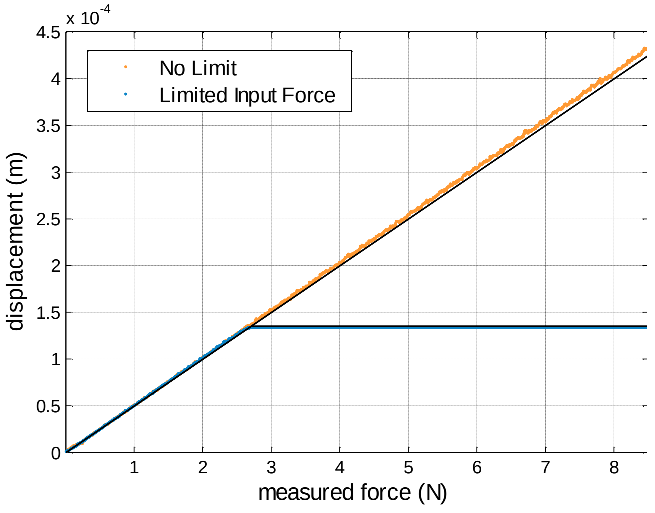


Figure 6. The displacement of the end-effector with respect to the penetrated object's surface as a function of the measured force. Without limiting the force (orange dots), the displacement is nearly proportional to the measured force. With the input force limited (blue dots), the displacement stops increasing and stays constant once the measured force reached the pre-defined threshold, 2.0N. The black line shows the ideal trend in both situations. In this figure, the measured force is not limited until it reaches 2.7N since the measured force is a summation of force applied by the human operator and a force offset of 0.7N used to compensate for the dummy tool's weight.

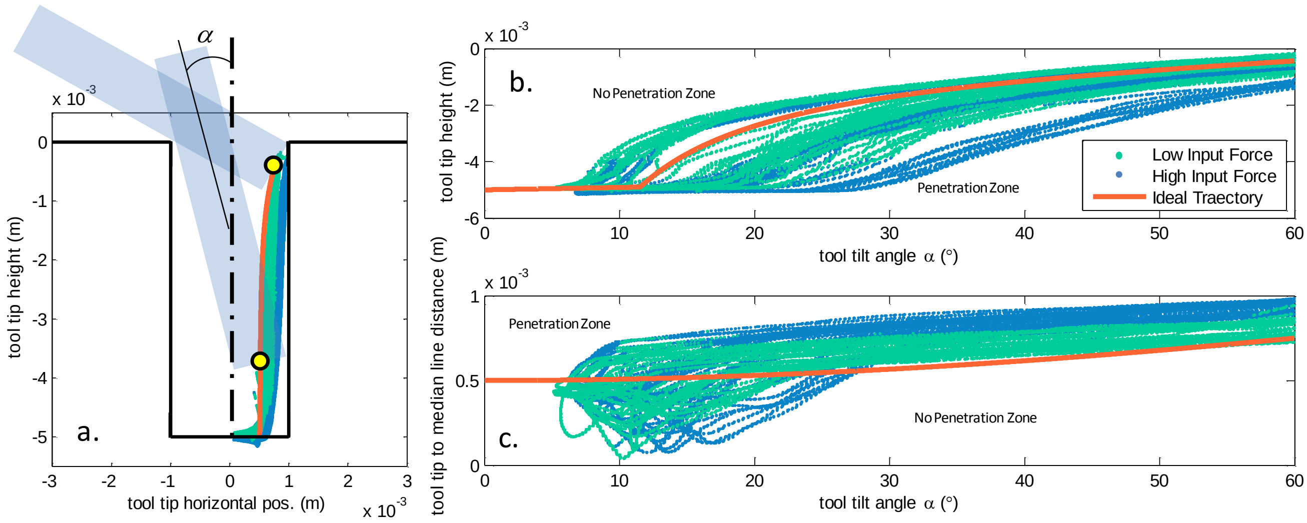


Figure 7. The logged position of the end-effector (yellow) during task 2. We calculate the ideal trajectory (orange) of the end-effector based on the geometrical constraint of the hole and the tool. (a) The trajectory of the actual end-effector emulates the ideal one. (b) The end-effector's height (with respect to the bottom of the hole) as a function of drill's tilted angle, denoted as α (rad). (c) The distance between the end-effector and the median line of the hole as a function of the drill's tilted angle. The ideal trajectory (orange) can be regarded as an implicit surface for the interaction. Based on this implicit surface, we could divide the area in the figures into "penetration zone" and "no penetration zone". Apparently, when the input force is low (<2N), the end-effector would track the ideal trajectory well (green dots) and stay close to the "no penetration zone". When the input force is higher than 2N, offset in the tool tip's height had started to show up. Despite the error in the height, the error in the distance to the median line was almost negligible (around 0.25mm).

possibility to fall "inside" the implicit surface. Thanks to the input force limitation method (section 2.7), the penetration depth was kept small with a maximal magnitude around 0.25 mm.

4 Discussion

4.1 Gradient-Based Point Reduction

The point reduction method proposed in this work combined penetration depths that have similar gradients. The idea is similar to the well-known direction clustering algorithm used in data analysis or pattern recognition. However, the direction clustering methods are computationally expensive and thus not suitable for our case.

In 1-DOF interaction, all colliding points have the same gradient. In such case, our point reduction method is identical to extracting the individual penetration depth that has the largest magnitude, similar to what was done in [3]. In [3], the magnitude of the rendered force totally depend on the highest penetration depth while the direction of the rendered force was the average of all gradients. However, their approach could lead to serious vibration during tasks such as the peg-in-hole, since the direction of the averaged gradient could change rapidly when gradients in every direction cancelling out each other.

In contrast, our point reduction method, rather than extracting the highest penetration depth among all colliding points, extracts the highest penetration depth among colliding points that have similar gradients. As a result, those colliding points kept in the list after the reduction process will contain most local information about the collision along their corresponding gradients.

The potential risk of this method is over-reduction due to the "winner takes all" rule used for keeping and eliminating points. If the point density is very high and both the penetration depth and gradient change gradually from one point to another, there could be only one point left in the end of process. The information stored in this remaining point will not be able to well represent the collision status, and could result in strange feedbacks. A more appropriate mechanism for integrating information from different points could be one of the important tasks in the future work.

4.2 Individual Penetration Depth

In this work, we took advantage of the high symmetry of the dental drill's geometry that we modeled the drill with polynomial set. The individual penetration depths were evaluated by calculating the shortest distance between the colliding point and a certain constraint described by a polynomial equation.

However, the use of shortest distance could create problems such as the famous "pop through" effect when the colliding point came across the median line of the tool's geometry [13]. Although we had used the inverse of surface normal corresponding to each colliding point as the gradient, as what was done in [3][19][20], the magnitude of the penetration depth still suffered from under estimation (Fig. 8).

Calculating the distance from the colliding point to the tool's surface along the inverse of surface normal, on the other hand, would consume a huge amount of computational power, especially when the tool was modeled with high order polynomial equations. Visual rendering method for finding such distance could be a potential solution, but it will require the tool to be discretized, e.g. the voxel-based model, which is not our best interest at this moment.

Fortunately, with the input force limiting method proposed in Section 2.7, the individual penetration depths were kept small during interactions (less than 0.25mm), and thus the error due to underestimation is negligible on our system.

4.3 Stiffness Shaping

On the current platform, theoretically, the maximal virtual stiffness for stable dynamic simulation with the numerical integrator is around 430,000 (N/m) (Appendix. A3). If the dynamics of the hardware is also taken into account, the maximal virtual stiffness that can be rendered without saturating the actuators would be around 80,000 (N/m). In all demonstrations, the default virtual stiffness, K, is 20,000 (N/m), indicating that the maximal stiffness ratio must not be higher than 4. Apparently, even with only one iteration performed, the shaped combined stiffness would never go beyond 1.5 (Fig. 5), suggesting a stable simulation in the dynamics of the virtual tool.

Although the shaped combined stiffness is always in a safe range, we would still want it to be as close to K as possible. As what was mentioned in Section 2.5, a stiffness ratio close to 1 allows us to calculate the individual damping coefficient, B_i, with Eq. (5) without the side-effect of over-damping or under-damping. With this in mind, we had assigned a default number of iteration number of 3.

On the other hand, an important assumption of the stiffness shaping method is that the angular movement of the tool can be regarded as static based on the fact that the angular velocity of the tool is sufficiently low in dental operations, and the high updating rate of the current system (2048Hz). However, to make the stiffness shaping algorithm more general, we must consider the case in which the tool has high angular velocity. In [24], the partial derivative of the reaction force with respect to both the translational and rotational movements were calculated for designing the virtual coupling to bound the overall combined stiffness. Although virtual coupling does not fit the requirement in our case, it would be interesting to investigate the impact of angular velocity on the combined stiffness by including the partial derivative of the individual penetration depth with respect to the angular velocity in the future work.

4.4 Damping Calculation

Similar to the stiffness shaping, the damping coefficient was calculated without taking the angular velocity into account. This could be problematic when the tool is rotated with respect to the end-effector in the virtual world while the end-effector has no translational velocity. A collision caused by such pure rotational movement could lead to under-damped feedbacks since the damping force

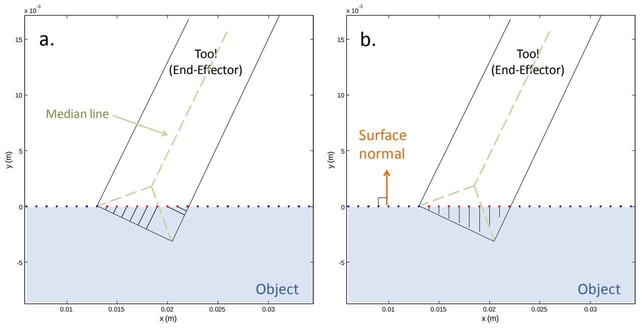


Figure 8. (a) In conventional shortest distance method, the individual penetration depth was the distance between the colliding point and the nearest point on the tool's surface, and the gradient was the unit vector pointing from the point to the surface. The gradient would flip suddenly when the colliding point crossed the median line of the tool (green). (b) In this work, the penetration depth was calculated as (a) but the gradient was the inverse of the surface normal (orange) corresponding to the colliding point. As a result, the lengths of individual penetration depths were always underestimated.

is solely determined by the translational velocity.

On current system, we did not observe under-damped behavior, presumably due to the slow rotational movement during the task.

4.5 Input Force Limiting

The limitation of the input force allows us to exploit the high renderable stiffness of an admittance control haptic interface. When the input force had saturated, the total force applied on the end-effector in the virtual environment would have a magnitude close to zero and thus the end-effector would no longer move. In other words, the rendered stiffness would totally depend on the hardware properties, such as maximal output power of the actuators, mechanical compliance of the interface, or the manually defined limitations such as maximal system current, maximal force output etc. Despite the limitations in hardware, the finite rendered stiffness is still sufficiently high to give the human operator an impression of interacting with a hard object.

A side effect of this method is that, if the input force saturated when the tool is manipulated with fast acceleration in the free air, such as shaking the dummy tool back-and-forth rapidly, the operator would feel an unnatural dragging force or an increased inertia.

This, however, is not a problem in our case. First, one seldom shakes a dental drill violently during the delicate dental operation. Second, even if one does shake the tool, the resultant input force observed on our system is still no more than 2N. Therefore, we set the maximal allowable input force to be 2N in all evaluation tasks.

4.6 Inherent Limitation in Applications

The proposed algorithm serves as an alternative solution to render simple object-to-object interaction other than

using costly 6-DOF haptic interfaces. However, the lack of torque feedback inherently limits its applicability in tasks that requires real torque feedback.
For example, this algorithm will not work when the tool is rotated in between two parallel surfaces since all gradients are parallel, either in the same direction or in opposite direction. Consequently, the human operator will not encounter reaction force by performing pure rotational movement on the tool. That is, the tool will be able to penetrate the surface of the two opposite walls freely without any corresponding feedback, which would deteriorate the fidelity of both visual and haptic feedback. In such an application, real torque feedback must be available on the interface in order to create a realistic feedback to the user.

In dental operation, the aforementioned situation is not likely to happen since the depth of the drilled hole should always be shorter than the length of the drill, indicating that the drill cannot be fully inside the hole.

5 Conclusion

In this work, we propose a method to realize the object-to-object interaction with 3-DOF haptic interfaces. We regulate the directional combined stiffness along the gradient of all colliding points to a desired stiffness *K* with an iterative approximation method. Regulating the combined stiffness on a directional basis has following benefits:

- It can bound the rendered stiffness to prevent divergence in dynamics simulation.
- It can decouple the magnitude and direction of the rendered force from the local point density of each contact area, i.e. the penetration depth along each direction will have equal weighting.
- It can generate a translational movement that closely related to geometrical constraints of the virtual objects, and use this movement as an alternative feedback to compensate for the lack of torque feedbacks during collisions that involved multiple contact sites.

We also propose a gradient-based point reduction method to keep the most information of the collision on a directional basis with the least number of colliding points. It has been shown that this method could largely improve the system's performance in terms of computational efficiency. Moreover, we achieve high perceived stiffness with our admittance control haptic interface by introducing a limitation on the input force applied to the virtual world. Last but not least, we demonstrate that, by using our algorithm, the resultant movement of the tool during a peg-in-hole task would be close to that of a frictionless contact in real peg-in-hole interactions.

References

[1] C. Basdogan, "Virtual environments for medical training: graphical and haptic simulation of laparoscopic common bile duct exploration," *Mechatronics, IEEE/ASME ...*, vol. 6, no. 3, pp. 269–285, 2001.

[2] A. Petersik, B. Pflesser, and U. Tiede, "Realistic haptic interaction in volume sculpting for surgery simulation," *Surg. Simul. ...*, vol. m, pp. 194–202, 2003.

[3] D. Wang and Y. Zhang, "Development of dental training system with haptic display," *Robot Hum. ...*, pp. 159–164, 2003.

[4] D. Morris, C. Sewell, and N. Blevins, "A collaborative virtual environment for the simulation of temporal bone surgery," ... *Image Comput. ...*, pp. 319–327, 2004.

[5] M. Eriksson, "A haptic and virtual reality skull bone surgery simulator," ... *World Haptics*, 2005.

[6] Y. Liu and S. Laycock, "A Haptic System for Drilling into Volume Data with Polygonal Tools.," *TPCG*, vol. D, 2009.

[7] T. T. II, D. Johnson, and E. Cohen, "Direct haptic rendering of sculptured models," *Proc. 1997 ...*, no. Figure 1, pp. 1–10, 1997.

[8] K. Pekkan, B. Whited, K. Kanter, S. Sharma, D. de Zelicourt, K. Sundareswaran, D. Frakes, J. Rossignac, and A. P. Yoganathan, "Patient-specific surgical planning and hemodynamic computational fluid dynamics optimization through free-form haptic anatomy editing tool (SURGEM).," *Med. Biol. Eng. Comput.*, vol. 46, no. 11, pp. 1139–52, Nov. 2008.

[9] F. Dachille IX, H. Qin, and a. Kaufman, "A novel haptics-based interface and sculpting system for physics-based geometric design," *Comput. Des.*, vol. 33, no. 5, pp. 403–420, Apr. 2001.

[10] R. Van der Linde and P. Lammertse, "The HapticMaster, a new high-performance haptic interface," *Proc. ...*, pp. 1–5, 2002.

[11] T. H. Massie and J. K. Salisbury, "The PHANToM Haptic Interface : A Device for Probing Virtual Objects 3 . Three Enabling Observations 4 . Three Necessary Criteria for an Effective Interface," vol. 55, 1994.

[12] F. Barbagli and K. Salisbury, "The Effect of Sensor / Actuator Asymmetries in Haptic Interfaces."

[13] C. B. Zilles and J. K. Salisbury, "A constraint-based god-object method for haptic display," *Proc. 1995 IEEE/RSJ Int. Conf. Intell. Robot. Syst. Hum. Robot Interact. Coop. Robot.*, vol. 3, pp. 146–151, 1995.

[14] F. Ryden and H. Chizeck, "A proxy method for real-time 3-DOF haptic rendering of streaming point cloud data," vol. 6, no. 3, pp. 257–267, 2013.

[15] J. Wu, G. Yu, and D. Wang, "Voxel-based interactive haptic simulation of dental drilling," *ASME 2009 ...*, 2009.

[16] D. Wang, Y. Zhang, Y. Wang, P. Lü, R. Zhou, and W. Zhou, "Haptic rendering for dental training system," *Sci. China Ser. F Inf. Sci.*, vol. 52, no. 3, pp. 529–546, Mar. 2009.

[17] M. Ortega, S. Redon, and S. Coquillart, "A Six Degree-of-Freedom God-Object Method for Haptic Display of Rigid Bodies," *IEEE Virtual Real. Conf. (VR 2006)*, pp. 191–198, 2006.

[18] J. Colgate and G. Schenkel, “Passivity of a class of sampled-data systems: Application to haptic interfaces,” *Am. Control Conf. 1994*, pp. 3236–3240, 1994.

[19] W. a. McNeely, K. D. Puterbaugh, and J. J. Troy, “Six degree-of-freedom haptic rendering using voxel sampling,” *Proc. 26th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH ’99*, pp. 401–408, 1999.

[20] J. Barbic and D. James, “Six-dof haptic rendering of contact between geometrically complex reduced deformable models,” *Haptics, IEEE Trans.*, vol. 1, no. 1, pp. 39–52, 2008.

[21] Y. J. Kim, M. a. Otaduy, M. C. Lin, and D. Manocha, “Fast penetration depth computation for physically-based animation,” *Proc. 2002 ACM SIGGRAPH/Eurographics Symp. Comput. Animat. - SCA ’02*, p. 23, 2002.

[22] D. C. Ruspini, K. Kolarov, and O. Khatib, “The haptic display of complex graphical environments,” *Proc. 24th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH ’97*, pp. 345–352, 1997.

[23] W. Lorensen and H. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *ACM Siggraph Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.

[24] J. Barbic, “Real-time reduced large-deformation models and distributed contact for computer graphics and haptics,” 2007.

Appendices belonging to the Master's thesis:

Rendering 6-DOF Object-to-Object Interaction with 3-DOF Haptic Interfaces

T. C. Lee

Appendix A – Simulation in MATLAB

A considerable part of this project was done in a well-controlled environment built in MATLAB (The MathWorks, USA). This appendix provides the detail information of each simulation the author have built in this project. Appendix A.1 introduces the peg-in-hole simulation that used to test and validate rendering algorithm. Appendix A.2 shows a simplified environment built to test the stiffness shaping algorithm. In Appendix A.3, we demonstrate how to determine the maximal renderable virtual stiffness in numerical simulation.

A.1 Peg-In-Hole Simulation

Undesired system behaviors such as vibrations and sudden repulsive force were observed on Simodont® in a collision configuration that involves reaction torque. A good example of such collision is the peg-in-hole configuration. During virtual dental training procedure, the human operator would have to remove the corrosive area of the virtual tooth with dental drills or milling tools. The drilled hole will inevitably create a typical peg-in-hole configuration, which would be likely to fail the rendering algorithm.

There are various potential causes to the undesired behavior. To investigate the potential problems underlying the rendering algorithm, we needed a well-controlled environment that excludes the influence by the hardware, such as computational speed, controller bandwidth, sensor noise, etc.

A.1.1 Setup

A simple two-dimensional environment was built, on which a virtual hole was expressed with point-shell (Fig. A.1 blue dots) and a virtual tool with polynomial equation sets (Fig. A.1 black). Each blue point has its own surface normal vectors. During the simulation, the points that fell into the boundary of the tool were treated as the colliding points and marked in red color. The center-of-mass (COM) of the tool (i.e. the position of the end-effector) was located at the center of the tool's round tip. The magnitude and direction of the rendered force applied to the end-effector was visualized using a cyan line segment. The parameters and settings are listed in Table A.1.

Table A.1. Parameters in the peg-in-hole simulation

Parameter and setting name	Default Value	Unit
Stiffness (for penalty-based rendering)	20000	[N/m]
Relative damping (for penalty-based rendering)	0.7	
Mass of the end-effector (i.e. tool)	0.2	[Kg]
Tool diameter	1	[mm]
Tool length	5	[mm]
Hole width	1	[mm]
Point-to-point distance of the point-shell	0.1	[mm]
Simulation step	0.0005	[s]

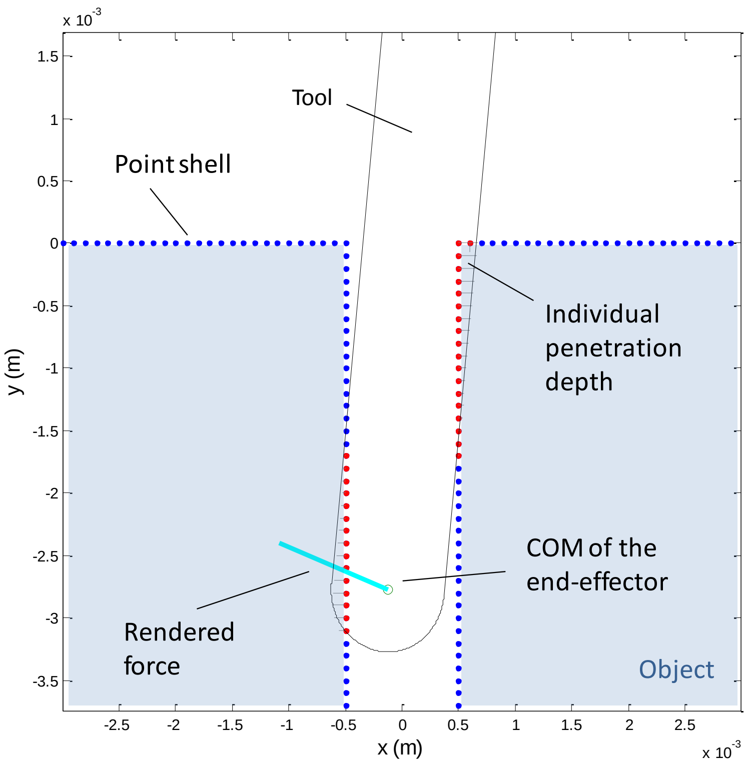


Figure A.1 The visualized peg-in-hole interaction in simulation. The object was expressed with point shell (blue dot) and the tool was expressed with polynomial equation set (black). The points fell inside the boundary of the tool were marked with red color. Moreover, the individual penetration depth (short line segment attached to the red dots) and rendered force (cyan line) were also visualized.

A.1.2 Dynamic Simulation

In order to simulate a system without reaction torque, the dynamic of the tool was based on the rendered force only. The torque term was simply set to be zero. The orientation of the tool was assigned according to a pre-determined profile. In other words, the tool's orientation is independent to the interaction between the tool and the object.

In each simulation step, the rendered force was first calculated using a certain type of rendering algorithm. Based on the rendered force, the new acceleration, velocity and position of the end-effector was calculated using the "leap-frog" integrator:

$$\begin{aligned} a[\mathrm{k}] &= F_{\mathrm{render}}[\mathrm{k}] / \mathrm{m}_{\mathrm{EE}} \\ v[\mathrm{k}] &= v[\mathrm{k}\text{-}1] + a[\mathrm{k}] \cdot \Delta t \\ p[\mathrm{k}] &= p[\mathrm{k}\text{-}1] + v[\mathrm{k}] \cdot \Delta t \end{aligned} \tag{A.1}$$

The advantage of the leap-frog integrator is its passivity in discrete system [1]. This integrator is also used to estimate the velocity and position command on Simodont®. An example of dynamics simulation of peg-in-hole interaction using the original algorithm (Fig. A.2a) and the proposed algorithm (Fig. A.2b) is shown below.

The default number of simulation step was 2000, but the simulation would be terminated automatically when following incidents occurred: 1) the end-effector had left the hole, i.e. its position along y-axis had become positive; 2) the orientation of the tool had become horizontal, i.e. parallel to the x-axis.

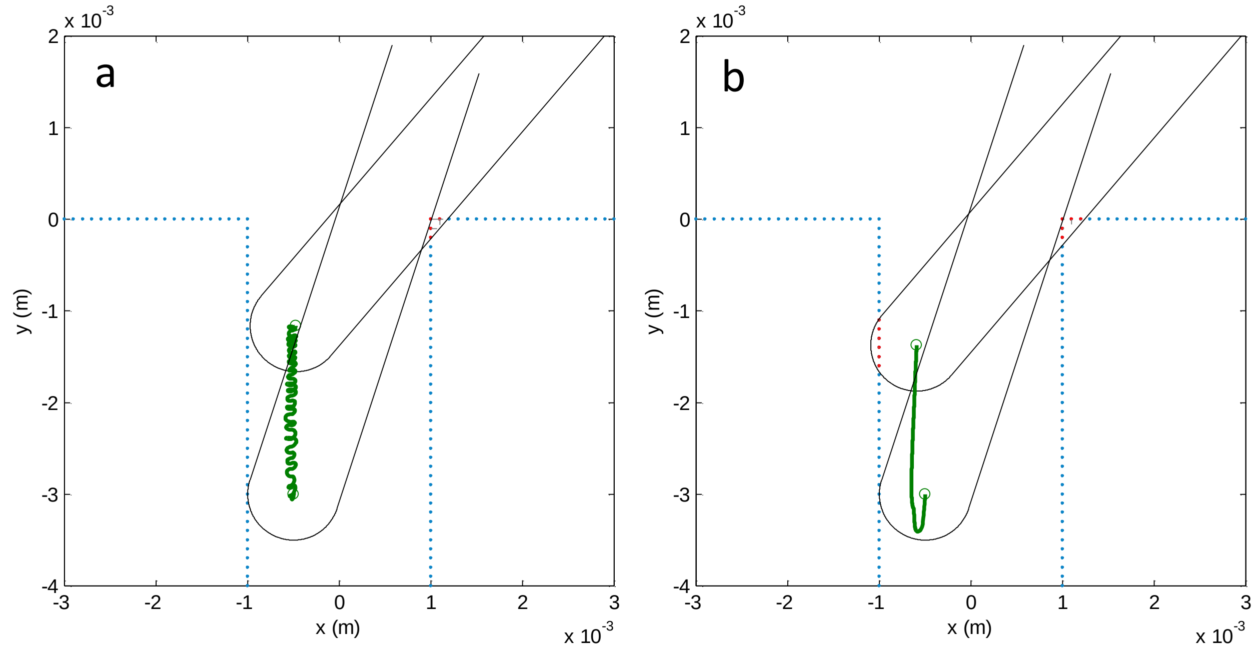


Figure A.2. Peg-in-hole simulations were conducted in a controlled environment built in MATLAB. The peg (black) was assigned with a constant angular velocity and an initial tilted angle. According to the logged trajectories (green lines), (a) the original rendering algorithm would give an obvious vibration while (b) the proposed rendering algorithm was capable of generating a smooth motion during the interaction.

A.2 Multiple Spring-Damper-Pairs Simulation

When there are multiple colliding points during collision (Fig. A.1), the easiest way to calculate the rendered force is to sum up the reaction force corresponding to each point. This, however, would increase the total rendered stiffness and damping (Fig. A.3). For a system with **n** colliding points, both the combined stiffness and combined damping could be at most **n** times higher than the default value **K**.

Given the discrete nature of a digital system, excessive stiffness and damping could cause system instability and thus a divergence in simulated motion (more details in Section A.3). Even if the combined stiffness and damping factors are in a range, the system could still be over-damped or under-damped due to the non-linear relationship between stiffness and damping (more details in Appendix C.4.3), which is not desired in a haptic system since it could deteriorate the fidelity of the haptic feedback.

This controlled environment was built to investigate the dynamics of a virtual mass when there are multiple spring-damping pairs attached to it in multiple directions. It was also used to test and compare each proposed or already existed haptic rendering algorithms.

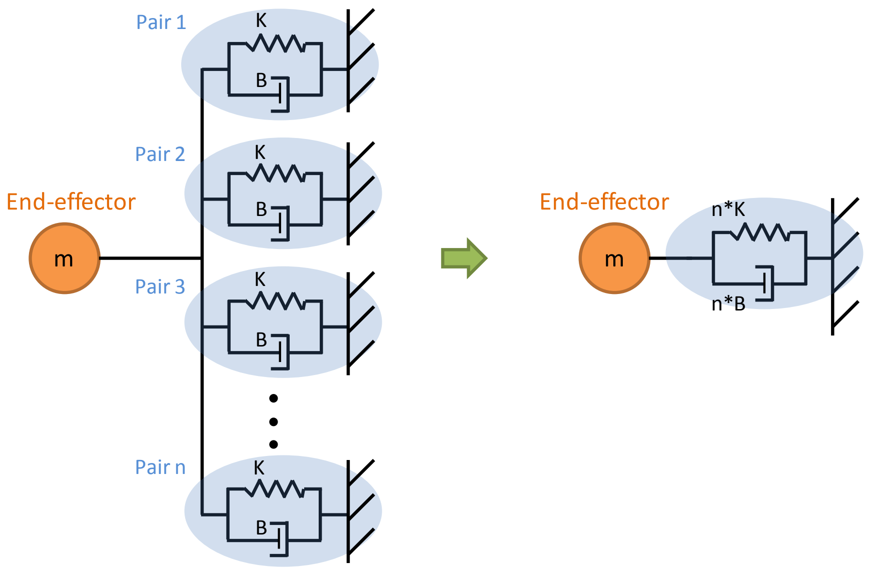


Figure A.3. In 3-DOF haptic rendering, when there are multiple colliding points during the collision, it can be thought of attaching multiple spring-damper pairs (blue) to a single virtual mass (orange). If there are **n** colliding points, the combined stiffness and damping can be at most **n** times higher than the default value.

A.2.1 Setup

A two-dimensional (2D) environment was built to simulate the interaction between a virtual point mass and multiple spring-damping pairs. This simulation is similar to the one introduced in section A.1, but in this case the end-effector is treated as a point without volume for simplicity's sake, as what was done in most conventional 3-DOF haptic rendering algorithms.

There are several parameters that can be set by the user (Table A.2). By changing those parameters, we can investigate to what extend each proposed rendering algorithm would improve or deteriorate the stability of the simulation. Also, we can have a direct impression about how the default limitations such as the maximal model force, maximal damping coefficient, etc., would impact the overall system behavior.

Table A.2. Parameters in the multiple spring-damper-pair simulation

Parameter and setting name	Default Value	Unit
Stiffness of each spring	20000	[N/m]
Relative damping of each spring-damper pair	0.7	
Mass of the end-effector	0.2	[Kg]
Number of opposite spring-damper pairs	User input	
Number of vertical spring-damper pairs	User input	
Number of horizontal spring-damper pairs	User input	
Number of spring-damper with random direction	User input	
Max damping coefficient	100	
Max magnitude of the rendered force	30	[N]
Simulation step	0.0005	[s]

Example

This environment served as a “test field” for immature algorithms. Thus, no meaningful result can be shown in this section. Nonetheless, we showed an example testing data of the investigation of the influence the limited damping coefficient and limited reaction force would bring to the system. We supposed that every spring would have its natural length when end-effector is at the origin, (0, 0). The initial conditions and parameters are shown in Table A.3.

The result shows that, without the limitation on damping coefficient and model force, despite the difference in dynamics, every algorithm had successfully moved the end-effector to the balancing point, i.e. the origin, with smooth trajectory. In contrast, if the damping and model force are limited, some rendering algorithms that did not properly handle the combined stiffness and combined damping would suffer from under-damping and fluctuation, while those had handle the combined properties well maintained the same performance (Fig. A.4).

Table A.3. Parameters in the demonstration

Parameter and setting name	Default Value	Unit
Stiffness of each spring	20000	[N/m]
Relative damping of each spring-damper pair	0.7	
Mass of the end-effector	0.2	[Kg]
Number of opposite spring-damper pairs (horizontal)	100	
Number of vertical spring-damper pairs (upward)	10	
Initial position of the end-effector	(-0.005, -0.005)	[m]
Initial velocity of the end-effector	0	[m/s]
Max damping coefficient (if applicable)	100	[Ns/m]
Max magnitude of the rendered force (if applicable)	30	[N]
Simulation step	0.0005	[s]

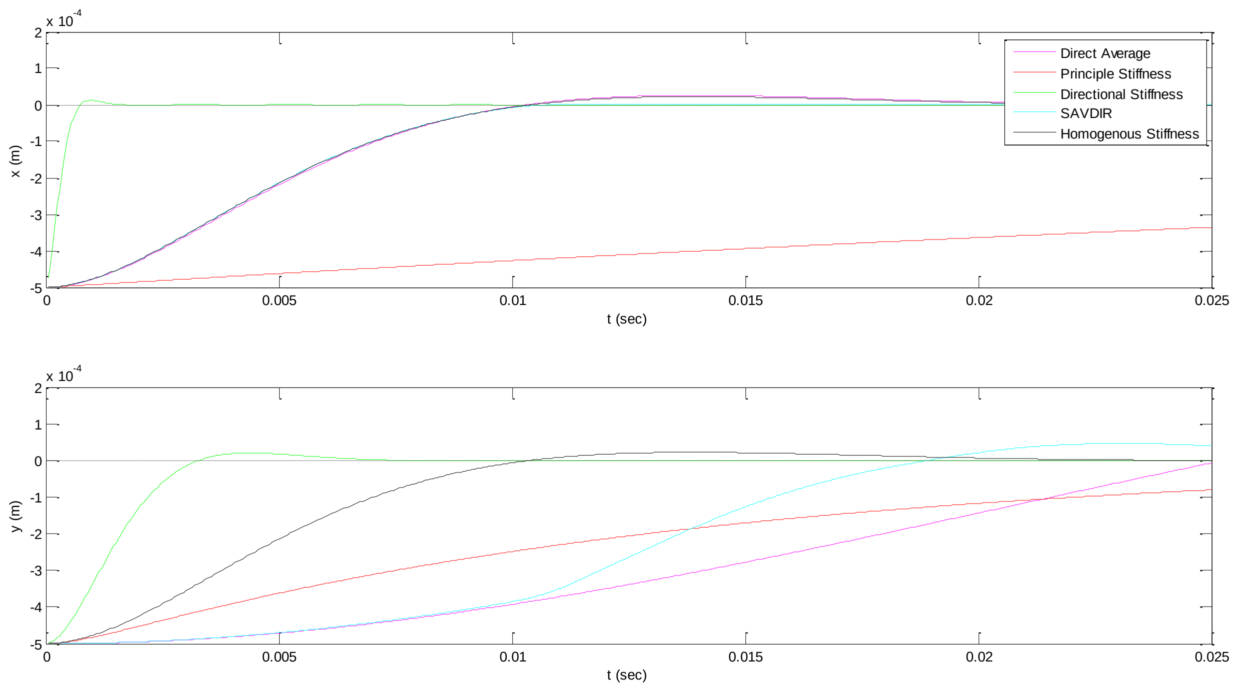


Figure A.4. The trajectory of the end-effector when the damping coefficient and model reaction force are not limited. Despite the difference in trajectory, all algorithms had successfully moved the end-effector to the balancing point (i.e. the origin) stably. The proposed algorithm is shown in black.

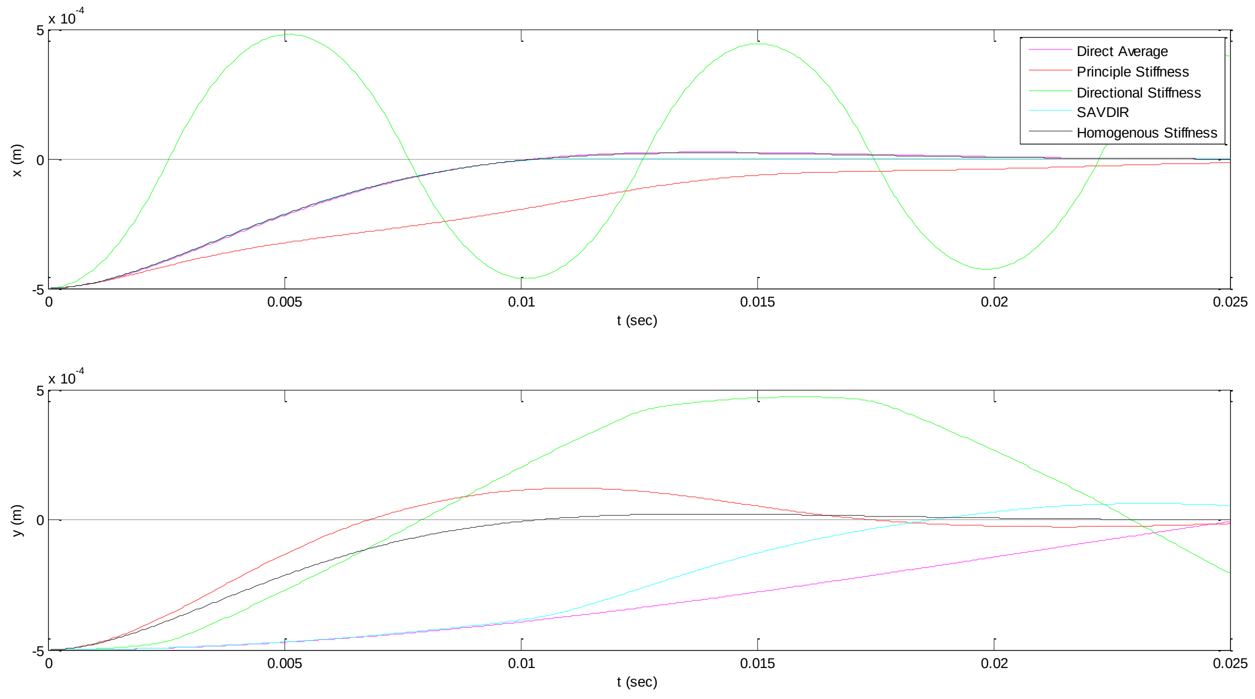


Figure A.5. The trajectory of the end-effector when the damping coefficient and model reaction force are limited. The proposed method (black) maintains an ideal performance, while other tested algorithms become under-damped due to their improperly handled combined stiffness and damping.

A.3 Evaluation of Maximal Renderable Virtual Stiffness

The maximal renderable virtual stiffness is a complicated issues since it involves physical properties and limitations of the hardware, such as the sampling rate, time delay, data precision, physical damping and friction of the haptic interface, etc, [2]. Since the analysis of the control loop is out of the scope of this work, here we provide an evaluation of renderable stiffness inside the virtual environment by only considering the simulation time step, virtual mass of the end-effector, virtual relative damping, and virtual stiffness in the numerical integration (Table. A.4).

Table A.4. Parameters used in the evaluation of renderable stiffness

Parameter and setting name	Symbol	Default Value	Unit
Virtual stiffness	K	N/A	[N/m]
Virtual relative damping	B	0.7	
Virtual mass of the end-effector	M	0.1	[Kg]
Simulation time step	Δt	0.0005	[s]

Based on [3], with the assumption of zero external force, the leapfrog integrator described in Eq. (A.1) can be expanded and rewritten as:

$$a[\mathrm{k}] = \frac{-K \cdot x[\mathrm{k}] - B \cdot v[\mathrm{k}]}{M}$$

$$\begin{aligned} v[\mathrm{k+1}] &= v[\mathrm{k}] + a[\mathrm{k}] \cdot \Delta t \\ &= \frac{-K \cdot \Delta t}{M} \cdot x[\mathrm{k}] + \left(1 - \frac{B \cdot \Delta t}{M}\right) \cdot v[\mathrm{k}] \end{aligned}$$

$$\begin{aligned} x[\mathrm{k+1}] &= x[\mathrm{k}] + v[\mathrm{k+1}] \cdot \Delta t \\ &= \left(1 - \frac{K \cdot \Delta t^2}{M}\right) \cdot x[\mathrm{k}] + \left(\Delta t - \frac{B \cdot \Delta t^2}{M}\right) \cdot v[\mathrm{k}] \end{aligned}$$

The relationship between the end-effector's position, ***x***, and velocity, ***v***, in two consecutive time steps, k and k+1 can thus be expressed in a matrix form, as:

$$\begin{bmatrix} x[k+1] \\ v[k+1] \end{bmatrix} = A_{2\times 2} \cdot \begin{bmatrix} x[k] \\ v[k] \end{bmatrix}, \qquad A = \begin{bmatrix} 1 - \frac{K \cdot \Delta t^2}{M} & \Delta t - \frac{B \cdot \Delta t^2}{M} \\ \frac{-K \cdot \Delta t}{M} & 1 - \frac{B \cdot \Delta t}{M} \end{bmatrix} \tag{A.2}$$

Since the damping coefficient is a function of stiffness, mass and relative damping, as:

$$B = 2 \cdot \zeta \cdot \sqrt{K \cdot M} \tag{A.3}$$

With Eq. (A.2) and Eq. (A.3), the matrix **A** can be rewritten as:

$$A = \begin{bmatrix} 1 - \frac{K \cdot \Delta t^2}{M} & \Delta t - \sqrt{\frac{K}{M}} \cdot 2 \cdot \zeta \cdot \Delta t^2 \\ \frac{-K \cdot \Delta t}{M} & 1 - \sqrt{\frac{K}{M}} \cdot 2 \cdot \zeta \cdot \Delta t \end{bmatrix} \tag{A.4}$$

Given the pre-determined relative damping, virtual mass and constant time step (Table A.4), we then analyze the root locus of the eigenvalues of matrix **A** in the following three cases:

- Case 1: Zero damping (B = 0), and increase the damping coefficient K incrementally until at least one of the eigenvalues falls outside of the unit circle.
- Case 2: Zero stiffness (K = 0), and increase the damping coefficient B incrementally until at least one of the eigenvalues falls outside of the unit circle.
- Case 3: Calculate the damping coefficient B with Eq. (A.3), and increase the stiffness K incrementally until at least one of the eigenvalues of matrix described by Eq. (A.4) falls outside of the unit circle.

The result shows that, when there is no damping (i.e. B=0), the numerical integration only goes unstable with an extremely high stiffness, around 1,600,000 [N/m] (Fig. A.4a). With Eq. (A.3), a stiffness of 1,600,000 would correspond to a damping coefficient of 560 [Ns/m.] In contrast, when the stiffness was set to be zero, the numerical integration goes unstable when the damping coefficient reached 400 [Ns/m] (Fig. A.4b). In other words, the damping coefficient would be the major factor that limits the stability range of the whole numerical simulation. Finally, the result of case 3 shows that, to make the numerical simulation stable, the virtual stiffness must be lower than around 433,000 [N/m], which correspond to a damping coefficient of 291 [Ns/m].

In conclusion, in this section we show that the performance of leapfrog integrator used in current algorithm is limited by the maximal renderable damping. However, both the maximal renderable value of the virtual stiffness and damping coefficient are way higher the actual rendered ones. On the current platform, the default virtual stiffness is 20,000 [N/m]. With the stiffness shaping method introduced in Appendix C, we can guarantee that the combined stiffness would not excess 1.5 times of the default value, i.e. the stiffness would always be less than 30,000 [N/m], indicating a stable result of the numerical integration.

In the future works, the limitation imposed by the damping coefficient should be further studied to increase the stable range of the simulation. Indeed, the stability issue discussed in this section is only for numerical simulation instead of the whole system. A detail inspection of the servo loop on the interface end would also be an interesting future work to increase the renderable stiffness.

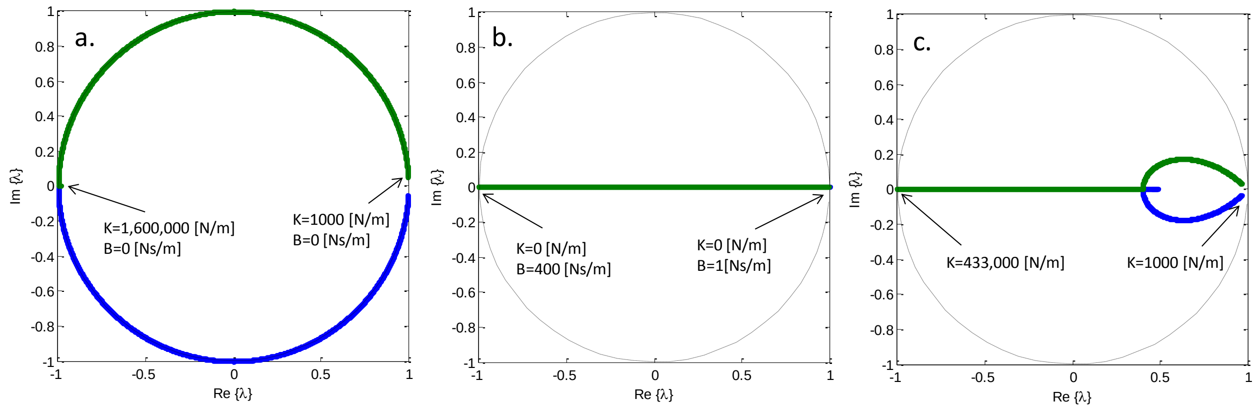


Figure A.4. The root locus of the eigenvalue of the transit matrix described in Eq. (A.2) (a-b) and in Eq. (A.3) (c). By setting either the stiffness or the damping to zero, we can intuitively see the impact of each factor to the stability of the numerical integration. Apparently, the damping coefficient dominates the stability of the simulation since its eigenvalue falls to the outside of the unit circle faster than when pure stiffness was used. By using the relative damping (ζ=0.7), we could get a theoretical maximal renderable stiffness of 433,000 (N/m) and corresponding damping coefficient of 291 (Ns/m).

Reference

[1] P. Lammertse, "MEMO: PL-00-026, Numerieke integratoren en differentiatoren," 2002.

[2] J. E. Colgate and J. M. Brown, "Factors affecting the Z-Width of a haptic display," *Proc. 1994 IEEE Int. Conf. Robot. Autom.*, pp. 3205–3210, 1994.

[3] P. Lammertse, "MEMO: PL-05-091, De leapfrog integrator," 2005.

Appendix B – Original Rendering Algorithm

In the virtual environment of the Simodon® dental trainer, tools are simply represented using polynomial equations due to the high symmetry in their shapes (e.g. a cylinder drill), while the environment (e.g. a tooth) is represented using point cloud due to the high complexity of its contour. When a collision happens, some points would fall into the tool's boundary, and the corresponding reaction force is then calculated based on these points' position and velocity with respect to the tool.

In the original penalty-based algorithm, the shortest distance between a colliding point and the tool's boundary is calculated and treated as the penetration depth (PD), and the corresponding direction is treated as the "gradient" (i.e. the direction of the penetration depth vector) (Fig. B1). Then, all the calculated PDs and gradients are summed and averaged to get the final rendering penetration depth vector, $\vec{d}_{render}$. Taking average is necessary since the density of the point shell is not homogeneous. Without taking the average, areas with higher point density will give higher rendering force even with small PDs, which could possibly distort the perception during the task.

This appendix is organized as follow: Section B.1 provides the detail mathematical procedures in determining the individual penetration depths. Section B.2 shows how to determine the rendered force based on the individual penetration depths. In all sections, we used cylinder-shape tool to demonstrate the calculation in order to keep the calculation neat. The procedure will be similar for calculating the penetration depth when using tools that has more complex geometries.

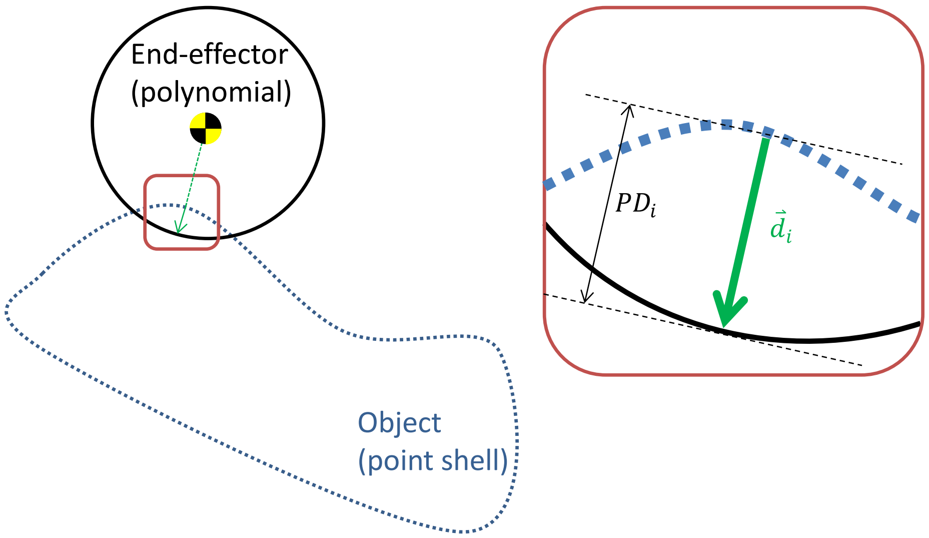


Figure B.1. When collision is detected, the shortest distance between the colliding point (blue) and the end-effector's surface (black) is treated as the penetration depth (PD), and the corresponding direction (pointing from the point to the surface) is treated as the gradient of the PD.

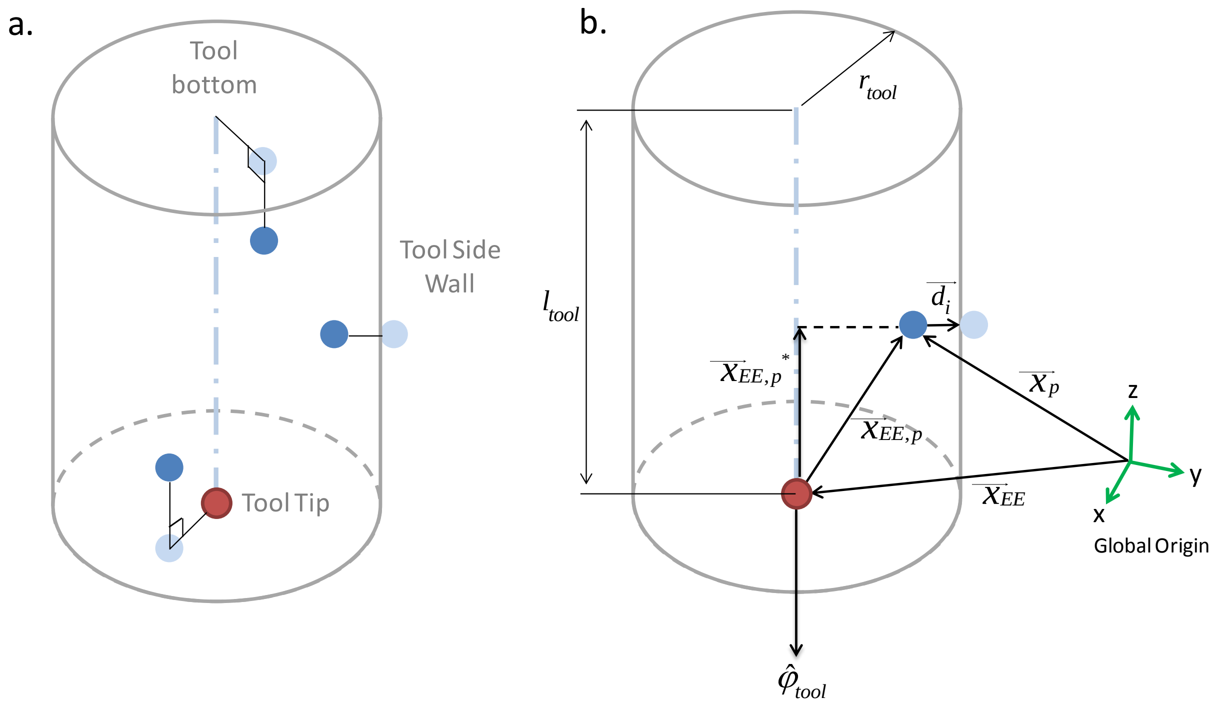


Figure B.2. (a) When a cylinder tool is used, there will be three possible constraint surfaces on which the nearest point (light blue) to the colliding point (dark blue) could be found, including the tool tip surface, the side wall, and the bottom surface. (b) The distance from a colliding point to the three constraint surfaces can be calculated using the relationship between the point and the tool tip.

The rendered force contains three elements: spring force, normal damping force and tangential damping force. The spring force is calculated based on the assigned stiffness, K, and $\vec{d}_{render}$, and the normal damping force is calculated with the assigned damping coefficient and the projection of the model velocity of the end-effector, $\vec{v}_{EE}$, on the direction of $\vec{d}_{render}$. The tangential damping force served as the Coulomb friction force, which will be a function of the tangential velocity and normal spring force.

B.1 Individual Penetration Depth Calculation

In this section, we show that how to determine the penetration depth using the shortest distance method. The idea is straightforward: if there is a point, denoted by $\boldsymbol{p}$, that is detected to be inside the tool's boundary, we will have to find out a point on the tool's surface that is nearest to this point. When a cylinder tool is used, there will be three possible constraint surfaces on which the nearest point could be found, including the tool tip surface, the side wall, and the bottom surface (Fig. B2a). We will calculate the distance from the colliding point to these three constraint surfaces, and choose the shortest distance among the three to be the individual penetration depth.

It is straightforward to calculate the distance between the colliding point and the two flat surfaces of the cylinder. According to Fig. B2b, the distance to the tool tip surface and the bottom surface can be calculated as:

$$\begin{aligned} d_{\mathrm{tip}} &= \left\langle \vec{x}_{\mathrm{EE,p}}, \hat{\varphi}_{\mathrm{tool}} \right\rangle \\ d_{\mathrm{bottom}} &= -l_{\mathrm{tool}} - d_{\mathrm{tip}} \end{aligned}$$

where $\hat{\varphi}_{\mathrm{tool}}$ is a unit vector that represents the tool's orientation.

The corresponding gradient, i.e. the unit vector pointing from the point the surface, are:

$$\begin{aligned} \hat{u}_{\mathrm{tip}} &= \hat{\varphi}_{\mathrm{tool}} \\ \hat{u}_{\mathrm{bottom}} &= -\hat{\varphi}_{\mathrm{tool}} \end{aligned}$$

To calculate the distance from the colliding point to the side wall of the cylinder, we will have to calculate the projection of the colliding point on the median line of the cylinder, as follow:

$$\vec{x}_{\mathrm{EE,p}}^{\;*} = \left\langle \vec{x}_{\mathrm{EE,p}}, \hat{\varphi}_{\mathrm{tool}} \right\rangle \cdot \hat{\varphi}_{\mathrm{tool}}$$

By calculating the distance and relative direction between the colliding point and its projection point on the median line, we can get the distance and gradient from the colliding point to the side wall:

$$\begin{aligned} d_{\mathrm{side}} &= \left\| \vec{x}_{\mathrm{EE,p}} - \vec{x}_{\mathrm{EE,p}}^{\;*} \right\| - r_{\mathrm{tool}} \\ \hat{u}_{\mathrm{side}} &= \frac{\vec{x}_{\mathrm{EE,p}} - \vec{x}_{\mathrm{EE,p}}^{\;*}}{\left\| \vec{x}_{\mathrm{EE,p}} - \vec{x}_{\mathrm{EE,p}}^{\;*} \right\|} \end{aligned}$$

The individual penetration depth is always smaller than zero since the point is inside the tool. Finally, we use the least negative value among d_{tip}, $d_{\mathrm{bottom}}$and d_{side} as the individual penetration depth, and the corresponding gradient as the gradient of the individual penetration depth. For example, if d_{side} has the least negative value, then the gradient will be $\hat{u}_{\mathrm{side}}$, and the individual penetration depth vector will be $d_{\mathrm{side}} \cdot \hat{u}_{\mathrm{side}}$.

B.2 Rendered Force Calculation

Suppose there are **n** colliding points, the individual penetration depth and gradient of each point will be denoted as d_i and $\hat{u}_i$, $i = 1,2,3 \dots n$, respectively. To calculate the magnitude and direction of the rendered force, we must merge all the individual penetration depths and gradients first to form a rendered penetration depth, d_{render}, and a render gradient, $\hat{u}_{\text{side}}$. In the original rendering algorithm, it is done as follow:

$$d_{\text{render}} = \frac{\sum_{i=1}^{n} d_i}{\left\| \sum_{i=1}^{n} \hat{u}_i \right\|} \qquad \hat{u}_{\text{render}} = \frac{\sum_{i=1}^{n} \hat{u}_i}{\left\| \sum_{i=1}^{n} \hat{u}_i \right\|} \tag{B.1}$$

The rendered force consists of three major elements: spring force, normal damping force, and friction force. The spring force is determined based on the penetration depth while the damping force based on the projection of end-effector's velocity, $\vec{v}_{EE}$, on the direction of rendered force. The friction force is comprised of the static friction force and the kinetic friction force. To calculate these forces, several virtual properties are introduced (Table B.1).

Table B.1. Virtual Properties

Property name	Symbol	Default Value	Unit
Stiffness	K	20000	[N/m]
Relative damping	ζ	0.7	
Mass of end-effector	M	0.1	[Kg]
Normal Damping Coefficient	B	$2 \cdot \zeta \cdot \sqrt{K \cdot M}$	
Constant Tangential Damping Coefficient	μ_s	5	
Force-Coupled Tangential Damping Coefficient	μ_f	10	

With the parameters defined in Table B1, the reaction forces were calculated as follow:

- Spring Force

$$\vec{f}_{\text{spring}} = -K \cdot d_{\text{render}} \cdot \hat{u}_{\text{render}}$$

- Normal Damping Force

$$\vec{v}_{\text{normal}} = \left\langle \vec{v}_{\text{EE}}, \hat{u}_{\text{render}} \right\rangle \cdot \hat{u}_{\text{render}}$$
$$\vec{f}_{\text{damping,n}} = -B \cdot \vec{v}_{\text{normal}}$$

- Tangential Damping Force

$$\vec{v}_{\text{tangential}} = \vec{v}_{\text{EE}} - \vec{v}_{\text{normal}}$$
$$\vec{f}_{\text{damping, t}} = -\left(\mu_{\text{s}} + \mu_{\text{f}} \cdot \left\| \vec{f}_{\text{spring}} \right\|\right) \cdot \vec{v}_{\text{tangential}}$$

- Rendered Force

$$\vec{f}_{\text{render}} = \vec{f}_{\text{spring}} + \vec{f}_{\text{damping,n}} + \vec{f}_{\text{damping,t}}$$

B.3 Problems in the Original Rendering Algorithm

In penalty-based haptic rendering, how to merge all the individual penetration depths to form a rendered penetration depth is still a challenge. Using the shortest distance method introduced in section B.2, the gradient of the colliding points, $\hat{u}$, are more or less in the same direction during a 1-DOF collision, e.g. poke a flat surface with the virtual tool. But when it comes to multi-lateral tasks, such as the peg-in-hole task, reaction forces of the colliding points are no longer parallel and start to cancel out each other. It appears that the original algorithm failed to render a stable reaction force for such multi-lateral collisions, and would result in vibrations and deviated force direction that seriously deteriorated the authenticity of the haptic feedback. In this section, issues in the current rendering algorithm are explained in detail and discussed.

B.3.1 Gradient Flipping

In conventional 3-DOF penalty-based haptic rendering in which the end-effector was modeled as a single point, a “pop-through” effect or discontinuity in force direction would occur when the end-effector came across the median line of the penetrated object [1]. In the current rendering algorithm, we also observed similar problems.

Thanks to the admittance control system structure, the haptic interface used in this project can render very high stiffness (default value is 20000 N/m). The high stiffness prevented high penetration depth, and thus the “pop-through” was seldom observed during 1-DOF collision.. However, in peg-in-hole task, reaction forces appear to come from opposite directions and can cancel out each other (Fig. B.3 black line segments). Without the torque feedback, the high virtual stiffness will try to maintain the balance between the forces from the two sides of the wall. This means that the magnitude of the penetration depth on the two sides of the wall should always be identical, and would depend on on the orientation of the tool. As the orientation of the tool deviate from that of the hole (Fig. B.3b), penetration depth would increase rapidly, while the force feedback would still have a net magnitude close to zero (due to the cancellation of opposite gradients). In other words, the tool could easily penetrate the environmental object with high penetration depth. Once the colliding point reaches the median line of the tool, the gradient of the penetration depth will “flip” by 180°, since its nearest point on the tool’s surface had switch to the other side with respect to the median line.

This artifact is similar to what was observed in conventional 3-DOF rendering when using the point end-effector to interact with thin objects. As a result, the rapidly changing gradients could cause a fast changing direction of the rendered force, and thus result in a perceivable vibration on the haptic interface. In the worst case, when there are too many flipped gradients, the rendered force would “pull” the tool into the object, causing a total failure of the haptic rendering.

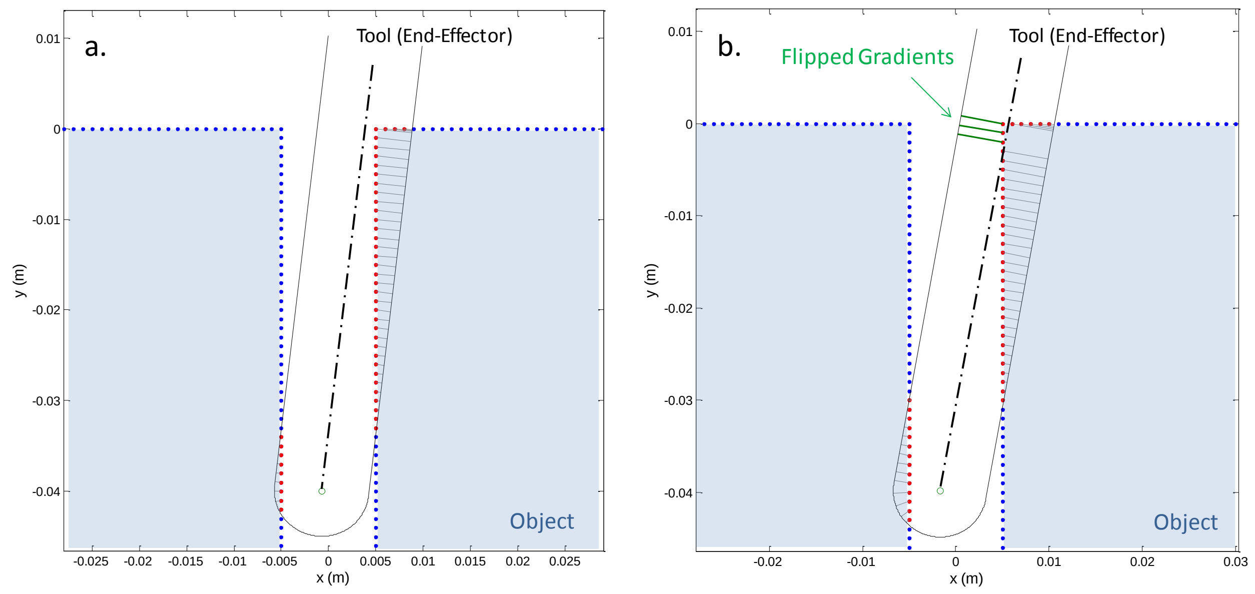


Figure B.3. (A) Ideally, the penetration depth calculated with the shortest-distance-method will always result in a force that pushes the tool out of the object's surface. (B) However, once the colliding points came across the median line (black dash line), the gradient will flip suddenly (green lines), causing a force that pulls the tool into the object's surface.

B.3.2 Improper Damping Force

In order to properly damp the interaction between virtual objects, one must calculate the damping coefficient, **B**, based on the given relative damping ratio, mass, and stiffness. When there are multiple colliding points, the combined stiffness formed by summing the individual penetration depths could rapidly build up. Excessive combined stiffness can cause system instability, and even if the combined stiffness is renderable, the corresponding damping must be carefully calculated to realize stable system behavior.

In the original algorithm, the damping force is comprised of two components: 1) a normal damping force that acts along the opposite direction of the rendered penetration depth, $-\hat{u}_{\text{render}}$, and 2) a tangential damping force that acts along the direction perpendicular to $-\hat{u}_{\text{render}}$. This configuration can be expressed in an equivalent system, in which the center of mass (COM) of the end-effector has penetrated an implicit plane (Fig. B.4, red line) whose surface normal vector is parallel to the direction of the rendering force. And the shortest distance between this implicit plane and the COM of the end-effector is the magnitude of the rendering penetration depth $\vec{d}_{\text{render}}$ (Fig. B.4 right frame).

According to Section B.2, the magnitude of the normal damping force was defined to be proportional to the projected velocity of the end-effector, $\vec{v}_{\text{normal}}$. This method works fine when the gradients of the colliding points are more or less in the same direction. In such condition, Eq. (B.1) would result in a combined stiffness around the default value, K , and thus the system can be properly damped using the damping coefficient, B .

However, in a distributed-contact configuration such as the peg-in-hole task (Fig. B.5), the implicit plane is no longer parallel to the colliding surface. In other words, the damping force must be calculated separately for each colliding surface rather than solely based on the implicit surface. This would require the evaluation of combined stiffness, which was not taken into account in the original algorithm. More detail about the calculation of combined stiffness and the corresponding damping is introduced in Appendix C.

Consequently, when the interaction involves distributed contact, the improperly evaluated damping forces (Fig. B.5 black arrows) could result in an under-damped system.

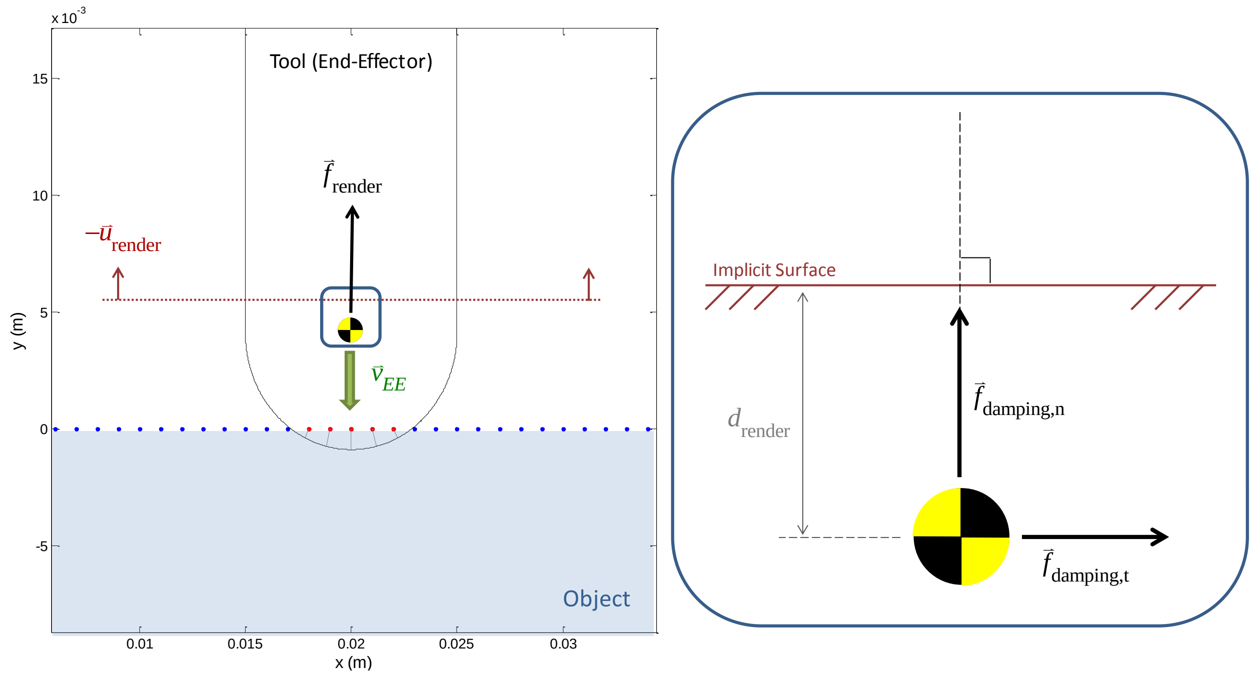
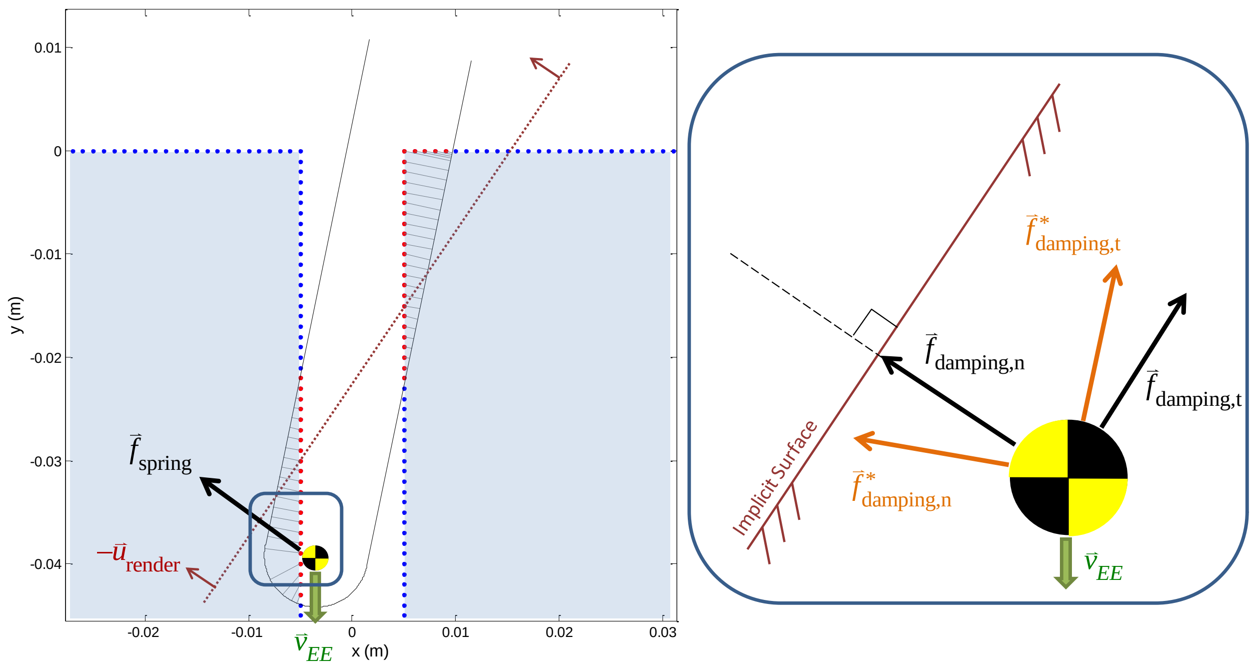


Figure B.4. The normal and tangential forces in unilateral collision condition. The implicit plane (dark red) formed by the rendering spring force vector is parallel to the surface of the object (light blue) so that the original rendering algorithm can properly reflect the surface property.



y (m)
x (m)
$\vec{f}_{\text{spring}}$
$-\vec{u}_{\text{render}}$
$\vec{v}_{EE}$
Implicit Surface
$\vec{f}^{*}_{\text{damping,t}}$
$\vec{f}_{\text{damping,n}}$
$\vec{f}_{\text{damping,t}}$
$\vec{f}^{*}_{\text{damping,n}}$

Figure B.5. The normal and tangential forces in multi-lateral collision condition, in this case the peg-in-hole task. The implicit plane (dark red) is no longer parallel to the in-contact surface of the object. The actual normal damping force is not perpendicular to the implicit plane anymore, and the tangential damping force is not perpendicular to the normal damping force anymore.

B.3.3 Improper Average Method

In the original algorithm, the rendered penetration depth, d_{render}, is calculated by dividing the summation of individual penetration depths by the length of the summation of individual gradients, as follow:

$$d_{\text{render}} = \frac{\sum_{i=1}^{n} d_i}{\left\| \sum_{i=1}^{n} \hat{u}_i \right\|_2}$$
$$\vec{u}_{\text{render}} = \frac{\sum_{i=1}^{n} \hat{u}_i}{\left\| \sum_{i=1}^{n} \hat{u}_i \right\|_2} \qquad \text{(B.1)}$$

In peg-in-hole task, a huge amount of gradients are opposite to each other, making the denominators in the above equations very small or even zero. In contrast, the summation of individual penetration depth is relatively large in magnitude. As a result, the rendered penetration depth would have an extremely large magnitude. Moreover, the rendered gradient could rapidly change in its direction (in the worst case scenario, a 180° change in direction in single time step).

The aforementioned problems can easily be demonstrated with a simple example: Suppose that in time **k**, there are **n** colliding points (**n** >> 1) with their penetration depth vectors pointing rightward, and another **n+1** points with their penetration depth vectors pointing leftward. Assume that the magnitudes of all individual penetrations are 1, and the rightward gradient and leftward gradient are denoted as $\hat{u}$ and $-\hat{u}$, respectively, the resulting d_{render} and $\hat{u}_{\text{render}}$ would be:

$$\vec{u}_{\text{render}}[k] = \frac{n \cdot \hat{u} + (n+1)\cdot(-\hat{u})}{\left\| n \cdot \hat{u} + (n+1)\cdot(-\hat{u}) \right\|_2} = \frac{-\hat{u}}{\left\| -\hat{u} \right\|_2} - \hat{u}$$

$$d_{\text{render}}[k] = \frac{n + (n+1)}{\left\| -\hat{u} \right\|_2} = 2n+1$$

If at the next time step, due to a tiny change in the configuration, there are 2 more points fall inside the tool, both of which have penetration depth vector pointing rightward, the resulting d_{render} and $\hat{u}_{\text{render}}$ would be:

$$\hat{u}_{\text{render}}[k+1] = \frac{(n+2)\cdot \hat{u} + (n+1)\cdot(-\hat{u})}{\left\| (n+2)\cdot \hat{u} + (n+1)\cdot(-\hat{u}) \right\|_2} = \frac{\hat{u}}{\left\| \hat{u} \right\|_2} = \hat{u}$$

$$d_{\text{render}}[k+1] = \frac{(n+2) + (n+1)}{\left\| \hat{u} \right\|_2} = 2n+3$$

By comparing the rendered penetration depth and gradient of the two cases, it is clear that when **n** is way higher than 1 (i.e. n>>1), a few extra colliding points will only cause a slight change in the magnitude of the rendered penetration depth, d_{render} (and so the rendering force), but could make a sudden change in the rendered gradient, $\hat{u}_{\text{render}}$ (and so the direction of the rendering force). Consequently, a high force that has rapidly changing direction could cause serious stability problem, as the serious vibrations observed on the current haptic interface.

Reference

[1] C. B. Zilles and J. K. Salisbury, “A constraint-based god-object method for haptic display,” *Proc. 1995 IEEE/RSJ Int. Conf. Intell. Robot. Syst. Hum. Robot Interact. Coop. Robot.*, vol. 3, pp. 146–151, 1995.

Appendix C – Proposed Rendering Algorithms

C.1 Individual Penetration Depth Calculation

Despite the drawbacks described in Appendix B, the shortest distance method has an important advantage in terms of computational efficiency, especially when the tool has a symmetrical geometry. As such, we implemented a simple fix to the original version of shortest distance method and use it to calculate the individual penetration depth in this work.

In order to fix the gradient flipping and pop-through problem, the gradient of the penetration depth vector was replaced by the inverse of the surface normal vector of the environmental object. Each colliding point contains a local surface normal vector, determined by taking average of its neighboring surfaces that were formed by marching cube method [1]. During the computation of the ith individual penetration depth, we simply assigned the inverse of its surface normal to the gradient of its penetration depth vector, as:

$$\hat{u}_i = -\hat{u}_{\mathrm{surface,i}}$$

Since the surface normal vector of the environmental object is invariant during the interaction, the gradient flipping would no longer occur. Similar approach was also adopted in the previous studies, e.g. [2][3][4]. Indeed, this method gave a steady rendered gradient, but it could suffer from the underestimation of the actual penetration depth (Fig. C.1). Fortunately, the error would be negligible as long as the interpenetration is kept small enough. An alternative method to calculate the precise distance from a point to the surface of a cylindrical tool is provided in Appendix F.1.

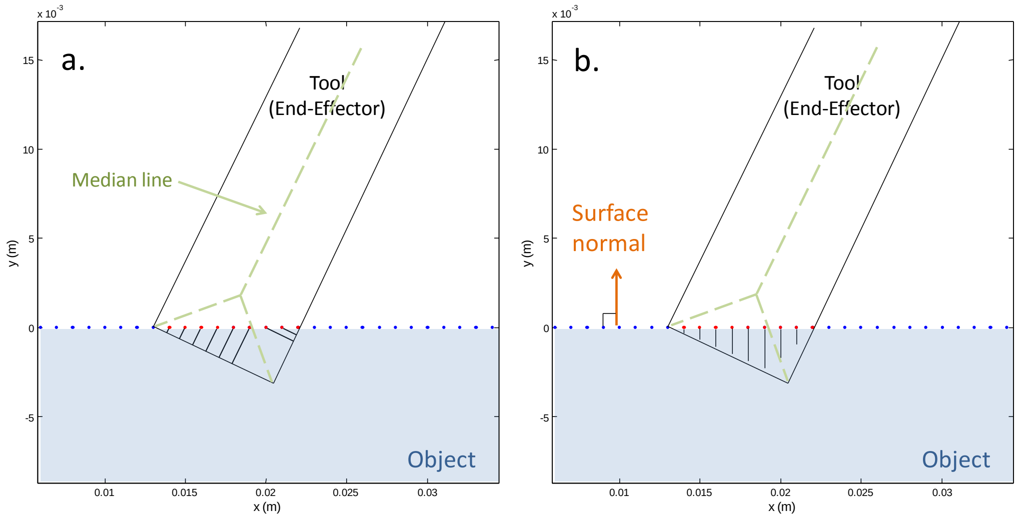


Figure C.1. (a) In the original shortest distance method, the gradient (black) would flip when the colliding point came across the median line (light green). (b) By assigning the inverse of surface normal (orange) to the gradient, the direction of the individual penetration would only depend on the geometry of the penetrated object regardless the median line of the tool. But this method would suffer from the underestimation of the penetration depth as some of the black segments in (b) do not reach the boundary of the tool.

C.2 Local Average (optional step)

In the virtual world of the current system, the environmental object, e.g. a tooth, was modeled as point-shell for haptic rendering. In order to visualize the virtual object, the raw density file was first transformed to triangle mesh using the marching cube algorithm [1]. The vertexes of the triangle mesh were then used to form the point shell.

In the marching cube algorithm, there are 14 basic patterns to describe the intersection between surfaces and a cube. If a cube is intersected by a surface, there could be a least one intersecting point and at most 5 intersecting points, indicating that the point density on the point-shell can vary across the surface of the modeled object. This property is not desired in penalty-based haptic rendering since the rendered force can easily be dominated by contacting areas that have higher point densities. As a result, the reaction force in certain direction would be underestimated or even ignored, making a unrealistic haptic feedback.

A simple local average method was used in this work to make the weighting of each colliding area identical in the calculation of rendered force. The idea is straightforward: Despite the inhomogeneous density of the raw intersecting points, the cube contains these points are uniformly distributed in the three-dimensional grid with constant density. By using the cube as the colliding point, each contact area would have same point density. To do so, a "local penetration depth" is first calculated by taking average of the individual penetration depth vectors of all colliding point in a single cube, as follow:

$$d_{\text{cube,i}} = \left\| \frac{1}{n} \cdot \sum_{i=1}^{n} d_i \cdot \hat{u}_i \right\|_2$$

$$\hat{u}_{\text{cube,i}} = \frac{\sum_{i=1}^{n} d_i \cdot \hat{u}_i}{\left\| \sum_{i=1}^{n} d_i \cdot \hat{u}_i \right\|_2}$$

where **n** is the total number of colliding point in a single cube. In the rest of this work, the locally averaged penetration depth, $d_{cube,i}$, and gradient, $\hat{u}_{cube,i}$, will be used to determine the rendered penetration depth vector, . To make the equations compact, $d_{cube,i}$ and $\hat{u}_{cube,i}$ will be denoted as d_i and $\hat{u}_i$, respectively.

This algorithm is not a necessary step when the stiffness shaping algorithm (Section C.4) is already in use. The reason is that, the stiffness shaping algorithm was designed to decouple the rendered force from the local point density so that the problem caused by inhomogeneous point density would no longer exist.

C.3 Gradient-Based Point Reduction

Updating rate is an important factor in haptic rendering that a low updating rate (<1000Hz) could result in perceivable discontinuity in the haptic feedbacks. On the current platform, the default updating rate is 2048Hz, which imposed a challenge to the efficiency of the rendering algorithm.

Since the optimization of the algorithm is out of the scope of this work, the point reduction method proposed here can be regarded as a temporary solution to increase the computational efficiency. Nonetheless, this point reduction method had still provided satisfying result that it attenuate the computational load and maintain the haptic feedback quality.

The idea of this method is to keep the colliding points that have the maximal local penetration depth among colliding points that have similar gradient. There are two major parameters for control the point reduction rate (Table C.1).

Table C.1. Main parameters in the point reduction algorithm

Parameter Description	Symbol	Default Value	Unit
Threshold of inner-product value for combining points	α_{TH}	0.995	
Max number of point after reduction	n_{ext}	100	

Suppose there are **n** colliding points stored in a raw data list, L_p, before the point reduction. An empty list, L_p^{ext}, that contained n^{exp} slots would be first constructed in order to store the information of the colliding points extracted from L_p. To initiate the point reduction process, a random colliding point in L_p was assigned to the first slot of L_p^{ext} as a kernel. A loop was then run through all colliding points in L_p to determine if it should be added to L_p^{ext} or eliminated. The major steps of this algorithm are introduced in Algorithm C.1.

Algorithm C.1. Gradient-Based Point Reduction

```
1   FOR Read the ith colliding point, $p_{\mathrm{i}}$, from the raw data list, $L_{\mathrm{p}}$, i = 1~n
2       FOR Read the jth colliding point, $p_j^{ext}$, from $L_p^{ext}$, j = 1~m
3           IF $\langle \hat{u}_{\mathrm{i}}, \hat{u}_j^{ext} \rangle > \alpha_{\mathrm{TH}}$
4               IF $d_{\mathrm{i}} > d_j^{ext}$
5                   Replace $p_j^{ext}$ with $p_{\mathrm{i}}$
6                   Set “pointReplaced” to “true”
7                   BREAK
8               END IF
9           END IF
10      END FOR
11      IF (pointReplaced==True) Append $p_{\mathrm{i}}$ to the extracted point list $L_{\mathrm{p}}$; END IF
12  END FOR
```

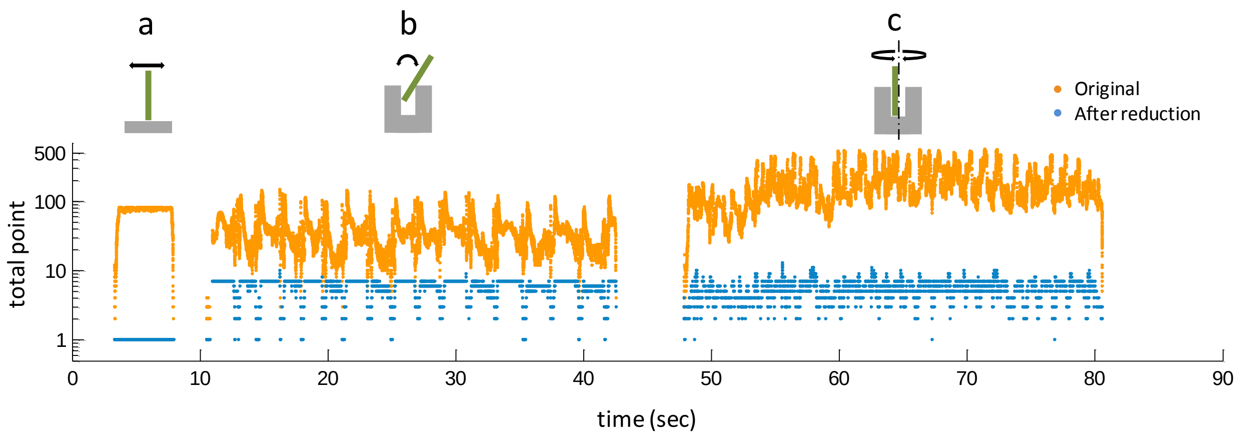


Figure C.2. The number of colliding point before (orange dots) and after (blue dots) the gradient-based point reduction. (a) When the human operator use the virtual tool to interact with a flat surface, the number of point after the reduction process would always equal to 1 since all the colliding points have the same gradient. (b-c) If there are multiple contact sites, the algorithm would extract the locally maximal individual penetration depth on a directional basis.

With this algorithm, there will be only one colliding point left when interacting with a flat surface where all raw colliding points have the same gradient (Fig. C.2a). Moreover, the corresponding penetration depth of this colliding point would be the largest among all raw points. In other words, the rendered penalty spring force will totally depend on the maximal individual penetration depth.

When there are multiple contact sites (Fig. C.2b-c), such as what would happen in the peg-in-hole task, the algorithm will extract the locally maximal individual penetration depth on a directional basis. The data of the validation test shows that when the operator rotated the tool with respect to a vector perpendicular to the orientation of the hole (Fig. C.2b), the original and the reduced number of colliding points were 34.1±19.5 and 5.5±1.9, respectively. When the tool was spun with respect to the orientation of the hole (Fig. C.2c), the original and the reduced number of colliding points were 163.7 ± 103.1 and 5.2 ± 1.7, respectively.

Apparently, this largely reduced the redundant information from the original point list. Throughout the validation test, the haptic feedback was stable and realistic, indicating that this algorithm had preserved most information of the collision with a small amount of points. This concept is similar to, but not as rigorous as, the direction clustering technique commonly used in data analysis. However, our method requires less computational power, which is preferred in our haptic rendering case.

C.4 Rendered Penetration Depth Calculation

In this section, we show how to use the individual penetration depths calculated in the previous sections to determine the rendered penetration depth vector, $\vec{d}_{\text{render}}$. Since our system cannot perform torque feedback, the gradient and magnitude of the rendered penetration depth must be manipulated in a way that it can generate reasonable dynamics to the end-effector during interaction that involves multiple contact sites, such as the peg-in-hole situation (Fig. C.3a).

There existed several way to "merge" the individual penetration depths, but none of them can be applied to our case directly. By summing all the individual penetration depth vector directly [6][7], the combined stiffness could rapidly build up when there are many colliding points (Fig. C.3c), and lead to system instability [8]. Taking average of the individual penetration depth vectors, on the other hand, can solve the problem of excessive stiffness [3]. However, this method would still deteriorate the haptic feedback when distributed contact is involved. If most of the colliding points have the same or parallel gradients in their penetration depth vectors, the rendered force will be dominated by these points while the contribution of other colliding points is averaged out (Fig. C.3c). This is not desired since the feedback would be coupled to the point density again.

Virtual coupling was a common way to limit the excessive stiffness when there are multiple contact sites and multiple colliding points during the interaction [4]. However, virtual coupling only bounds the combined stiffness after the rendered penetration depth was determined, indicating that it does not decouple the rendered force from the density of point. As a result, if there are more colliding points that have similar or parallel gradients in their penetration depth vectors, they will dominate the render force (Fig. C.3d).

The domination of the rendered force's direction by points that has similar gradients could lead to serious problem in interactions that involves multiple contact sites. Take the peg-in-hole task for an example, when the peg is rotated, since most colliding points would be on the side wall of the hole, the reaction force will mainly pointing horizontally, and will thus let the peg easily penetrate the upper corner of the hole (Fig. C.3a).

Here, we proposed a method that could regulate the combined stiffness along the gradient of all penetration depth vectors to a desired value **K**, while decoupling the force's direction from point density (Fig. C.3a). This method can also be thought as taking average of the individual penetration depths based on their gradients rather than the total number of colliding points.

The following sections are organized as follow: In section C.4.1, we will show how to determine the directional combined stiffness. In section C.4.2, we will introduce a method to regulate all the directional combined stiffness to the desired value, in our case, **K**. In section C.4.3, we will demonstrate how to calculate the damping force.

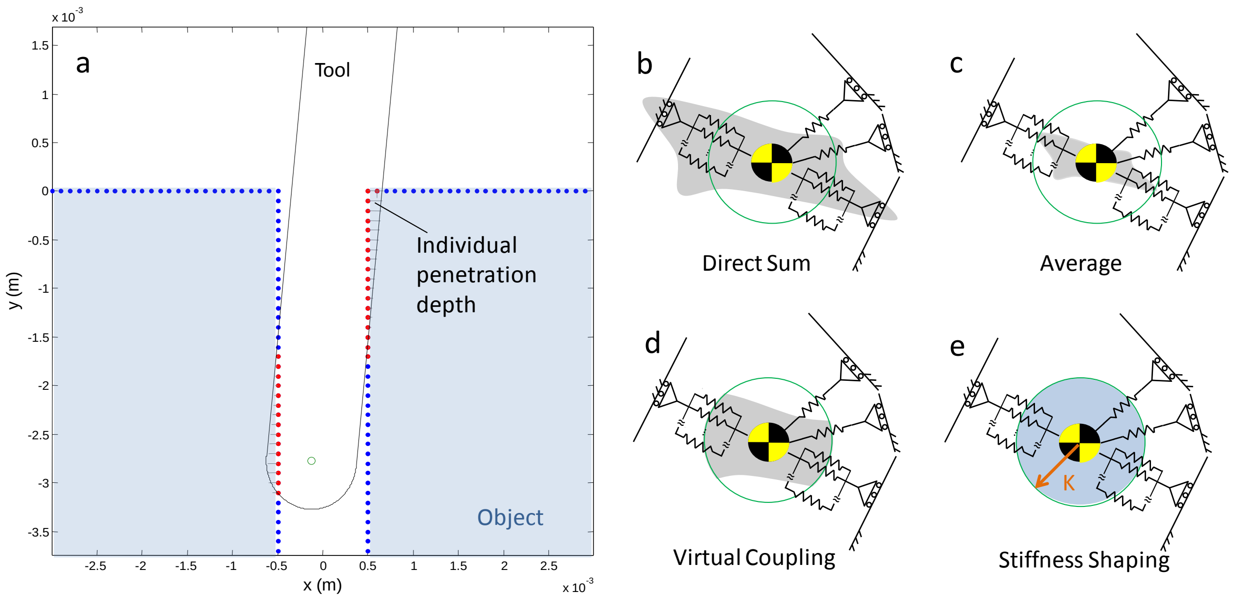


Figure C.3. (a) In peg-in-hole task, most colliding points (red dot) have similar or parallel gradients of their penetration depth (horizontal in this case). When the tool is rotated clockwise, it could easily penetrate the upper corner of the hole since the direction of the rendered force is almost horizontal. (b-e) The underlying stiffness field (shaded area) can vary depending on how the individual penetration depths are merged. The radius of the green circle represents the maximal renderable stiffness. (b) Direct summation method would result excessive stiffness and thus instability. (c) Average method could suffer from the coupling between the direction of the rendered force and the point density, which would result in underestimation of reaction force in particular directions. (d) Virtual coupling method also has the same problem as the average method. The direction of the rendered force would depend on points that have similar or parallel gradients. (e) We proposed a method to regulate the stiffness field along the gradient of all colliding points. This will decouple the point density from the rendered force in all directions.

C.4.1 Directional Combined Stiffness

Suppose there are **n** colliding points at certain time step (Fig. C.3a), the system can be thought as a mass attached with **n** springs in various directions. Each spring has a stiffness of **K**. Here we have assumed that angular velocity of the tool is constant throughout the whole time step based on the fact that the rotational movement in dental operation is always slow and gentle, and the interval of a time step is only 0.5 ms. To evaluate the combined stiffness along $\hat{u}_i$, i.e. the gradient of the ith point, we have to calculate the reaction force along $\hat{u}_i$ by applying a infinitesimal displacement along $\hat{u}_i$. The total reaction force can be calculated as follow:

$$\vec{f}_{c,i} = \sum_{j=1}^{n} \vec{f}_j = -K \cdot \Delta x \cdot \sum_{j=1}^{n} \left\langle \hat{u}_i, \hat{u}_j \right\rangle \cdot \hat{u}_j, \qquad \Delta x \to 0 \tag{C.1}$$

Since we are interested in the reaction force along $\hat{u}_i$, the reaction force in Eq. (C.1) must be projected to $\hat{u}_i$ so that:

$$\vec{f}^{*}_{c,i} = \left\langle \vec{f}_{c,i}, \hat{u}_i \right\rangle \cdot \hat{u}_i = -K \cdot \Delta x \cdot \sum_{j=1}^{n} \left\langle \hat{u}_i, \hat{u}_j \right\rangle^2 \tag{C.2}$$

By rearranging Eq. (C.2), we can obtain the combined stiffness along $\hat{u}_i$:

$$K_{c,\mathrm{i}} = \frac{\left\| \vec{f}^{*}_{c,i} \right\|}{\Delta x} = K \cdot \sum_{j=1}^{n} \left\langle \hat{u}_i, \hat{u}_j \right\rangle^2 = \rho_i \cdot K \tag{C.3}$$

The coefficient ρ_i is used to represent the ratio of the combined stiffness to the default stiffness. Apparently, ρ_i will be no less than 1, and can have a maximal value of **n** when all the **n** springs have same gradients. Using the same approach, we can calculate the combined stiffness along all **n** directions.

C.4.2 Stiffness Shaping

Based on Eq. (C.3), we can organize all stiffness ratio ρ_i in a matrix form as follow:

$$\boldsymbol{\rho}_{n\times1} = \boldsymbol{A}_{n\times n} \cdot \mathbf{1}_{n\times1}$$

$$\boldsymbol{\rho} = \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_n \end{bmatrix} \qquad \boldsymbol{A} = \begin{bmatrix} \left\langle \hat{u}_1, \hat{u}_1 \right\rangle^2 & \cdots & \left\langle \hat{u}_1, \hat{u}_n \right\rangle^2 \\ \vdots & \ddots & \vdots \\ \left\langle \hat{u}_n, \hat{u}_1 \right\rangle^2 & \cdots & \left\langle \hat{u}_n, \hat{u}_n \right\rangle^2 \end{bmatrix} \qquad \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

As mentioned before, the stiffness ratio ρ_i would have a value no less than 1. If the stiffness ratio is so high that the combined stiffness exceeds the maximal stiffness that can be rendered by the hardware, the system would become unstable [8]. According to Eq. (C.3), if ρ_i has a value of 1, the combined stiffness will have a desired value **K**. To realize this, we introduce a new parameter $\boldsymbol{c}$, called the stiffness scaling coefficient, to manipulate the stiffness of each spring so that the value of all ρ_i becomes 1.That is:

$$\rho_i^* = 1 = \sum_{j=1}^{n} c_j \cdot \left\langle \hat{u}_i, \hat{u}_j \right\rangle \tag{C.4}$$

Based on Eq. (C.4), the matrix form can be rewritten as:

$$\boldsymbol{\rho}^*_{\mathrm{n}\times 1} = \mathbf{1}_{\mathrm{n}\times 1} = \boldsymbol{A}_{\mathrm{n}\times\mathrm{n}} \cdot \boldsymbol{c}_{\mathrm{n}\times 1}$$

$$\begin{bmatrix} \rho_1^* \\ \vdots \\ \rho_n^* \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \left\langle \hat{u}_1, \hat{u}_1 \right\rangle^2 & \cdots & \left\langle \hat{u}_1, \hat{u}_n \right\rangle^2 \\ \vdots & \ddots & \vdots \\ \left\langle \hat{u}_n, \hat{u}_1 \right\rangle^2 & \cdots & \left\langle \hat{u}_n, \hat{u}_n \right\rangle^2 \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_i \end{bmatrix}$$

If the matrix $\boldsymbol{A}$ is invertible, then the vector $\boldsymbol{c}$ will be equal to $\boldsymbol{A}^{-1}\mathbf{1}$. However, most of the time, $\boldsymbol{A}$ is non-singular and thus non-invertible. This indicates that $\boldsymbol{c}$ should be solved with linear programming methods, which could compromise the high computational efficiency of penalty-based rendering method. As such, rather than solving the vector $\boldsymbol{c}$ analytically, we proposed an iterative method to make all the stiffness ratio ρ sufficiently close to 1 by approximating values in **c**.

At each iteration, the stiffness ratio ρ is first evaluated as Eq. (C.4), and is then used to update the stiffness scaling coefficient **c** as follow:

$$\rho_i^{m+1} = \sum_{j=1}^{n} \mathrm{c}_j^m \cdot \left\langle \hat{u}_i, \hat{u}_j \right\rangle^2$$

$$c_i^{m+1} = \frac{c_i^m}{\rho_i^{m+1}}$$

where m denotes for the iteration number. According to the logged real-time data, after one iteration, the resulting stiffness ratio would be already in the range of 0.8~1.3. This range would further shrink as the number of iteration increased (Fig. C.4), and would converge to 1 theoretically. Since the maximal stiffness that can be rendered by our interface stably is around 60,000 N/m, and the default stiffness is 20,000N/m, one iteration number would be sufficient. However, the more accurate the combined stiffness is, the less the under-damping or over damping would be (Section C.4.3). On our system, we assigned a default number of iteration of 3 to the system.

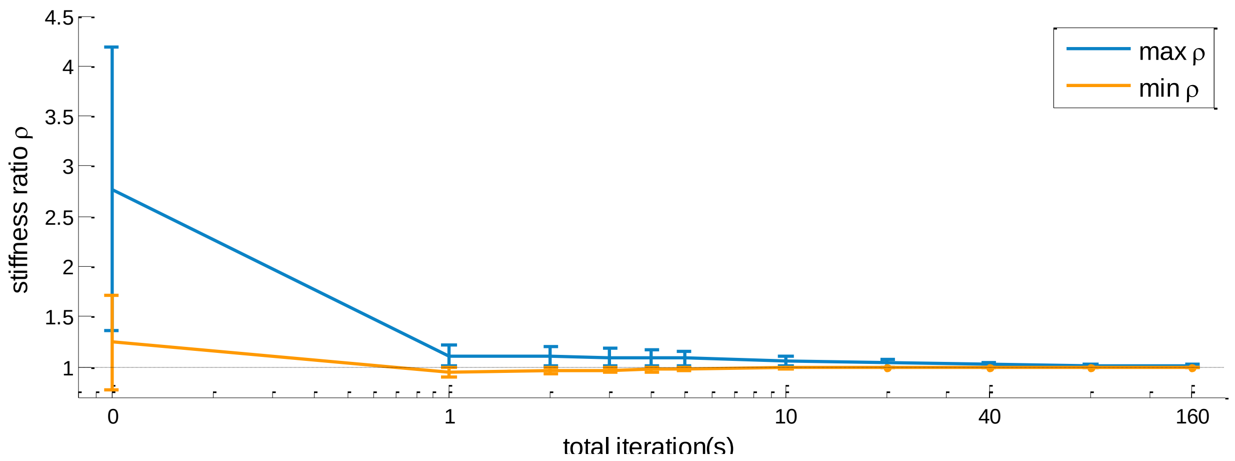


Figure C.4. The logged real-time data shows that the maximal (blue) and minimal (orange) directional stiffness were converging to 1 as the iteration number increased. A dashed black line is used to represent the desired stiffness ratio. A ratio equals to 1 means that the combined stiffness is equal to the desired stiffness **K**. Both x and y axis are expressed in log scale.

In the following calculations, we use the asterisk mark to represent the parameters after the iterative approximation to prevent confusion. With the approximated stiffness coefficient, c^*, the stiffness of the spring correspond to the ith colliding point would become:

$$K_{\mathrm{i}} = c_{\mathrm{i}}^{*} \cdot K$$

And the rendered spring force can be calculated as:

$$\begin{aligned} \vec{f}_{\text{spring}} &= \sum_{i=1}^{n} \vec{f}_{\text{spring,i}} \\ &= -\sum_{i=1}^{n} K_{\mathrm{i}} \cdot d_{\mathrm{i}} \cdot \hat{u}_{\mathrm{i}} \\ &= -\sum_{i=1}^{n} \left(c_{\mathrm{i}}^{*} \cdot K\right) \cdot d_{\mathrm{i}} \cdot \hat{u}_{\mathrm{i}} \\ &= -K \cdot \sum_{i=1}^{n} c_{\mathrm{i}}^{*} \cdot d_{\mathrm{i}} \cdot \hat{u}_{\mathrm{i}} \\ &= -K \cdot \vec{d}_{\text{render}} \end{aligned} \tag{C.5}$$

Eq. (C.5) shows that regulating the stiffness corresponding to each colliding point can also be thought as manipulating the individual penetration depth to realize a desired stiffness **K** along the gradients of all colliding points.

C.4.3 Damping Regulation

As what is mentioned in Appendix B.3.2, during a collision with multiple contact sites, calculate the damping force using the velocity of the end-effector along the direction of the rendered force is likely to result in an under-damped system.

In this work, the overall damping force is calculated in two steps. First, we scale the damping coefficient corresponding to each colliding point by the stiffness scaling factor, **c**, acquired in section C.4.3. Second, we calculate the damping force of each colliding point, and sum them up to form the rendered damping force.

The first step is straightforward. Here we use B_i to represent the damping coefficient corresponding to the ith colliding point. Given a default relative damping coefficient B, the damping coefficient of the ith colliding point, B_i would be:

$$B_i = c_{\mathrm{i}}^{*} \cdot B$$

The corresponding damping force, $f_{\text{damping,i}}$, can be calculated using B_i and the projected velocity of the end-effector along the gradient of the penetration depth, $\hat{u}_i$. By summing all $f_{\text{damping,i}}$ up, the rendered damping force would be:

$$\begin{aligned}
\vec{f}_{\text{damping}} &= \sum_{i}^{n} \vec{f}_{\text{damping,i}} = -\sum_{i}^{n} B_{\mathrm{i}} \cdot \langle \vec{v}_{\mathrm{EE}}, \hat{u}_{\mathrm{i}} \rangle \cdot \hat{u}_{\mathrm{i}} \\
&= -\sum_{i}^{n} \left(c_{\mathrm{i}}^{*} \cdot B \right) \cdot \langle \vec{v}_{\mathrm{EE}}, \hat{u}_{\mathrm{i}} \rangle \cdot \hat{u}_{\mathrm{i}} \\
&= -B \cdot \sum_{i}^{n} c_{\mathrm{i}}^{*} \cdot \langle \vec{v}_{\mathrm{EE}}, \hat{u}_{\mathrm{i}} \rangle \cdot \hat{u}_{\mathrm{i}} = -B \cdot \vec{v}_{\text{render}}
\end{aligned}$$

It is shown that the manipulated individual damping force can be thought as forming a "regulated" penetration velocity, $\vec{v}_{\text{render}}$. This penetration velocity can then be used to calculate the tangential damping force in the same manner described in Appendix B.2.

One thing worth a mentioning is that, the way we determine the individual damping coefficient B_i is only valid under the assumption that the ith combined stiffness, K_i, is close enough to K (i.e., ρ_i^* is close enough to 1). If the combined stiffness is not well regulated and is way higher or lower than K, the system will suffer from over-damping or under-damping, respectively. This effect can be illustrated by calculating the "combined" damping along the gradient of ith colliding point:

$$B_{\text{combined,i}} = 2 \cdot \zeta \cdot \sqrt{M \cdot \left(\rho_{\mathrm{i}}^{*} \cdot K \right)} = 2 \cdot \left(\sqrt{\rho_{\mathrm{i}}^{*}} \cdot \zeta \right) \cdot \sqrt{M \cdot K}$$

Apparently, the relative damping ratio will increase when ρ_{i}^{*} is higher than 1 and decrease when ρ_{i}^{*} is lower than 1.

C.4.4 Input Force Limiting

In previous sections, we have introduced a new rendering algorithm to calculate the rendered force $\vec{f}_{\text{render}}$.. On the current system, the dynamics of the end-effector was determined by the summation of rendered force and the measured force (Fig. C.5). The dynamics of the end-effector was treated as the reference signal of the controller, and the controller drove the actuators to generate the dynamics of the dummy tool on the haptic interface.

A particular feature of admittance control systems, such as the current platform, is that it can theoretically generate infinity stiffness. According to the system structure (Fig. C.5), as long as $\vec{f}_{EE}$ is zero, the end-effector should become static regardless of the force input by the human operator, $\vec{f}_{\text{hand}}$. In other words, the human operator would perceive an infinite stiffness. In practice, it is impossible to keep the position of the dummy tool perfectly static due to the hardware limitations such as mechanical compliance, maximal output torque of the actuators, etc.

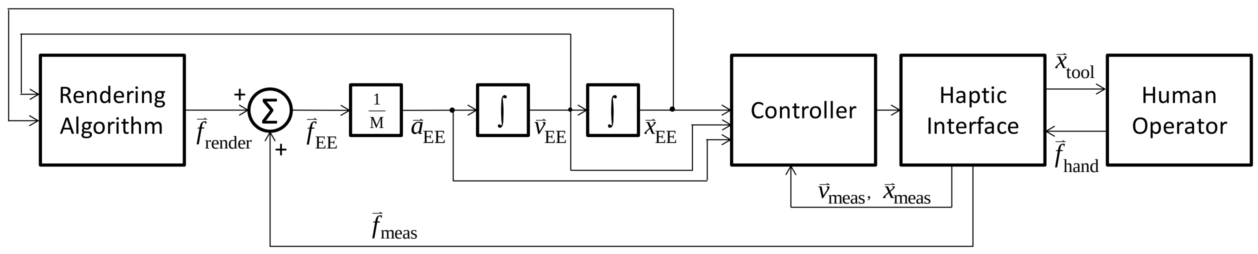


Figure C.5. The original system structure of the current platform. The dynamics of the end-effector is determined by the summation of model rendered force and measured force.

Fortunately, in our system, the maximal translational force on the interface end is 30N, which is high enough to give a perceived stiffness way higher than other impedance controlled interface. Moreover, despite the limited perceived stiffness, the static end-effector in the virtual environment would give the subject an impression of infinitive stiffness, similar to what was observed in conventional constraint-based rendering algorithm.

In order to exploit the high renderable stiffness of an admittance control system, we introduced an upper bound, f_{th}, on the magnitude of the measured force (Fig. C.6). That is:

$$\vec{f}^{\,*}_{\text{meas}} = \begin{cases} \vec{f}_{\text{meas}}, & \left\| \vec{f}_{\text{meas}} \right\|_2 \le f_{\text{th}} \\ \dfrac{\vec{f}_{\text{meas}}}{\left\| \vec{f}_{\text{meas}} \right\|_2} \cdot f_{\text{th}}, & \text{otherwise} \end{cases}$$

When the tool and the environmental object are in contact, interpenetration would induce reaction force, $\vec{f}_{\text{render}}$ and would require more input force, $\vec{f}_{\text{meas}}$, to increase the interpenetration. Once the magnitude of $\vec{f}_{\text{meas}}$ reached f_{th}, the dynamics of the end-effector would be limited so that its velocity in the virtual environment can be expressed as:

$$\vec{v}_{EE} = \begin{cases} \vec{v}_{EE}, & \langle \vec{v}_{EE}, \nabla f_{\text{render}} \rangle \leq 0 \\ \vec{v}_{EE} - \langle \vec{v}_{EE}, \nabla f_{\text{render}} \rangle \cdot \dfrac{\nabla f_{\text{render}}}{\left\| \nabla f_{\text{render}} \right\|_2}, & \langle \vec{v}_{EE}, \nabla f_{\text{render}} \rangle > 0 \end{cases}$$

In other words, when the force limit is reached, the human operator can only move the end-effector in a direction that would decrease the reaction force from the object. If the operator keeps applying force toward the direction of ∇f_{render}, the end-effector will not move and thus realize a high perceived stiffness.

The downside of this method is that if the measured force surpass the threshold, f_{th}, when the end-effector is moved in free air without collision, the human operator might feel a dragged sensation or higher inertia. According to the real-time logged data, the measured force is always less than 2N when no collision occurs (except that the human operator violently shake the dummy tool back and forth on purpose, which is not going to happen in dental operations). Therefore, we assigned a default value of 2N to f_{th} to limit the measured force throughout all tasks.

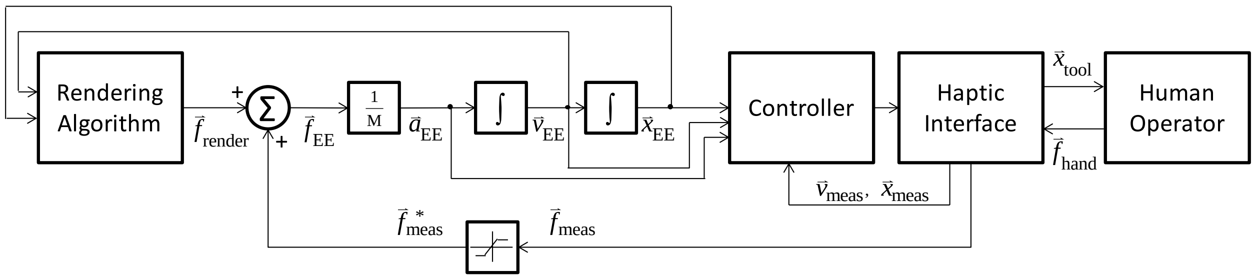


Figure C.6. The system structure with limitation on the input force. The input force, i.e. the measured force, would saturate at certain level, and thus the end-effector would stop moving beyond this point regardless how high the force is applied to the haptic interface by the human operator.

Reference

[1] W. Lorensen and H. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *ACM Siggraph Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.

[2] F. Ryden and H. Chizeck, “A method for constraint-based six degree-of-freedom haptic interaction with streaming point clouds,” *Robot. Autom. (ICRA), 2013 …*, pp. 2353–2359, 2013.

[3] W. a. McNeely, K. D. Puterbaugh, and J. J. Troy, “Six degree-of-freedom haptic rendering using voxel sampling,” *Proc. 26th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH ’99*, pp. 401–408, 1999.

[4] J. Barbic and D. James, “Six-dof haptic rendering of contact between geometrically complex reduced deformable models,” *Haptics, IEEE Trans.*, vol. 1, no. 1, pp. 39–52, 2008.

[5] D. Wang and X. Zhang, “Configuration-based optimization for six degree-of-freedom haptic rendering for fine manipulation,” *Haptics, IEEE Trans. …*, pp. 906–912, May 2013.

[6] A. Petersik, B. Pflesser, and U. Tiede, “Realistic haptic interaction in volume sculpting for surgery simulation,” *Surg. Simul. …*, vol. m, pp. 194–202, 2003.

[7] Y. Liu and S. Laycock, “A Haptic System for Drilling into Volume Data with Polygonal Tools.,” *TPCG*, vol. D, 2009.

[8] J. Colgate and G. Schenkel, “Passivity of a class of sampled-data systems: Application to haptic interfaces,” *Am. Control Conf. 1994*, pp. 3236–3240, 1994.

Appendix D – Man-Machine Experiments

In order to evaluate to what extend the proposed rendering can improve the overall performance on current platform, we have conducted two simple man-machine experiments in which the peg-in-hole scenario was used as a tasking environment. Every experiment would be conducted twice, one with the original rendering algorithm introduced in Appendix B, one with the new rendering algorithm proposed in this work (Appendix C).

D.1 General Setup

D.1.1 Platform

All man-machine experiments were conducted on the Simodont® dental trainer. The haptic interface of Simodont® is an admittance-control device. It measures the force input by the human subject and renders 3-DOF translational displacement feedback (Fig. D.1). The default updating rate of the haptic simulation is 2048Hz.

The interface is an asymmetric system that it has 6-DOF input (both position and orientation) and only 3-DOF actuator output. During the experiment, subjects were asked to grasp the dummy dental drill to interact with the virtual objects.

D.1.2 Experimental Setup

A 1.5cm×1.5cm×1cm virtual cuboid (Fig. D.2a) was built to be the environment of the experiments. A circular hole with a radius of 1mm and a depth of 5mm was located at the middle of the cuboid's top surface. The end-effector was modeled as a dental drill (Fig. D.2b), but only the stylus of the drill was haptically rendered. The cylindrical stylus had a diameter of 1mm and a length of 10mm.

During the experiment, the subject was asked to hold the dummy drill as holding a pen while look at the visual display at the front. The experiment contains two individual tasks. Before each of the task, the subjects were allowed to practice with several sessions to get familiar with the procedure of each task.

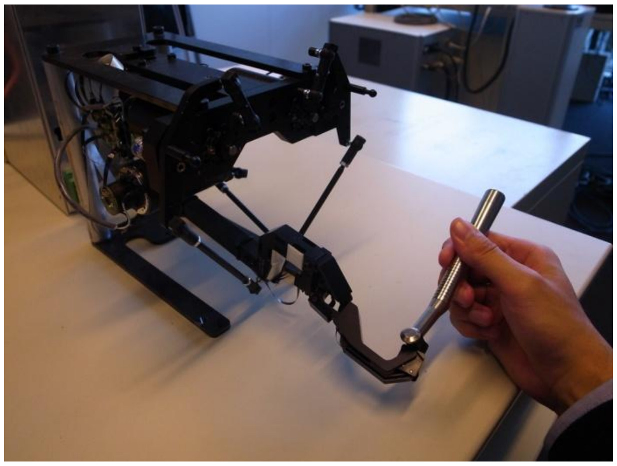


Figure D.1. The 3-DOF admittance control haptic interface on Simodont® dental trainer.

a.

b.

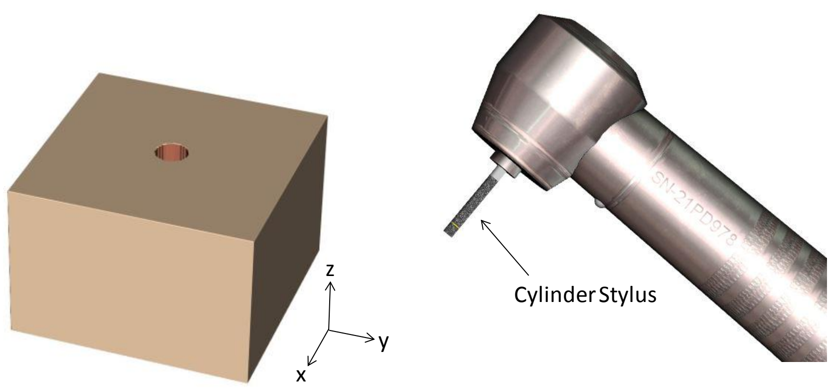


Figure D.2. (a) A 1.5cm×1.5cm×1cm virtual cuboid was built to be the environment of the experiments. A circular hole with a radius of 1mm and a depth of 5mm was located at the middle of the cuboid's top surface. (b) The end-effector was modeled as a dental drill, but only the stylus of the drill was haptically rendered. The cylindrical stylus had a diameter of 1mm and a length of 10mm.

D.1.3 Subjects

There are 6 subjects participated in both experiments, including 3 female subjects and 3 male subjects (including the author). Four of the subjects have no or small experience in operating haptic interfaces.

D.2 Vertical Peg-in-hole Task

The goal of this task is to evaluate the performance of the rendering algorithms under distributed contact scenario that only involved translational force feedback, i.e. no induced torque.

D.2.1 Experiment Setup

In this task, the orientation of the stylus was overridden and fixed to be parallel to the orientation of the hole (Fig. D.3a), and cannot be changed by the subject. Task would only be activated when the subject performed a downward force against the bottom of the hole with a magnitude more than 1N. In this manner, we could make sure that the stylus is always in contact with the hole.

There were two target positions on the bottom of the hole, one at the center, one at a distance of 0.52mm from the center (Fig. D.3c). Since the diameter of the hole and the cylinder stylus is 2mm and 1mm, respectively, the drill would only be in contact with the bottom surface of the hole (i.e. 1-DOF contact) at the central target position (Fig. D.3d). In contrast, the stylus would definitely be in contact with the side wall of the hole if the subject tried to move the stylus toward the target located 0.52mm aside the center (Fig. D.3e). The two target positions were chosen based on the result of the pilot experiments, in which we found that it was almost impossible to remove a target located more than 5.2mm aside the center of the hole with the original algorithm.

During the experiment, a circular panel was displayed on the task window to show the subject the position of the drill tip (Fig. D.5a, blue dot) and the target position (Fig. D.5a, red dot). A force level gauge was displayed on the task window to visualize the model reaction force along z-axis.

D.2.2 Procedure

Each subject was asked to complete three trials of the task, each trial contained 10 targets, 5 for each target position. The target showed up one at a time, in a pseudo-random order. Once a target showed up, the subject had 10s to remove the target by moving the stylus tip toward the target and maintaining the position for 1s. The target would disappear if it is not removed within 10s, and the next target would show up. A trail was said to be completed after all the 10 targets were removed or disappeared. The specific instruction for each subject is as follow:

"In this experiment, you are going to remove targets that have different positions on the bottom of the hole using the tip of the dental drill. The circular panel on the upper left corner visualizes the target position (red dot) and the current position of the dill's tip (blue dot). To activate the task, you will have to press the tool down against the bottom of the hole. The magnitude of the downward force will be displayed on the force gauge. You must maintain the force to be higher than the threshold throughout each task. Once a task is activated, you will have 10 seconds to remove the target by moving the drill tip (blue dot) to the target position. The target will only be removed if the drill tip stays in the red circular area for 1s continuously. Each session contains 10 targets, which will show up one at a time in a pseudo-random order."

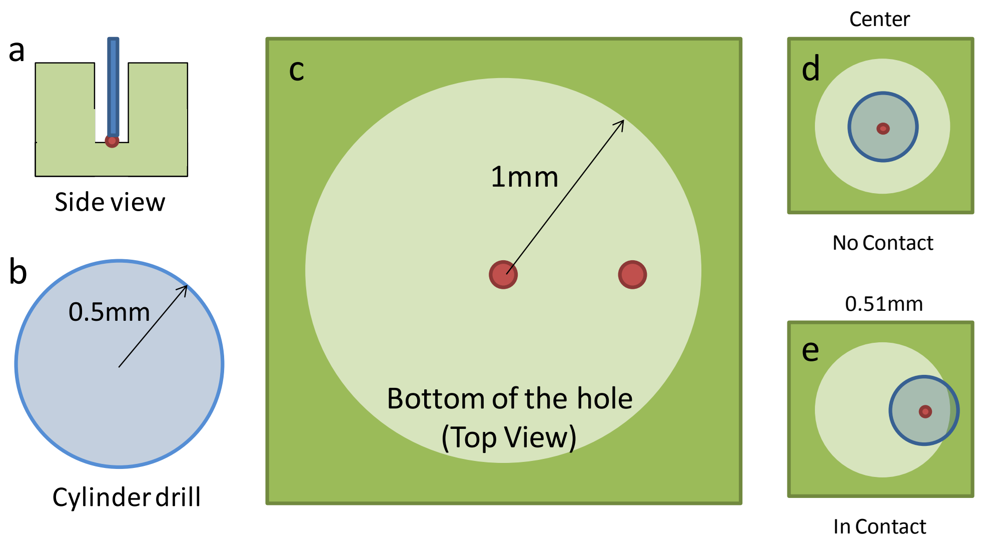


Figure D.3. (a) The orientation of the drill (blue) was fixed and always perpendicular to the bottom of the hole throughout the whole task. (b-c) The radius of the drill and the hole are 0.5mm and 1mm, respectively. Two target positions (red dot) are assigned in the experiment, one at the center and at a distance of 0.52mm aside the center. (d) The drill would only be in contact with the bottom surface of the hole (i.e. 1-DOF contact) when being moved to the central target. (e) The drill would definitely be in contact with the side wall of the hole if the subject tried to move the stylus toward the target located 0.52mm aside the center.

D.3 Rotational Peg-in-hole Task

The goal of this task is to evaluate the performance of the rendering algorithms under a distributed contact scenario that could induce reaction torque.

D.3.1 Experiment Setup

In this task, the subject was allowed to change the orientation of the end-effector manually. There were two target heights, 3mm and4mm, with respect to the bottom of the hole (Fig. D.4a). In reality, larger reaction torque should be induced in the former case. Since our system cannot generate torque feedback, contact that involves higher reaction torque would impose higher challenge to the rendering algorithm. The two target heights were chosen based on the result of the pilot experiments, in which we found that it was almost impossible to remove the a target located lower than 3mm height with the original algorithm.

A simple panel was displayed on the tasking window to show the subject the height of the drill tip (Fig. D.5b, blue dot) and the target height (Fig. D5.b, red dot), with respect to the bottom of the hole. A force level gauge was also displayed on the task window to visualize the model reaction force along z-axis.

The task would only be activated when the subject performed a downward force against the hole with a magnitude more than 1N. This guaranteed that the stylus was in contact with both the corner and the side wall of the hole, a configuration that "should" induced torque theoretically.

D.3.2 Procedure

Each subject was asked to complete three trials of the task, each trial contained 10 targets, 5 for each target height. The target showed up one at a time, in a pseudo-random order. Once a target showed up, the subject had 10s to remove the target by moving the stylus tip toward the target through rotational movement, and maintaining the position for 1s. The target would disappear if not removed within 10s, and the next target would show up. A trial was said to be completed after all the 10 targets were removed or disappeared. The specific instruction for each subject is as follow:

"In this experiment, you are going to remove targets on the side wall of the hole using the drill's tip through rotational movement. The height meter on the upper left corner will visualize the target height (red dot) and the current height of the drill's tip (blue dot). To activate the task, you will have to press the tool down against the corner and side wall of the hole. The magnitude of the downward force will be displayed on the force gauge. You must maintain the force to be higher than the threshold throughout each task. Once a task is activated, you will have 10 seconds to remove the target by moving the drill tip (blue dot) to the target height by rotating the tool. The target will only be removed if the drill tip stays in the red circular area for 1s continuously. Each session contains 10 targets, which will show up one at a time in a pseudo-random order."

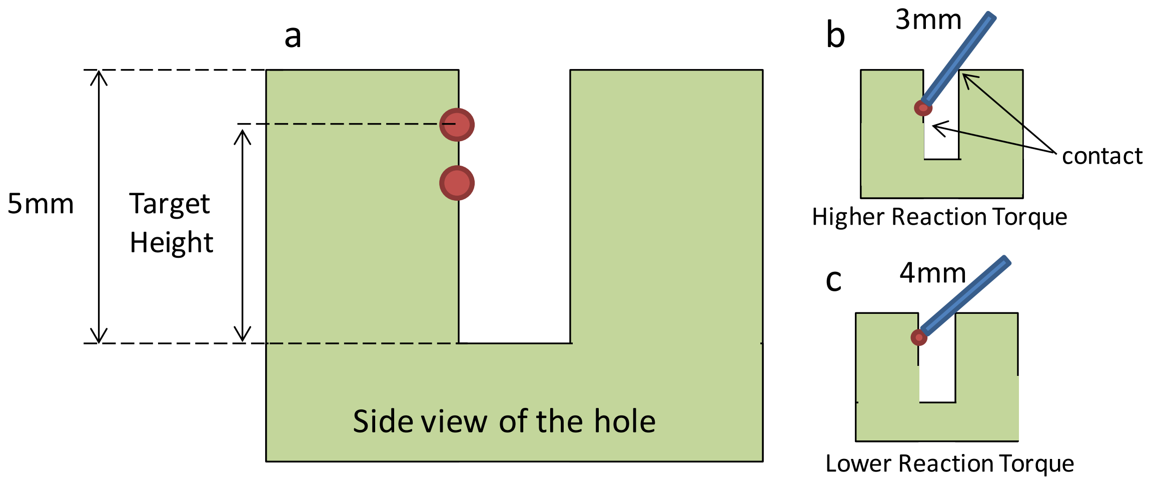


Figure D.4. (a) There are two target heights (relative to the bottom of the hole), including 3mm and 4mm. (b-c) Subject will have to manipulate the drill (blue) to make contact with the upper right corner and side wall of the hole, while track the target height by rotate the drill. The higher target will create less reduction torque compared to the lower one. Since the haptic interface cannot generate torque, the lower target would be more challenging to the rendering algorithms.

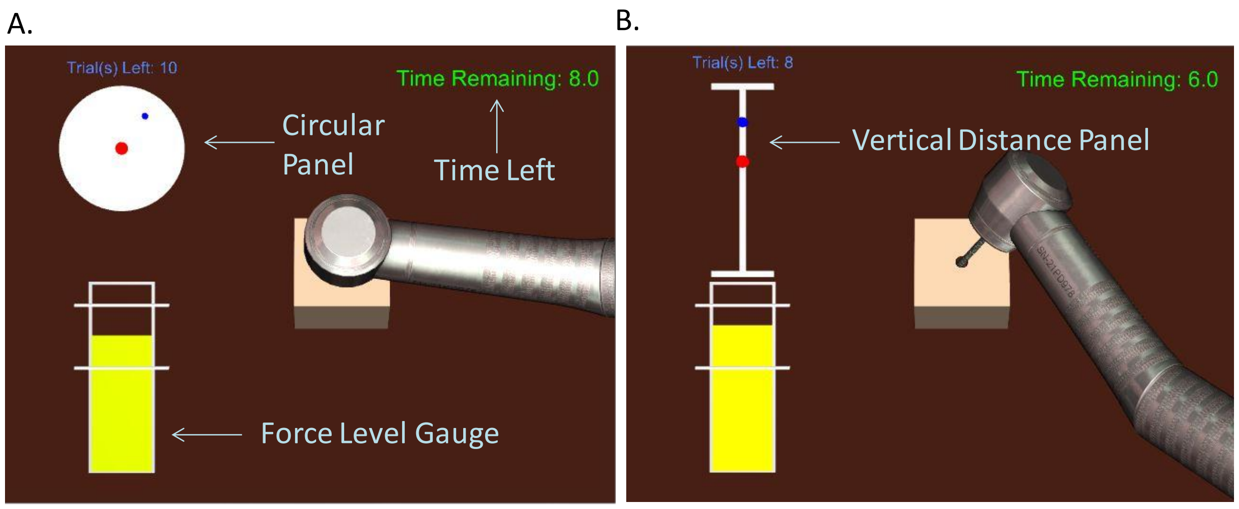


Figure D.5. The layout of the tasking window in the two man-machine experiments. A force gauge was displayed at the lower left corner. The subject must perform enough downward force to activate a task. Once a task was activated, the subject would have 10s to remove the target. The remaining time was shown on the upper-right corner. (a) In the vertical peg-in-hole experiment, a circular panel (upper-left) was shown to represent the target location (red dot) and the drill's tip position (blue dot) on the bottom surface of the hole. (b) In the rotational peg-in-hole experiment, a simple bar panel (upper-left) was shown to indicate the target height (red dot) and the drill tip's height (blue dot).

D.4 Result

In the following sections, for simplicity's sake, we denote the original algorithm with an abbreviation of "DP", and the proposed algorithm with "HS". The "DP" means "drillpoly", which is the name of the original algorithm, while the "HS" means homogeneous stiffness field, which is the essential idea of the proposed method.

We compare the performance of each rendering algorithm in terms of:

- Success rate of removing the target. A value of 1 means all targets are removed in time and a value of 0 means none of the target are removed in time.
- Completion time of removing each target.
- Model Force. The criteria of extracting the model force data from the real-time logged data is that the end-effector's position should be within a circular range on the bottom of the hole with a radius 0.2 mm from the target's position or within a small interval with a range of ±0.2mm relative to the target's height.
- Measured velocity. Same criteria for extracting valid data as the model force.

A t-test is performed to compare the mean value of each metric in every experiment.

D.4.1 Result of Questionnaires

Every subject was asked to fill in two questionnaires, a NASA-TLX form (Form D.1) and a custom made form (Form D.2). With the NASA-TLX form, we evaluate the general effort of completing the task by taking average of the value of all the six metrics [1]. The general effort was expressed in a scale from 0 to 100. A value of 100 means that the subject has to put in extremely high effort in order to accomplish the task. As for the custom form, we simply took the average among the two metrics and transform it in to a scale from 0 to 1. A value of 1 means that doing the task with the proposed method, HS, would consume way more effort than the original algorithm does. Conversely, a value of 0 means that the proposed method consumes way less effort than the original algorithm does for accomplishing the task.

The result of the NASA-TLX form (Fig. D.6, upper frames) shows that the general effort the subject has to pay during the task 1 using the original algorithm (mean=63.45, SE=8.34) is significantly higher ($t(6)=2.836$, $p<0.05$) than that using the proposed method (mean=29.24, SE=6.92). In task 2, the general effort corresponding to the original algorithm (mean=66.37, SE=7.54) is also significantly higher ($t(6)=6.657$, $p<0.005$) than that corresponding to the proposed algorithm (mean= 6.73, SE=2.96).

As for the relative effort, the result shows that, doing the task with the proposed algorithm will require less effort in both the task 1 (mean=0.083, SE=0.062) and task 2 (mean=0.067, SE=0.029) with respect to that with the original algorithm (Fig. D.6, lower frame).

[1] E. a. Bustamante and R. D. Spain, "Measurement Invariance of the Nasa TLX," *Proc. Hum. Factors Ergon. Soc. Annu. Meet.*, vol. 52, no. 19, pp. 1522–1526, Sep. 2008.

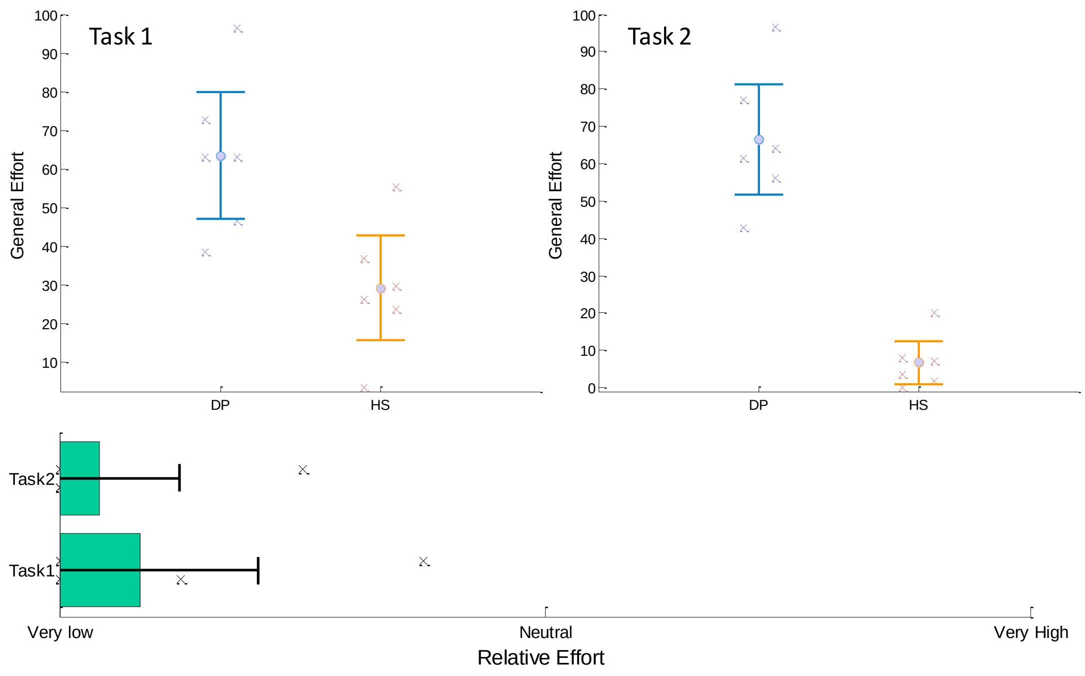


Figure D.6. The result of the NASA-LTX form (upper frames) and the custom questionnaire (lower frame). All the result indicate that the effort the subject has to put in for accomplishing the task by using the proposed algorithm is significantly lower than that using the original algorithm.

D.4.2 Logged Data in the Vertical Peg-in-hole Task

The real-time logged data shows that, when removing the side target, using the proposed method would give a significantly better performance in terms of success rate (Fig D.7a), completion time (Fig D.7b), model force (Fig D.7c), and the measured velocity (Fig D.7d). The statistical results are provided in Table A.1.

In contrast, the performance of both algorithms in removing the central target has no significant difference (Table A.2). This fits our expectation since there will only be 1-DOF contact when removing the central target, and both algorithms are stable in 1-DOF interaction. When removing the side target, however, the distributed contact created a violent vibration and sometimes rendering error when the original algorithm was used.

Table A. 1. Result of the side target (0.52mm from the center of the bottom of the hole)

	DP (mean ± SE)	HS (mean ± SE)	t score	p value
Success rate	0.32 ± 0.09	0.97 ± 0.02	-9.87	<0.005
Completion time (s)	7.78 ± 0.66	3.63 ± 0.58	6.876	<0.005
Model force (N)	2.28 ± 0.19	1.79 ± 0.15	3.121	<0.05
Measured velocity (m/s)	0.009 ± 0.001	0.002 ± 0.0001	4.991	<0.005

Table A.2. Result of the central target (at the center of bottom of the hole)

	DP (mean ± SE)	HS (mean ± SE)	t score	p value
Success rate	0.64 ± 0.13	0.63 ± 0.19	0.85	>0.5
Completion time (s)	6.09 ± 0.92	6.47 ± 1.17	-0.478	>0.5
Model force (N)	1.63 ± 0.16	1.65 ± 0.18	-0.28	>0.5
Measured velocity (m/s)	0.002 ± 0.0001	0.002 ± 0.0001	2.484	>0.05

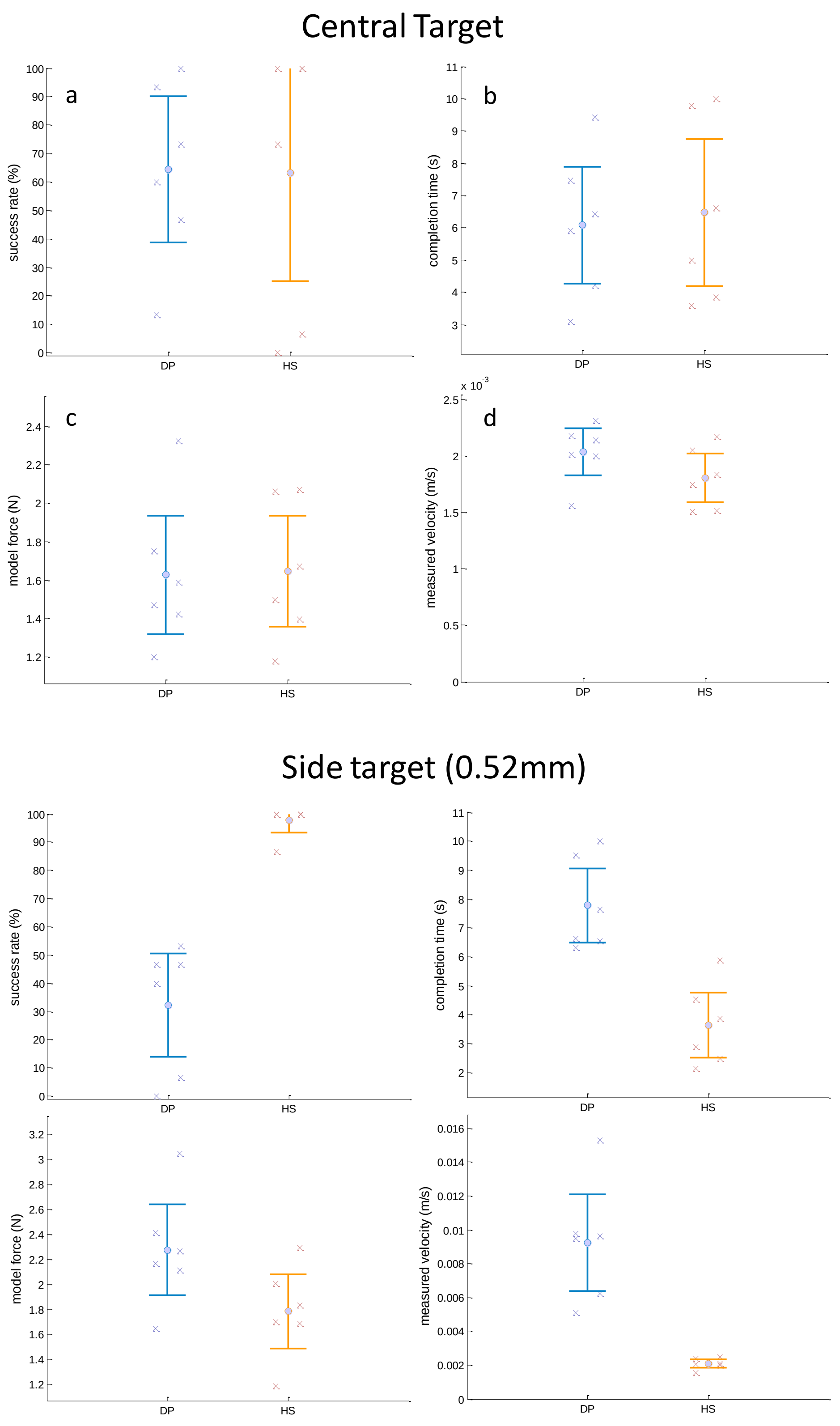


Figure D.7. The comparison of the performance in vertical peg-in-hole task when using the original (DP) and the proposed (HS) method, in terms of success rate (a), completion time (b), model force (c), and measured velocity (d).

D.4.3 Logged Data in the Rotational Peg-in-hole Task

In this case, the real-time logged data shows that, using the proposed method would give a significantly better performance than using the original algorithm, in terms of success rate (Fig. D.8a), completion time (Fig. D.8b), model force (Fig. D.8c), and the measured velocity (Fig. D.8d), regardless the target height. The statistic results corresponding to the lower and higher target are provided in Table A.3 and Table A.4, respectively.

The possible reason for the significant difference in the performance of the two rendering algorithms in both tasks could be that both targets would inherently involve reaction torque, which cannot be dealt with by the original algorithm. As a result, even if the success rate of removing the higher target is higher than that of removing the lower one, the performance of the original algorithm is still poor, compared to that of the proposed one.

Table A.3. Result of the lower target (3mm height with respect to the bottom of the hole)

	DP (mean ± SE)	HS (mean ± SE)	t score	p value
Success rate	0.004 ± 0.004	1.0 ± 0.0	-21.5	<0.005
Completion time (s)	9.7 ± 0.30	2.20 ± 0.26	23.16	<0.005
Model force (N)	3.2 ± 0.24	1.88 ± 0.21	5.219	<0.005
Measured velocity (m/s)	0.03 ± 0.004	0.002 ± 0.0004	5.676	<0.005

Table A.4. Result of the higher target (3mm height with respect to the bottom of the hole)

	DP (mean ± SE)	HS (mean ± SE)	t score	p value
Success rate	0.44 ± 0.11	0.99 ± 0.01	-5.103	<0.005
Completion time (s)	7.01 ± 0.85	2.28 ± 0.34	5.673	<0.005
Model force (N)	2.24 ± 0.11	1.83 ± 0.15	3.644	<0.05
Measured velocity (m/s)	0.014 ± 0.001	0.002 ± 0.0003	11.085	<0.005

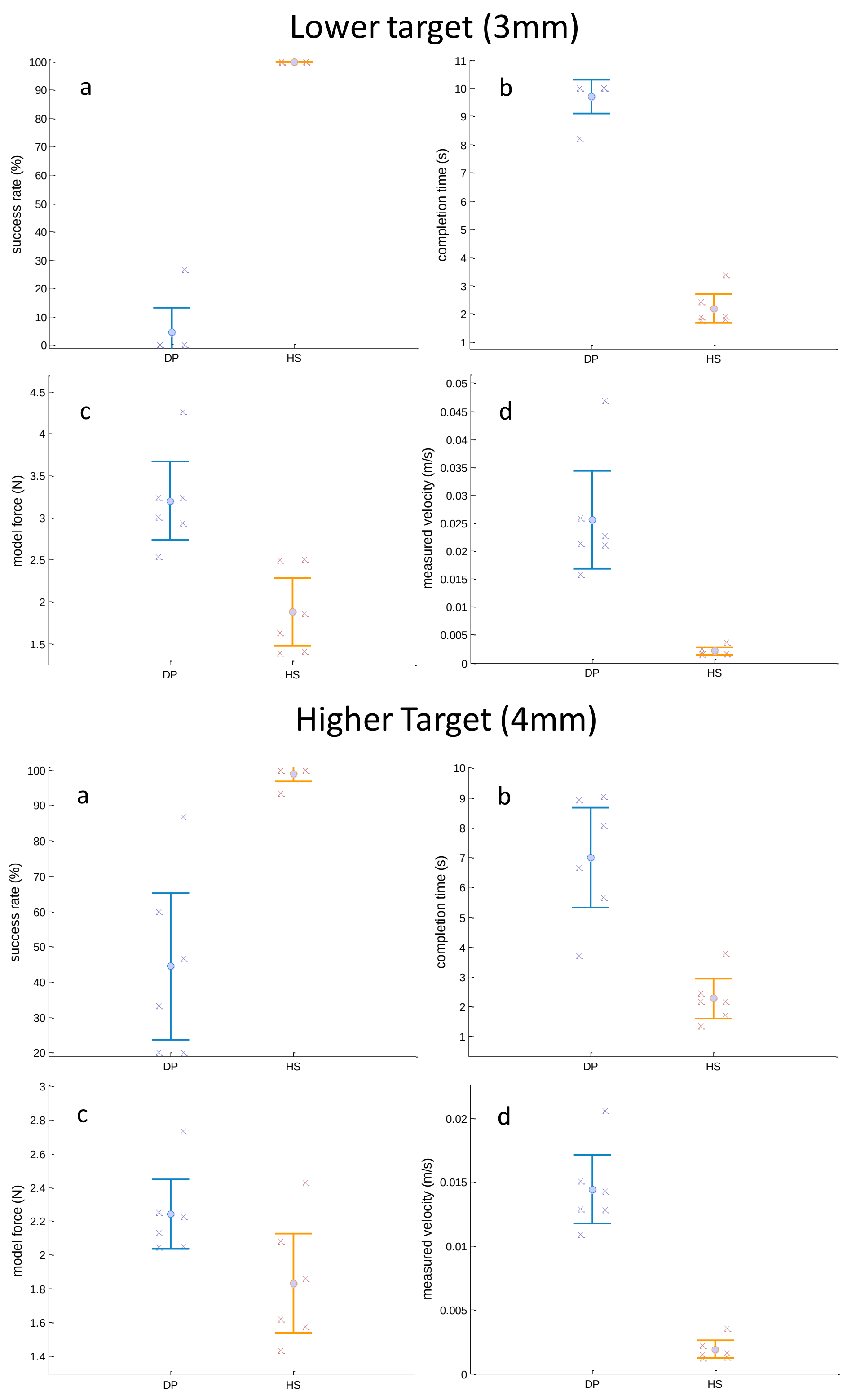
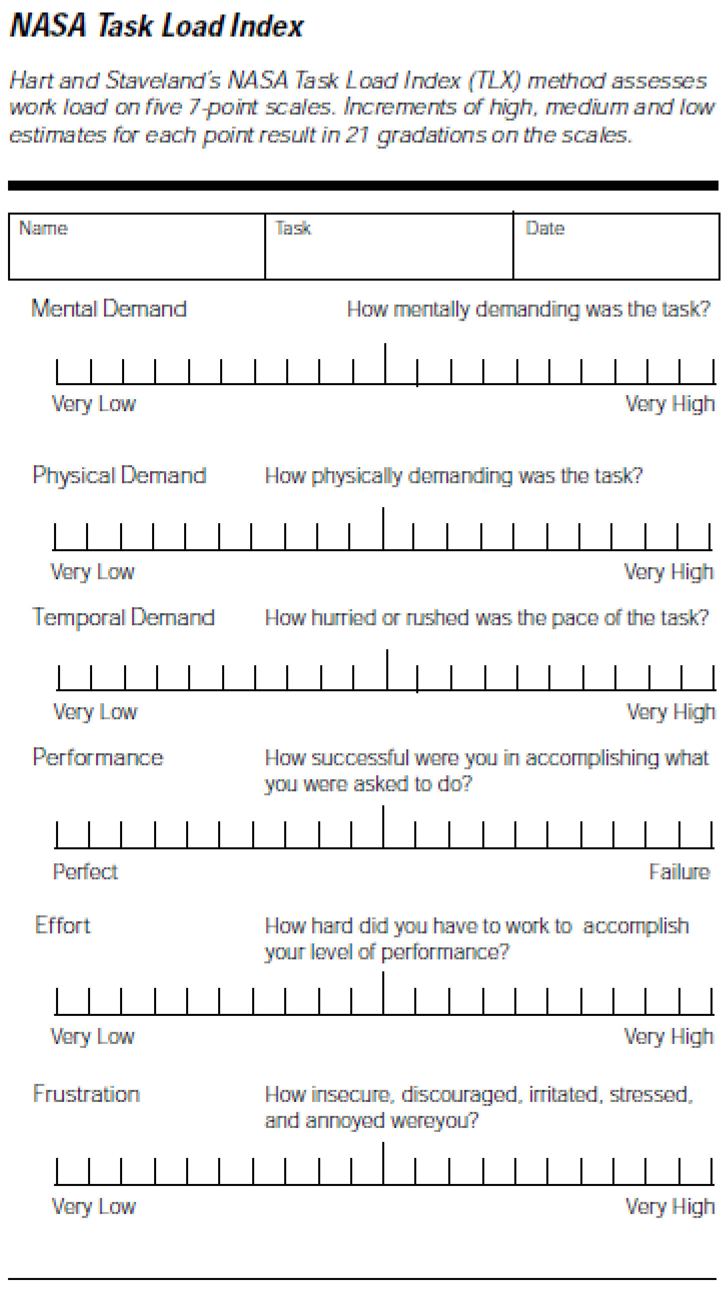


Figure D.8. The comparison of the performance in vertical peg-in-hole task when using the original (DP) and the proposed (HS) method, in terms of success rate (a), completion time (b), model force (c), and measured velocity (d).

Form D.1. *The NASA TLX form used in both man-machine experiment for evaluating the general effort.*



NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Name	Task	Date

Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?

Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?

Very Low — Very High

Form D.2. *The Custom-made questionnaire for evaluating the relative effort.*

Vertical Peg-in-hole

- **Algorithm 1** feels more realistic than **Algorithm 2**.

○	○	○	○	○
Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree

- It is more difficult to accomplish the task using **Algorithm 2** than using **Algorithm 1**.

○	○	○	○	○
Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree

Rotational Peg-in-hole

- **Algorithm 1** feels more realistic than **Algorithm 2**.

○	○	○	○	○
Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree

- It is more difficult to accomplish the task using **Algorithm 2** than using **Algorithm 1**.

○	○	○	○	○
Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree

Appendix E – More Data of Validation Test

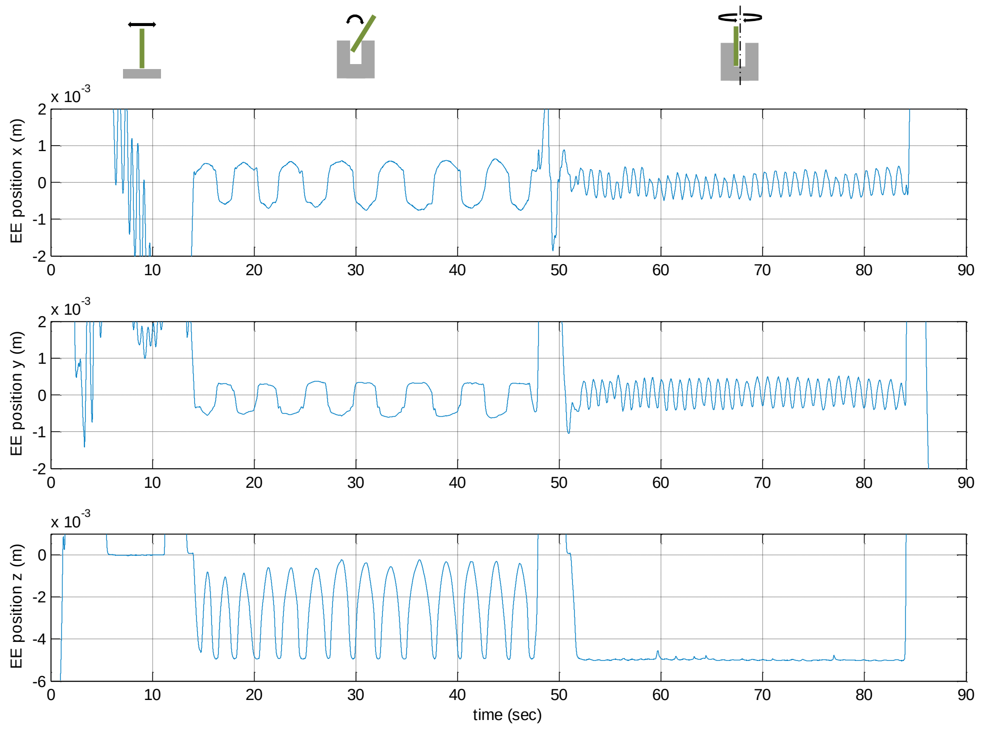


Figure E.1. The position of the end-effector as a function of time.

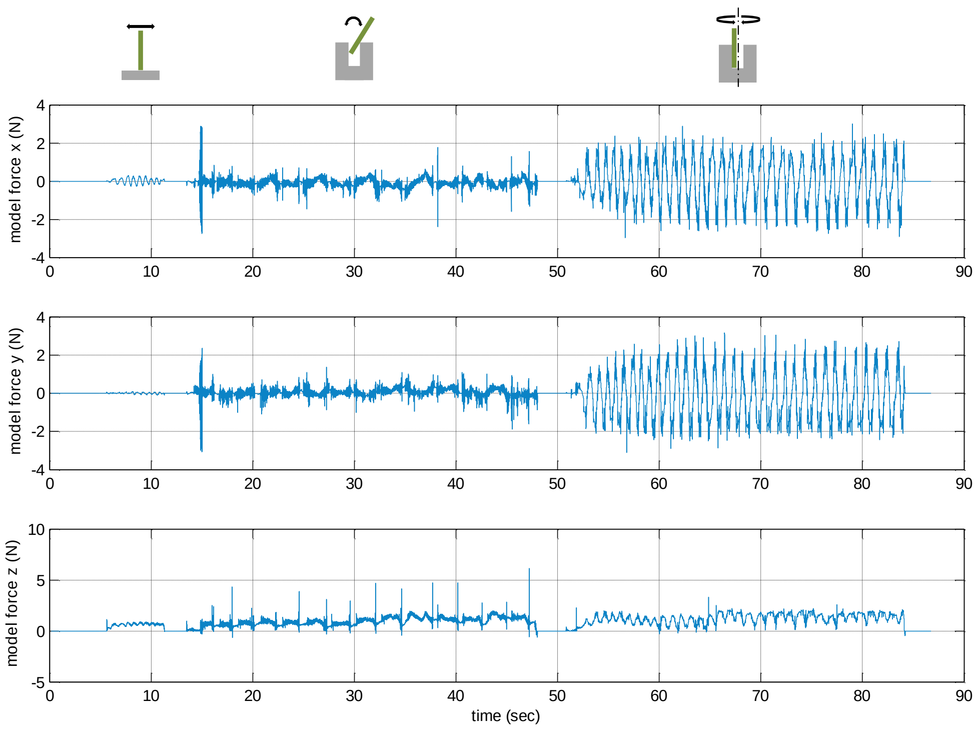


Figure E.2. The model force, i.e. the reaction force from the model, as a function of time.

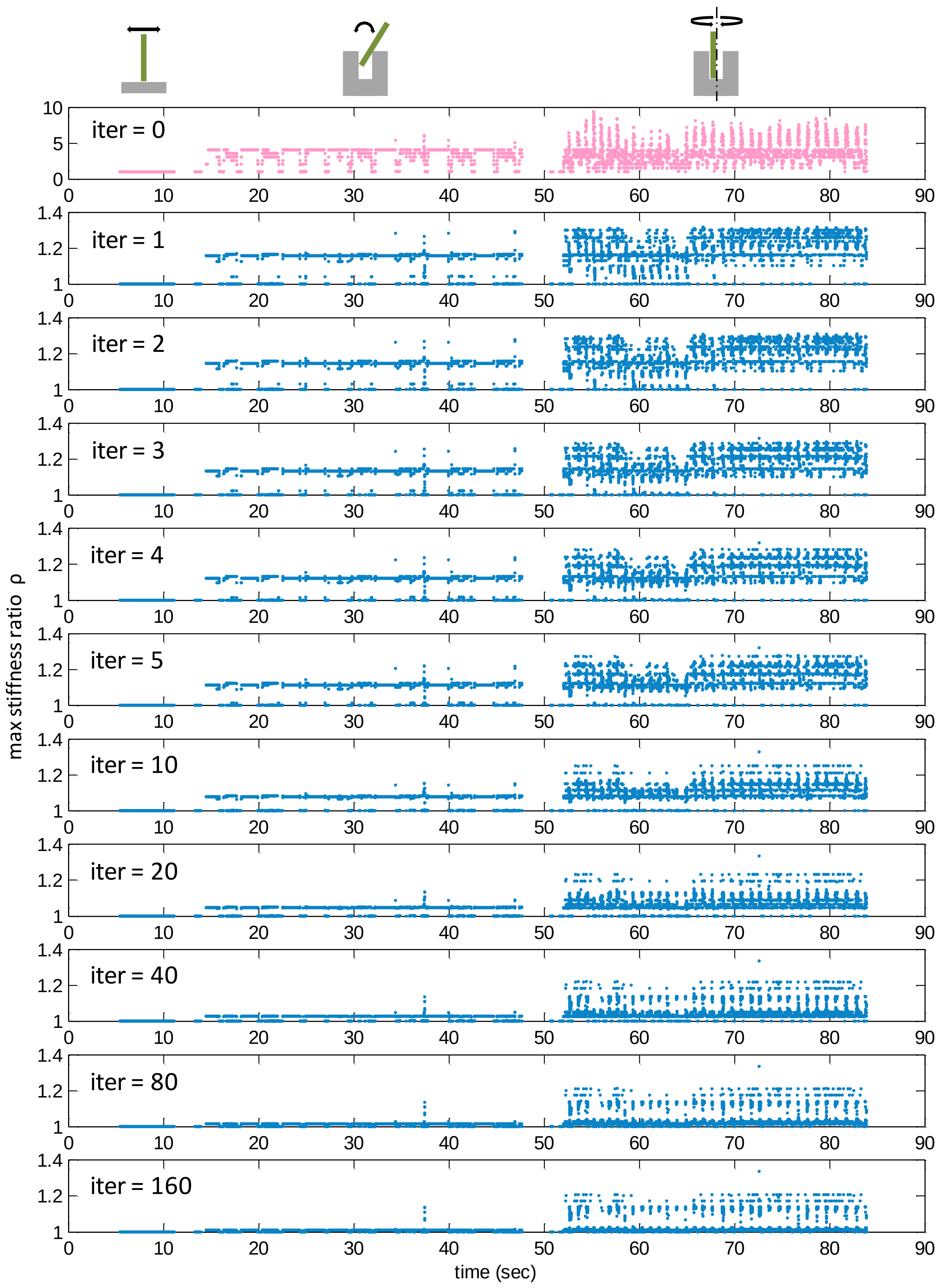


Figure E.3. The maximal combined stiffness of each iteration number as a function of time.

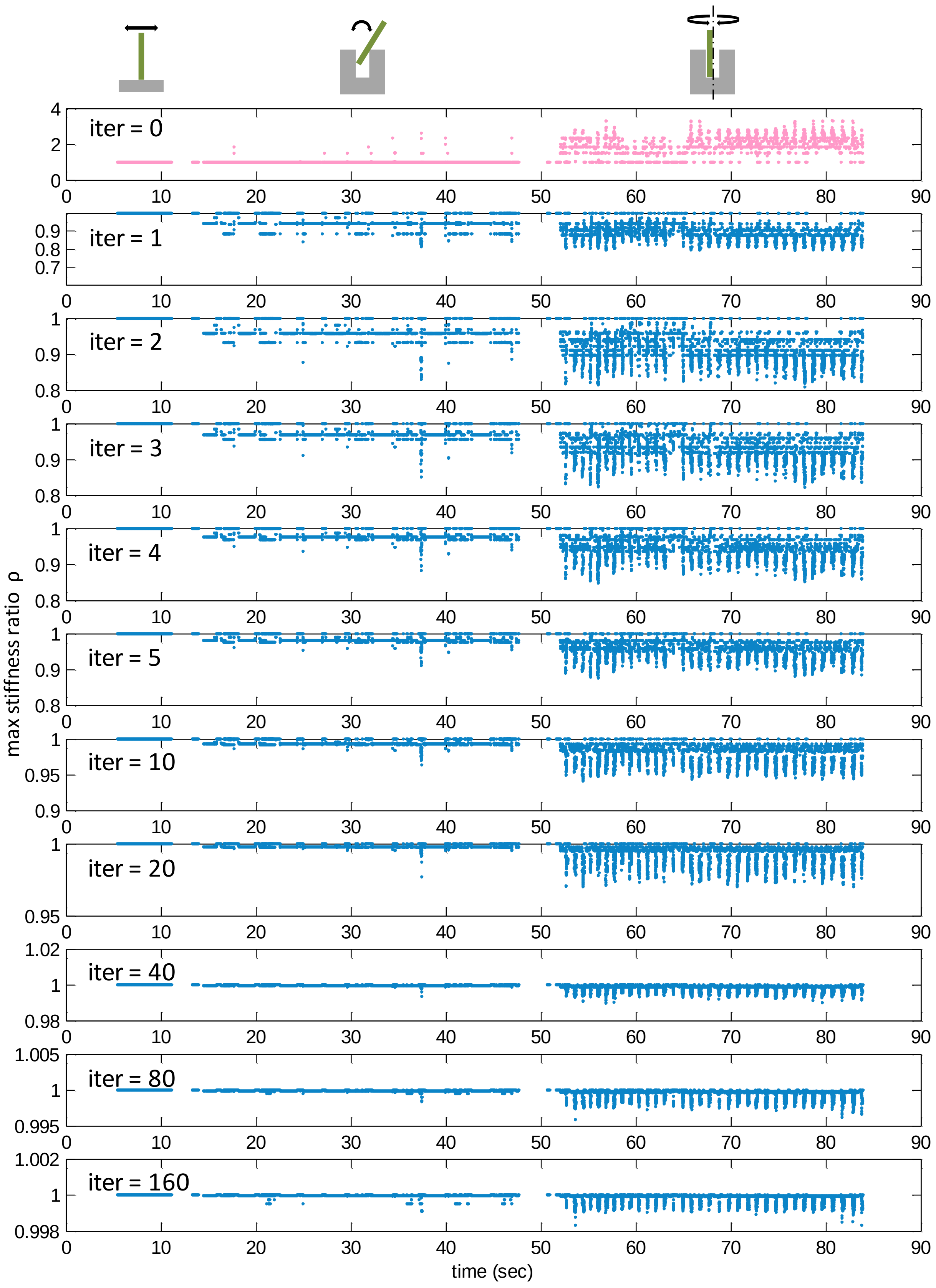


Figure E.4. The minimal combined stiffness of each iteration number as a function of time.

Appendix F – Optional Rendering Algorithms

In this section, we introduce several rendering algorithms that were developed during this work but not included into the final version of algorithm. The "normal projection" method is an alternative way to evaluate the penetration depth that it calculates the distance between the colliding point and the tool's surface along the inverse of surface normal vector. We will demonstrate how to calculate such distance when a cylindrical tool is used, and discuss the pros and cons of this method. On the other hand, the "Tandpasta method" is an alternative way to shape the combined stiffness. It was particularly designed for peg-in-hole scenario that it can boost the reaction force to push the peg out when the peg was rotated inside the hole. We will also give a short discussion on the advantages and disadvantages of this method.

F.1 Normal Projection Method

As what was mentioned in Appendix C.1, in the final version of our algorithm, we used the inverse of the surface normal vector as the gradient and the shortest distance between the colliding point and the tool's surface as the magnitude of the penetration depth. This hybrid method would nevertheless underestimate actual penetration depth (Fig. C.1).

In this section, we showed how to calculate the precise distance from a colliding to the surface of a cylindrical tool along the inverse direction of the colliding point's surface normal vector. All parameters used in this section are listed in Table F.1.

For simplicity's sake, we ignore the sub-index "i" of all parameters in the following calculation. Imagine an arrow starts from the colliding point $\vec{x}_{\mathrm{p}}$ and moves along the point's gradient, $\hat{u}$. After traveling for a certain distance, the arrow would hit the boundary of the tool. We denoted this penetrating point as $\vec{x}_{\mathrm{penetrate}}$. The arrow's traveling distance is the penetration depth, denoted as d.

$$\vec{x}_{\mathrm{penetrate}} = \vec{x}_{\mathrm{p}} + d \cdot \hat{u} \tag{F.1}$$

For a simple cylinder drill, there are three types of geometrical boundary the arrow could hit: 1) cylinder side wall, 2) tool tip's surface and 3) tool's bottom surface. We will demonstrate how to calculate the distance from the colliding point to each of these three surfaces in the following sections. In the end, the shortest penetration depth among the three will be the rendered penetration depth.

Table F.1 All parameters used in the normal projection method

$\hat{u}_i$	Gradient of the ith colliding point
$\hat{\varphi}_{\text{tool}}$	Orientation of the tool
$\vec{x}_p$	Position of the ith point with respect to the global reference frame
$\vec{x}_{p,EE}$	Position of the ith colliding point with respect to the position of the end-effector
$\vec{x}_{p,EE\_\varphi}$	Projection of $\vec{x}_{p,EE}$ along the orientation of the tool, relative to the tool origin.
$\vec{x}_{\text{penetrate}}$	The position where the point is projected on the tool surface along its gradient
$\vec{x}_{\text{penetrate},\varphi}$	The projection point of $\vec{x}_{penetrate}$ on the direction of the tool, $\vec{n}_{EE}$
$\vec{x}_{\text{penetrate,EEdistal}}$	Position of the intersection point relative to the center of the bottom surface
$\vec{x}_{EE}$	Position of the end-effector in the global reference frame
$\vec{r}_p$	A vector between the centerline of the tool and the point
$\vec{r}_n$	$\left(\vec{n}_p - \left(\vec{n}_p \cdot \vec{n}_{EE}\right) * \vec{n}_{EE}\right)$
d_i	Penetration Depth of the ith colliding point (m)
r_{tool}	Cylindrical drill's radius
l_{tool}	The length of the drill

Condition 1: The intersection point is on the cylinder's wall (Fig. F.1):

If the intersecting point is on the cylinder's wall, the shortest distance between $\vec{x}_{\text{penetrate}}$ and the middle line of the tool should be just equal to r_{tool}. Based on this fact, we can use following steps to calculate the penetration depth, *d*.

$$\vec{x}_{\text{penetrate},\varphi} = \left\langle (\vec{x}_{\text{penetrate}} - \vec{x}_{\text{EE}}), \hat{\varphi}_{\text{tool}} \right\rangle \cdot \hat{\varphi}_{\text{tool}} \tag{F.2}$$

$$\vec{x}_{\text{penetrate,EE}} = \vec{x}_{\text{penetrate}} - \vec{x}_{\text{EE}} \tag{F.3}$$

$$\left\| \vec{x}_{\text{penetrate,EE}} - \vec{x}_{\text{penetration},\varphi} \right\|_2 = r_{\text{tool}} \tag{F.4}$$

With Eq. (F.1), Eq. (F.2) and Eq. (F.3), the Eq. (F.4) can be rewritten as:

$$\begin{aligned}
\left\| \vec{x}_{\text{penetrate,EE}} - \vec{x}_{\text{penetration},\varphi} \right\|_2 &= \left\| \left(\vec{x}_{\text{p}} + d \cdot \hat{u} - \vec{x}_{\text{EE}} \right) - \left\langle \left(\vec{x}_{\text{p}} + d \cdot \hat{u} - \vec{x}_{\text{EE}} \right), \hat{\varphi}_{\text{tool}} \right\rangle \cdot \hat{\varphi}_{\text{tool}} \right\|_2 \\
&= \left\| \left(\vec{x}_{\text{p}} - \vec{x}_{\text{EE}} \right) - \left\langle \left(\vec{x}_{\text{p}} - \vec{x}_{\text{EE}} \right), \hat{\varphi}_{\text{tool}} \right\rangle \cdot \hat{\varphi}_{\text{tool}} + d \cdot \left(\hat{u} - \left\langle \hat{u}, \hat{\varphi}_{\text{tool}} \right\rangle \cdot \hat{\varphi}_{\text{tool}} \right) \right\|_2 \\
&= \left\| \left(\vec{x}_{\text{p}} - \vec{x}_{\text{EE}} \right) - \left\langle \left(\vec{x}_{\text{p}} - \vec{x}_{\text{EE}} \right), \hat{\varphi}_{\text{tool}} \right\rangle \cdot \hat{\varphi}_{\text{tool}} + d \cdot \left(\hat{u} - \left\langle \hat{u}, \hat{\varphi}_{\text{tool}} \right\rangle \cdot \hat{\varphi}_{\text{tool}} \right) \right\|_2 \\
&= r_{\text{tool}}
\end{aligned} \tag{F.5}$$

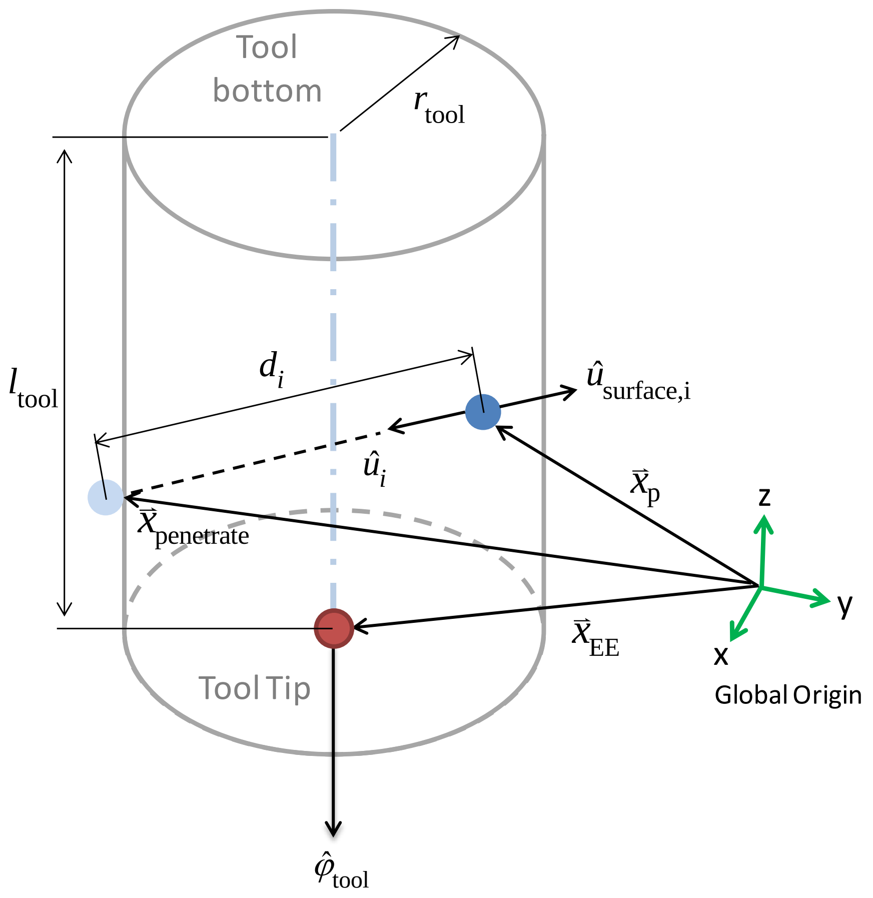


Figure F.1. The relationship between parameters when the penetration point is on the side wall of the cylindrical tool.

To make the equation neat, we defined two parameters that:

$$\vec{r}_{\text{p}} = \left(\vec{x}_{\text{p}} - \vec{x}_{\text{EE}}\right) - \left\langle \left(\vec{x}_{\text{p}} - \vec{x}_{\text{EE}}\right), \hat{\varphi}_{\text{tool}} \right\rangle \cdot \hat{\varphi}_{\text{tool}}$$
$$\vec{r}_n = \hat{u} - \left\langle \hat{u}, \hat{\varphi}_{\text{tool}} \right\rangle \cdot \hat{\varphi}_{\text{tool}}$$

In fact, the length of the vector $\vec{r}_p$ is the distance between the colliding point and the central line of the cylindrical tool. With these new parameters, Eq. (F.5) can be written as:

$$\left\| \vec{r}_{\text{p}} - d \cdot \vec{r}_{\text{n}} \right\|_2 = r_{\text{tool}}$$

To calculate the penetration depth, *d*, we take the square of both sides:

$$\left(\vec{r}_{\text{p}} - d \cdot \vec{r}_{\text{n}}\right) \cdot \left(\vec{r}_{\text{p}} - d \cdot \vec{r}_{\text{n}}\right) = \left(\left\langle \vec{r}_{\text{p}}, \vec{r}_{\text{p}} \right\rangle + 2 \cdot d \cdot \left\langle \vec{r}_{\text{p}}, \vec{r}_n \right\rangle + d^2 \cdot \left\langle \vec{r}_{\text{n}}, \vec{r}_{\text{n}} \right\rangle \right) = r_{\text{tool}}{}^2 \quad \text{(F.6)}$$

Eq. (F.6) is a typical second-order equation. The solution of variable *d* would be:

$$d = \frac{-2\cdot\langle \vec{r}_\mathrm{p}, \vec{r}_\mathrm{n}\rangle \pm \sqrt{\left(2\cdot\langle \vec{r}_\mathrm{p}, \vec{r}_\mathrm{n}\rangle\right)^2 - 4\cdot\langle \vec{r}_\mathrm{n}, \vec{r}_\mathrm{n}\rangle\cdot\left(\langle \vec{r}_\mathrm{p}, \vec{r}_\mathrm{p}\rangle - r_\mathrm{tool}{}^2\right)}}{2\cdot\langle \vec{r}_\mathrm{n}, \vec{r}_\mathrm{n}\rangle}$$

$$= \frac{-\langle \vec{r}_\mathrm{p}, \vec{r}_\mathrm{n}\rangle \pm \sqrt{\langle \vec{r}_\mathrm{p}, \vec{r}_\mathrm{n}\rangle^2 - \langle \vec{r}_\mathrm{n}, \vec{r}_\mathrm{n}\rangle\cdot\left(\langle \vec{r}_\mathrm{p}, \vec{r}_\mathrm{p}\rangle - r_\mathrm{tool}{}^2\right)}}{\langle \vec{r}_\mathrm{n}, \vec{r}_\mathrm{n}\rangle}$$

In this case, the two roots of the equation must be one positive number and one negative number, but we only consider the positive one here (i.e. only consider the intersection point in the direction of the gradient). Therefore, the penetration depth, d, is:

$$d = \frac{-\langle \vec{r}_\mathrm{p}, \vec{r}_\mathrm{n}\rangle + \sqrt{\langle \vec{r}_\mathrm{p}, \vec{r}_\mathrm{n}\rangle^2 - \langle \vec{r}_\mathrm{n}, \vec{r}_\mathrm{n}\rangle\cdot\left(\langle \vec{r}_\mathrm{p}, \vec{r}_\mathrm{p}\rangle - r_\mathrm{tool}{}^2\right)}}{\langle \vec{r}_\mathrm{n}, \vec{r}_\mathrm{n}\rangle} \qquad \text{(F.7)}$$

Condition 2: Intersection point is on the tool tip's surface (Fig. F.2)

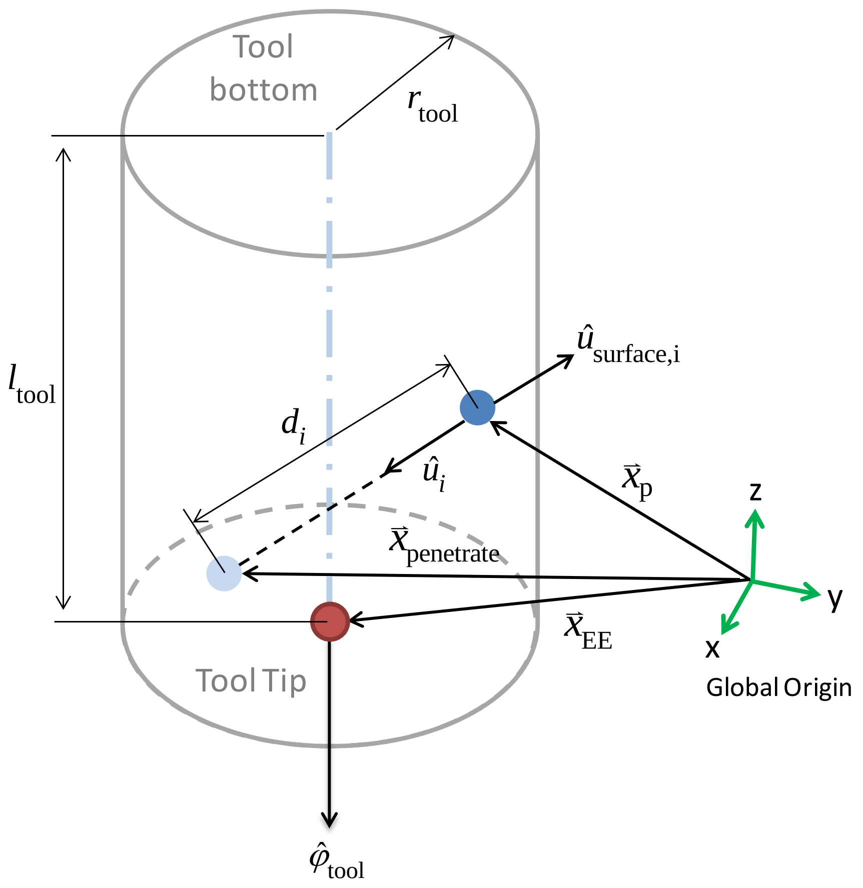


Figure F.2. The relationship between parameters when the penetration point is on the tool tips' surface.

If the intersection point is on the tool tip's surface, then the inner-product between $\vec{x}_{\text{penetrate,EE}}$ and the tool orientation, $\vec{\varphi}_{\text{tool}}$, should be zero (i.e. perpendicular to each other). Therefore:

$$\vec{x}_{\text{penetrate,EE}} \cdot \hat{\varphi}_{\text{tool}} = 0 \tag{F.8}$$

With Eq. (F.1) and Eq. (F.3), the Eq. (F.8) can be rewritten as:

$$\left(\vec{x}_{\text{p}} + d \cdot \hat{u} - \vec{x}_{\text{EE}}\right) \cdot \hat{\varphi}_{\text{tool}} = 0$$

With simple arrangement, the penetration depth can be calculated as:

$$d = -\frac{\left\langle \left(\vec{x}_{\text{p}} - \vec{x}_{\text{EE}}\right), \hat{\varphi}_{\text{tool}} \right\rangle}{\left\langle \hat{u}_{EE}, \hat{\varphi}_{\text{tool}} \right\rangle} \tag{F.9}$$

Condition 3: Intersection point on bottom surface of the cylindrical tool (distal to the tip)(Fig. F.3)

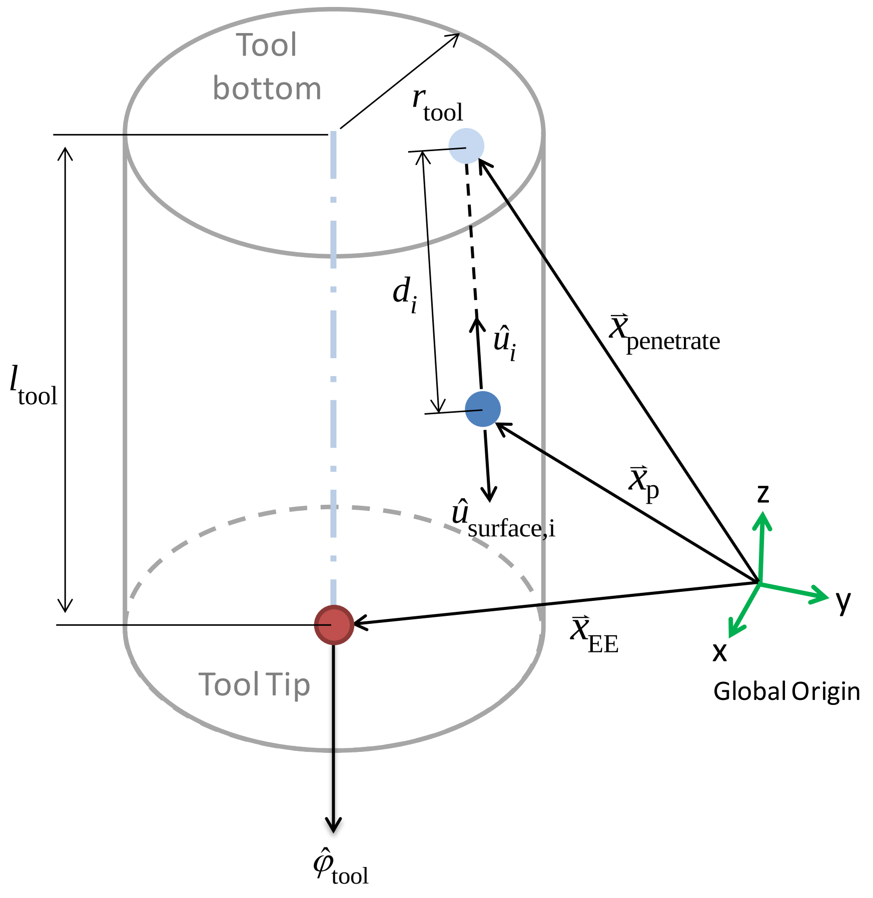


Figure F.3. The relationship between parameters when the penetration point is on the bottom surface.

First, we calculate the location of the penetration point with respect to the center of the bottom surface:

$$\vec{x}_{\text{penetrate,EEdistal}} = \vec{x}_{\text{penetrate}} - \left(\vec{x}_{\text{EE}} - l_{\text{tool}} \cdot \hat{\varphi}_{\text{tool}}\right) \tag{F.10}$$

where l_{tool} is the length of the cylinder.

Similar to last part, if the penetration point is on the bottom surface, the inner-product of the vector $\vec{x}_{\text{penetrate,EEdistal}}$, and the tool orientation, $\vec{\varphi}_{\text{tool}}$, should be zero. That is:

$$\left\langle \vec{x}_{\text{penetrate,EEdistal}}, \hat{\varphi}_{\text{tool}} \right\rangle = 0 \tag{F.11}$$

With Eq. (F.1) and Eq. (F.9), the Eq. (F.10) can be rewritten as:

$$\left\langle \left(\vec{x}_{\text{p}} + d \cdot \hat{u} - \vec{x}_{\text{EE}} + l_{\text{tool}} \cdot \hat{\varphi}_{\text{tool}}\right), \hat{\varphi}_{\text{tool}} \right\rangle = 0$$

With simple arrangement, the penetration depth can be calculated as:

$$d = -\frac{\left\langle \left(\vec{x}_{\text{p}} - \vec{x}_{\text{EE}} + l_{\text{tool}} \cdot \hat{\varphi}_{\text{tool}}\right), \hat{\varphi}_{\text{tool}} \right\rangle}{\left\langle \hat{u}, \hat{\varphi}_{\text{tool}} \right\rangle} \tag{F.12}$$

Finally, by comparing the three penetration depths calculated with Eq. (F.7), Eq. (F.9) and Eq. (F.12), we would choose the minimal positive value as the rendered penetration depth.

The advantage of this method is that, the penetration depth will not saturate as what would happen when using the shortest distance method. If the tool is pushed against a flat surface, the magnitude of the reaction force would faithfully reflect the interpenetration level between the two objects. In contrast, when the shortest distance method is used, the reaction force will saturate and stop increasing after the interpenetration has reached certain level (Fig. C.1).

However, the normal projection method could suffer from a sudden force jump due to its projection nature in calculating the penetration depth. When the tool approaches a corner, the penetration depth could suddenly show up (Fig. F.4, red arrows) and create an abrupt force that pull or push the tool away. The force was so large that it would feel like the tool is "jumping" by itself. Possible solution would be to limit the increasing rate of every individual penetration depth during sudden collision. But this cannot solve the unnatural reaction force that would occur when the tool is slowly pressed against the object's surface. In such case, the operator would perceive a "slip away" motion from the interface. Limit the maximal allowable penetration depth, on the other hand, will compromise the advantage of normal projection, that is, the constant ratio between the reaction force and the level of interpenetration depth.

On the other hand, if the level of penetration is always small, as what would happen when the input force is limited (Appendix C.4), the error between the penetration depths calculated using the hybrid shortest distance method (Appendix C.1) and the normal projection method would be negligible. In such situation, hybrid shortest distance is a better choice due to its high computational efficiency compared to the normal projection method.

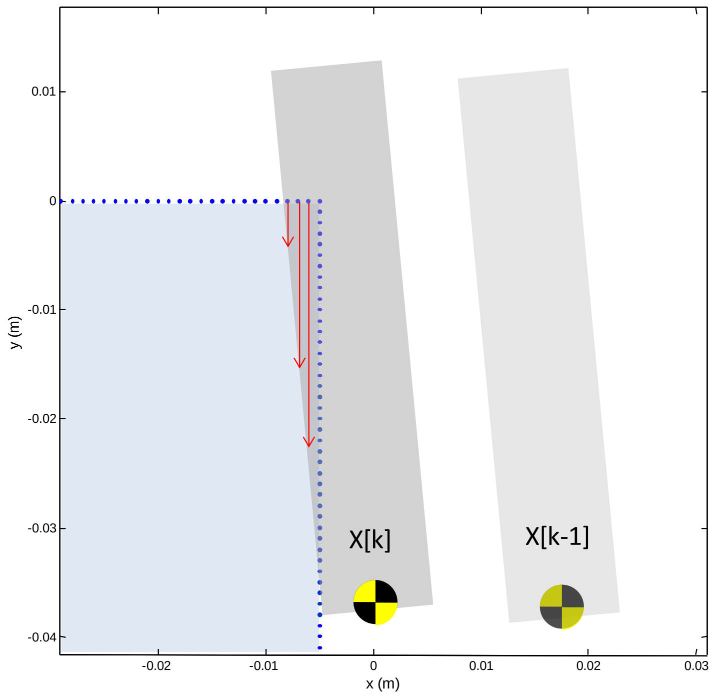


Figure F4. When the normal projection method is used, the penetration depth could increase suddenly due to the projection nature. For example, if the tool is moved toward a corner, some penetration depth can become very large in single time step (red arrows), causing an unnatural haptic feedback.

F.2 The “Tandpasta” Method

Beside stability, one of the major targets for rendering peg-in-hole task with 3-DOF haptic interface is to push the peg out of the hole. Since the 3-DOF interface cannot prevent the user from rotating the end-effector, the only way to decrease the penetration between virtual tool and environment without changing the tool’s orientation is to move it translationally until there is no more penetration. In section C.3, we introduced a stiffness shaping method that could regulate the combined stiffness along the gradient of all colliding points to a desired value of **K**. The tandpasta method would also regulate the combined stiffness on a directional basis, but it further boost the combined stiffness in certain direction to push the tool out of the hole. The idea came from the imagination of the toothpaste squeezed out when we impose pressure on the side surface of the toothpaste tube. We named this method with a name “Tandpasta”, which means toothpaste in Dutch.

In a typical peg-in-hole task, most of the colliding points belong to the side wall of the hole, while only few of them are on the upper surface of the corner (Fig. F.5a). This means that most part of the reaction force is actually pushing the peg back and forth with respect to the side wall of the hole, and the force magnitude would be so high that it could easily cause unstable result to the system.

The “Tandpasta” method aims to redistribute the elastic energy among all virtual springs, while keeps the total elastic energy unchanged. The idea is to transfer part of the elastic energy of those virtual springs that are fighting against each other (i.e. those belongs to the colliding points on the side wall of the hole) to those who encounter less or no opposite spring force (i.e. upward springs) (Fig. F.5b). This is similar to squeeze the toothpaste out of the tube, except the toothpaste is now the elastic energy while the pressure on the tube is the virtual spring force.

Suppose there are **n** colliding points in a certain time step, and each point has its corresponding gradient, $\hat{u}_i$, and we want to know to what extend the spring force are fighting against each other along each gradient $\hat{u}_i$, we can introduce a force cancelling ratio, β, which is defined as:

$$\beta_i = \frac{\left| \sum_{j=1}^{n} \langle \hat{u}_i, \hat{u}_j \rangle \right|}{\sum_{j=1}^{n} \left| \langle \hat{u}_i, \hat{u}_j \rangle \right|}, \qquad i = 1, 2, 3 \ldots n$$

The force cancelling ratio β would range from 0 to 1. A value of zero means that all spring force are cancelled out in that direction, and a value of one means that all gradients are in the same direction and no force is cancelled out. With this ratio, we can scale the stiffness of the spring corresponding to each colliding point:

$$K_i = \beta_i \cdot K$$

where **K** is the default virtual stiffness. Next, in order to make sure the total elastic energy stored in the system remains the same before and after the scaling, an energy normalization coefficient, σ_E, is introduced:

$$\sigma_E = \frac{\text{Scaled System Elastic Energy}}{\text{Original System Elastic Energy}} = \frac{\frac{1}{2}\cdot\sum_{i=1}^{n}(\beta_i \cdot K)\cdot d_i^{\,2}}{\frac{K}{2}\cdot\sum_{i=1}^{n} d_i^{\,2}} = \frac{\sum_{i=1}^{n}\beta_i \cdot d_i^{\,2}}{\sum_{i=1}^{n} d_i^{\,2}}$$

Finally, the rendered spring force is:

$$\vec{f}_{\text{spring}} = -\frac{K}{\sigma_E}\cdot\sum_{i=1}^{n}\beta_i \cdot d_i \cdot \hat{u}_i$$

The advantage of this method is that it reduced the chance of quick penetration when the tool is rotated in the hole by increasing the weighting of those spring forces that push the peg out (Fig. F.5b). This advantage, however, is also its disadvantage, in terms of feedback fidelity. In our pilot study, subject reported that the boosted outward force had made the haptic feedback unrealistic that they felt the tool jumped outside the hole by itself. In other words, the redistribution of elastic energy had made the system too "active". Another drawback of this method is that it could consume considerable computational energy, which largely limited its usefulness on a real-time system.

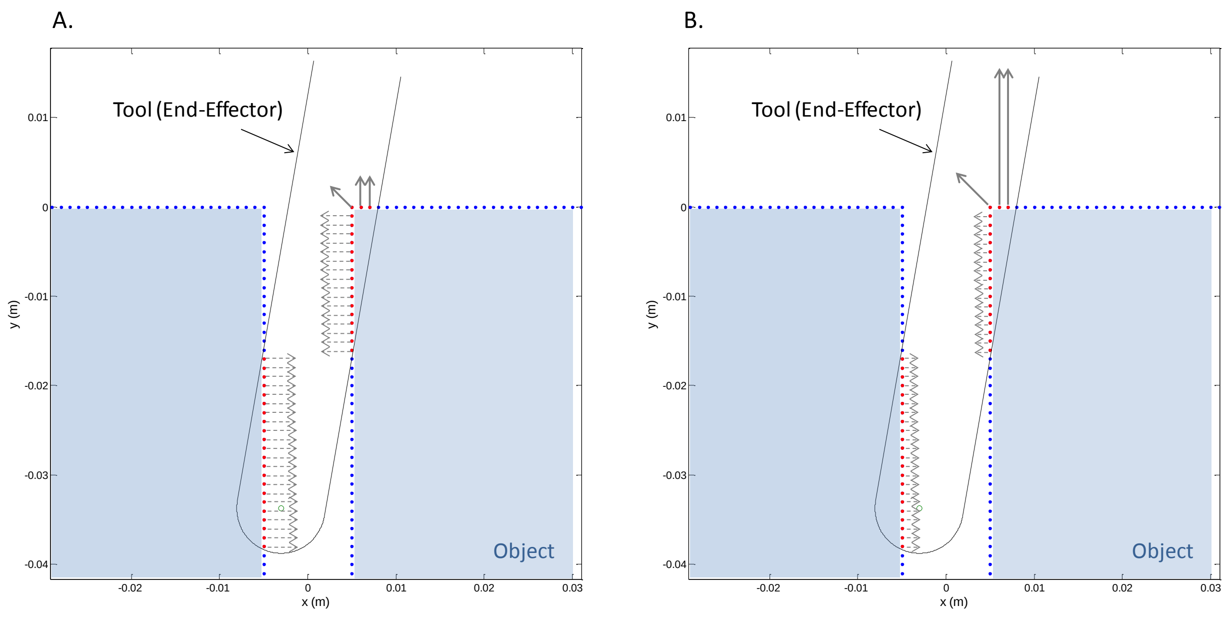


Figure F5. The visualized weighting and direction of the spring force of each colliding point (a) before and (b) after the scaling. It can be seen that the weighting of those forces fighting against each other (gray dash arrows) are decreased, and that of those forces that are pushing the peg out of the hole are increased.