2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Accelerating Optimization-based Haptic Rendering by Parallel Quadratic Programming Method

Ge Yu, Dangxiao Wang, and Yuru Zhang, *Senior Member, IEEE*

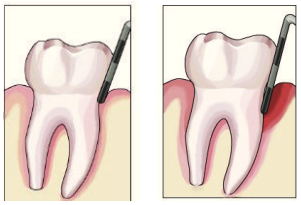
***Abstract*—It is a challenging problem to achieve fast and realistic six degree-of-freedom (DOF) haptic simulation of scenarios involving large number of multi-region contacts. In this paper, we propose an optimization-based constrained method enhanced by parallel quadratic programming to solve the rendering problem. Hierarchical sphere-tree models are used to represent the moving haptic tool and its surrounding static objects. Given a moving graphic tool as the avatar of the haptic tool in the virtual environment, we compute its quasi-static motion by solving a configuration-based optimization. Instead of using traditional active-set method, we transform the original optimization problem into its dual problem and solve the optimum about the graphic tool using a parallel quadratic programming method. Our algorithm has been implemented with a 6-DoF Phantom Premium 3.0. We validate the proposed algorithm in several benchmarks involving complex, large-region contacts. The results demonstrate that the proposed method can achieve a two to three times speed improvement than the active-set method. A further speed-up for haptic rendering may be achieved by the parallel implementation on parallel processor such as graphic processing units.**

I. Introduction

6-DoF haptic rendering, defined as the process of computing and generating forces and torques in response to user interactions with virtual objects, can greatly benefit many applications involving medical training tasks and dexterous engineering manipulation, such as dental surgical simulation and virtual assembly [1-5]. In these applications, large area/volume contacts are commonly to be interacted, such as probing the periodontal pocket depth to diagnose whether it's the healthy periodontium or with inflammation in Fig.1. When a dentist inserts the dental probe into the bottom of pocket, the working end of the probe will totally be wrapped with the tooth and gingival. Another example is a virtual assembly task as shown in Fig.2. The collision occurs between the outer surface of the splined shaft with numbers of keys and the inner surface of the splined hole with the same number of keyways, which leads to a rather large contact region. In these cases, the performances of stability and real-time for force and torque output become particularly important to simulate haptic-enabled tasks realistically.

*Research supported by the National Natural Science Foundation of China

Ge Yu, Dangxiao Wang and Yuru Zhang are with the State Key Lab of Virtual Reality Technology and Systems, Beihang University, Beijing, 100191, China. Dangxiao Wang (corresponding author to provide e-mail: hapticwang@buaa.edu.cn); Ge Yu; Yuru Zhang.



(a) (b)

Figure 1. (a) Healthy periodontium (b) Periodontium with inflammation

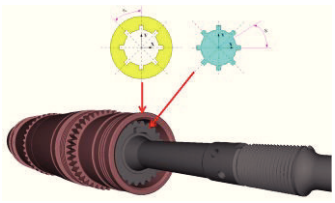


Figure 2. Virtual assembly of splined shaft and hole

A. 6-DoF Haptic Rendering Approaches

Several 6-DoF haptic rendering approaches have been proposed in recent years, which can be generally classified into two groups according to the collision response as penalty-based and constraint-based methods.

Most of existing methods are penalty-based in which the graphic tool as a dynamic object governed by Newtonian principle [2-7]. These algorithms may allow interpenetration between virtual tool and virtual environments such as the graphic tool can traverse through thin objects, or introduce some forms of virtual coupling [8] to keep feedback force stable, which results in a reducing perception of geometric details by filtering the changes of forces orientation. Compared with penalty-based methods, constraint-based methods can eliminate the haptic and visual artifacts by obtaining an exact contact configuration of the tool. Duriez et al. modeled the non-penetration and friction contact as a linear complementary problem (LCP) to solve 6-DoF haptic rendering problem [9], [10]. Based on the 3-DoF god-object approach, Ortega et al. [11] extended it into 6-DoF haptic rendering, making the generalized acceleration of the graphic tool as the variable and simulating the motion of the god-object by solving the Gauss's projection problem. These approaches could allow a more realistic force, but the high computational cost will lead to a low efficiency for detailed objects. In our previous work [12], [13], a novel configuration-based 6-DoF haptic rendering method used sphere-trees to model arbitrary objects and formulated non-penetration constraints for *active-set* based optimization. For a common model with 4681 spheres (an octree with 4 levels), if more than 100 pairs of spheres are detected being intersected, the rate of haptic loop will decrease far away from real-time feedback of 1kHz which is also not appropriate for simulating large area/volume contacts in real time. The time cost of optimization becomes the bottleneck to achieve the realistic haptic feedback.

B. Challenges

To achieve a stable and high fidelity haptic simulation in large contacts scenarios, the biggest problem to be solved is how to accelerate the optimization process for a real-time force/torque feedback.

In this paper, we extend our previous configuration-based haptic rendering approach by introducing a *PQP* (Parallel Quadratic Programming) method [14] to accelerate the optimization 6-DoF haptic rendering especially in large area contacts.

C. Organization of this paper

The remainder of paper is organized as follows: Section II illustrates the flowchart of proposed method. Section III shows how to get constraints from sphere-tree based collision detection. Section IV goes on to describe our optimization-based optimization with *PQP* solver in details, Then, Section V presents the experiment results with two demos and finally Section VI concludes the paper.

II. Flowchart of Optimization-based Haptic Rendering

A. Flowchart in haptic calculation loop

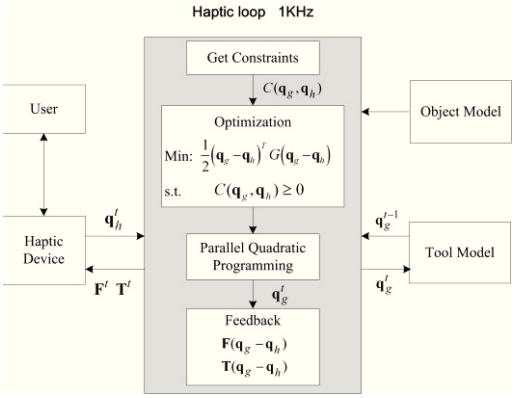


Figure 3. Framework for the proposed optimization-based haptic rendering with parallel quadratic programming.

Fig.3 illustrates the general flowchart of our proposed optimization-based 6-DoF haptic display with *PQP*. From the framework, we can first make clear the inputs and outputs of our optimization problem are:

- Given: the current configuration of haptic tool $\mathbf{q}_h^t$ and graphic tool $\mathbf{q}_g^{t-1}$ at previous time step (initialized the same as $\mathbf{q}_h^t$) where the subscript g and h represent the configuration of the desired graphic tool and the haptic tool mapping from the motion of device handle; the configuration of the object.
- Determine: the configuration of the graphic tool at current time step $\mathbf{q}_g^t$ while preventing interpenetration between the graphic tool and object; the feedback force $\mathbf{F}^t$ and torque $\mathbf{T}^t$.

The optimization-based implementation in a loop involves the following four steps: Getting constraints from collision detection, taking these constraints into an optimization problem, finding the optimum by an optimization solver and computing the feedback force and torque by the optimum.

Based on the analysis above, to obtain $\mathbf{q}_g^t$, we first setup the following optimization-based model:

$$\begin{cases} Min: \frac{1}{2}\left(\mathbf{q}_g^t - \mathbf{q}_h^t\right)^T \mathbf{G}\left(\mathbf{q}_g^t - \mathbf{q}_h^t\right) \\ subject.to: S_T \cap S_O = \varnothing \end{cases} \tag{1}$$

where

$$\begin{cases} \mathbf{q}_g^t = (x_g^t, y_g^t, z_g^t, \gamma_g^t, \beta_g^t, \alpha_g^t) \\ \mathbf{q}_h^t = (x_h^t, y_h^t, z_h^t, \gamma_h^t, \beta_h^t, \alpha_h^t) \end{cases} \tag{2}$$

The method formulates a total potential energy as a function of the difference between configurations of the haptic tool and the graphic tool. With a diagonal matrix $\mathbf{G} = diag(k_t, k_t, k_t, k_r, k_r, k_r)$ storing the three translational springs k_t and the three torsional springs k_r , to target the minimum total potential energy is our optimization objective. We choose $k_t = 1N/mm$ and $k_r = 1000mN \cdot m/rad$ in (1) to fully exploit the ability of our haptic device. Moreover, S_T and S_O denote the volumes of the graphic tool and the object respectively. The constraints aim to maintain exact contact between the graphic tool and the object, i.e. there is no penetration and no perceptible separation when contact force or torque occurs. In vector form, the multiple constraints over the combined graphic tool and haptic tool configurations can be expressed as some kind of $C(\mathbf{q}_g, \mathbf{q}_h) \geq 0$. After calculating the configuration of the graphic tool, the 6-DoF feedback force and torque can be derived from the difference between the graphic tool and haptic tool configuration by using the following model

$$\begin{bmatrix} \mathbf{F}^t \\ \mathbf{T}^t \end{bmatrix} = \mathbf{G}(\mathbf{q}_g^t - \mathbf{q}_h^t) \tag{3}$$

B. Analysis of real-time performance

There are mainly three factors influencing the performance of a configuration/optimization-based haptic rendering: the representation model, simplicity of constraint form and high efficiency of optimization solver.

We use Bradshaw's Sphere Tree Construction Tool-kit [15] to construct a medial-axis sphere octree for an object (or a tool), which contains the process to find the skeleton of an object using a voronoi diagram and creates a tight level-of-detail sphere-tree from the skeleton to fit the original triangle mesh of objects [16], [17]. The sphere model of an object supports fast collision detection and is easier to setup the constraints. In our optimization model, the biggest challenges include how to form the exact contact constraints from collision detection and how to accelerate the optimization process by using *PQP*. We will discuss these issues with more details in Section III and Section IV.

III. Construction of Non-Penetration Constraints

After establishing the tool and object sphere-tree models, we can access any sphere $s_{Ti}^L = (x_T^L, y_T^L, z_T^L, r_T)$ of the graphic tool model S_T and any sphere $s_{Oj}^L = (x_O^L, y_O^L, z_O^L, r_O)$ of the object model S_O in their own local coordinates system, where (x, y, z, r) represents the center and radius of a sphere in local coordinates, and the subscript T and O represent spheres from the tool and the object. We first denote the configuration $\mathbf{q}^t = (x^t, y^t, z^t, \gamma^t, \beta^t, \alpha^t)$ of virtual tool at the time step *t,* where (x^t, y^t, z^t) represent the position of the tool's centroid, and the rotation angle $(\gamma^t, \beta^t, \alpha^t)$ refers to the rotation angle around axis *Z*, axis *Y* and axis *X* of the world coordinate system. For collision detection in the next step, then we need to translate these local coordinates into $s_{Ti} = (x_T, y_T, z_T, r_T)$ and $s_{Oj} = (x_O, y_O, z_O, r_O)$ in global coordinates respectively under as follows:

$$s_{Oj} = \mathbf{I} \cdot s_{Oj}^L \qquad S_T = \sum_i s_{Ti} \tag{4}$$

$$s_{Ti} = \mathbf{T}(\mathbf{q}^t) \cdot s_{Ti}^L \qquad S_O = \sum_j s_{Oj} \tag{5}$$

where

$$\mathbf{T}(\mathbf{q}^t) = \begin{bmatrix} c\alpha^t c\beta^t & c\alpha^t s\beta^t s\gamma^t & c\alpha^t s\beta^t c\gamma^t + s\alpha^t s\gamma^t & x^t / r_{Ti} \\ s\alpha^t c\beta^t & s\alpha^t s\beta^t s\gamma^t + c\alpha^t c\gamma^t & s\alpha^t s\beta^t c\gamma^t - c\alpha^t s\gamma^t & y^t / r_{Ti} \\ -s\beta^t & c\beta^t s\gamma^t & c\beta^t c\gamma^t & z^t / r_{Ti} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Since S_O is fixed in the virtual environment, to be simplified, we make the configuration transformation matrix for object coordinate as an identity matrix so that both sphere coordinates of object in local coordinate and global coordinate are the same. As S_T moves as the graphic tool moves with time varying configuration transformation matrix $\mathbf{T}(\mathbf{q}^t)$, we can write s_{Ti} in a function form below:

$$s_{Ti}(\mathbf{q}^t) = (x_T(\mathbf{q}^t), y_T(\mathbf{q}^t), z_T(\mathbf{q}^t), r_T) \tag{7}$$

By given the configuration of tool $\mathbf{q}_h^t$ at time step *t*, the global coordinate of the object model or tool model can be obtained by the calculation above. Then, collision detection between the object model and tool model both represented as sphere-trees is conducted in a hierarchical fashion similar to that with the conventional discrete sphere-based bounding volume hierarchies (BVH). By checking from the two root levels of the tool and the object sphere-tree model, if the two root spheres intersect each other, intersection checking is further performed at the next lower level and repeated until either no intersection is found or all leaf sphere intersections are found. The output is a set of intersected pairs $\{s_{Ti}, s_{Oj}\}$ of leaf spheres from the object and tool model respectively. Then, the constraints can be written as:

$$(x_T - x_O)^2 + (y_T - y_O)^2 + (z_T - z_O)^2 \geq (r_T + r_O)^2 \tag{8}$$

Take (7) into (8), we can construct several quadratic constraint inequalities described by the configuration variables, so the constraints in configuration-space can be expressed as:

$$C_k(x_T(\mathbf{q}_g^t), y_T(\mathbf{q}_g^t), z_T(\mathbf{q}_g^t)) \geq 0 \qquad k = 1, ..., N \tag{9}$$

where *N* refers to the number of intersected sphere pairs.

To solve these non-linear constraints optimization is very time-consuming. Therefore, we need to find a trade-off between real-time performance and accuracy. A linearized incremental value of the graphical tool solution is proposed to formulate the constraints instead of quadratic variables. Because the change of position and orientation of the haptic device between adjacent time steps is rather small, we approximate the quadratic constraints conditions in (9) by using first-order Taylor expansion at $\mathbf{q}_g^{t-1}$ with (2) as below:

$$\begin{aligned} &C_k(x_T(\mathbf{q}_g^t), y_T(\mathbf{q}_g^t), z_T(\mathbf{q}_g^t)) \\ &= C_k(x_T(\mathbf{q}_g^{t-1}), y_T(\mathbf{q}_g^{t-1}), z_T(\mathbf{q}_g^{t-1})) + \mathbf{J}_k \cdot (\mathbf{q}_g^t - \mathbf{q}_g^{t-1}) \geq 0 \end{aligned} \tag{10}$$

where

$$\mathbf{J}_k = \begin{bmatrix} \frac{\partial C}{\partial x_T}\frac{\partial x_T}{\partial x_g^t} + \frac{\partial C}{\partial y_T}\frac{\partial y_T}{\partial x_g^t} + \frac{\partial C}{\partial z_T}\frac{\partial z_T}{\partial x_g^t} \\ \frac{\partial C}{\partial x_T}\frac{\partial x_T}{\partial y_g^t} + \frac{\partial C}{\partial y_T}\frac{\partial y_T}{\partial y_g^t} + \frac{\partial C}{\partial z_T}\frac{\partial z_T}{\partial y_g^t} \\ \frac{\partial C}{\partial x_T}\frac{\partial x_T}{\partial z_g^t} + \frac{\partial C}{\partial y_T}\frac{\partial y_T}{\partial z_g^t} + \frac{\partial C}{\partial z_T}\frac{\partial z_T}{\partial z_g^t} \\ \frac{\partial C}{\partial x_T}\frac{\partial x_T}{\partial \gamma_g^t} + \frac{\partial C}{\partial y_T}\frac{\partial y_T}{\partial \gamma_g^t} + \frac{\partial C}{\partial z_T}\frac{\partial z_T}{\partial \gamma_g^t} \\ \frac{\partial C}{\partial x_T}\frac{\partial x_T}{\partial \beta_g^t} + \frac{\partial C}{\partial y_T}\frac{\partial y_T}{\partial \beta_g^t} + \frac{\partial C}{\partial z_T}\frac{\partial z_T}{\partial \beta_g^t} \\ \frac{\partial C}{\partial x_T}\frac{\partial x_T}{\partial \alpha_g^t} + \frac{\partial C}{\partial y_T}\frac{\partial y_T}{\partial \alpha_g^t} + \frac{\partial C}{\partial z_T}\frac{\partial z_T}{\partial \alpha_g^t} \end{bmatrix}^T \tag{11}$$

We can also write the constraints in (10) as:

$$\mathbf{J}_k \cdot (\mathbf{q}_g^t - \mathbf{q}_h^t) \geq d_k \qquad k = 1, ..., N \tag{12}$$

where

$$d_k = \mathbf{J}_k \cdot \mathbf{q}_g^{t-1} - C_k(x_T(\mathbf{q}_g^{t-1}), y_T(\mathbf{q}_g^{t-1}), z_T(\mathbf{q}_g^{t-1})) - \mathbf{J}_k \cdot \mathbf{q}_h^t \tag{13}$$

To sum up, by taking (12), (13), we can rewrite the original optimization in (1) as:

$$\begin{cases} Min: \frac{1}{2}(\Delta\mathbf{q}^t)^T \mathbf{G}(\Delta\mathbf{q}^t) \\ subject.to: \mathbf{J} \cdot \Delta\mathbf{q}^t \geq \mathbf{d}, \mathbf{J} = (\mathbf{J}_1, \mathbf{J}_2 ..., \mathbf{J}_N)^T, \mathbf{d} = (d_1, d_2 ..., d_N)^T \end{cases} \tag{14}$$

where $\mathbf{J} \in R^{N \times 6}$ and

$$\Delta\mathbf{q}^t = \mathbf{q}_g^t - \mathbf{q}_h^t \tag{15}$$

IV. Solving Constrained Optimization with Parallel Quadratic Programming

The quadratic form in (14) makes the optimization problem a *QP* (Quadratic Programming) problem. Most *QP* problem solver requires long computation times at each iteration loop. Although many reports in the literature address speed-up methods [18], [19], they are heuristics without any guarantees on convergence to the global minimizers. Different with our previous work based on the *active-set* method, this paper introduced the *PQP* method to solve our optimization due to its extreme simplicity and fast convergence-two matrix-vector

products and a scalar divide, which offers considerable speed advantages than the conventional *active-set* method. Besides, *PQP* does not need to be transformed back and forth between primal and dual space, which makes *PQP* particularly efficient. More details about the *PQP* method to see [14].

Before we proceed to solve the QP problem, we need to demonstrate how to satisfy the assumptions in *PQP* first as below:

- The matrix $\mathbf{G}$ should be positive definite in (14).
- The primal quadratic programming problem in (14) should be feasible.

In our optimization model in (1) and (14), $\mathbf{G}$ is a diagonal matrix formed from the stiffness of three translational springs and three torsional springs, which meets the first conditions. The optimization objective aiming at the minimum total potential energy in (14) just satisfies the principle of minimum total potential energy, asserting that a structure or an object shall displace to a position that minimizes the total potential energy, which means there exists a solution to our *QP* problem that satisfies the constraints.

After the feasibility analysis above, we then use the *PQP* method [14] to convert the primal form in (14) into its dual form given below:

$$\begin{cases} Min: F(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T\mathbf{Q}\mathbf{y} - \mathbf{y}^T\mathbf{d} \\ subject.to: \mathbf{y} \geq 0 \end{cases} \tag{16}$$

where $y \in R^N$ is the dual variable, N refers to the number of intersected sphere pairs, and $\mathbf{Q} \in R^{N\times N}$ is also positive semi-definite with

$$\mathbf{Q} = \mathbf{J}\cdot\mathbf{G}^{-1}\cdot\mathbf{J}^T \tag{17}$$

Next, with any initial guess $\mathbf{y}^0 > 0$ (here we choose $\mathbf{y}^0 = [0.001, 0.001, ...0.001]^T$), the repeated iterations are performed by the multiplicative update rule (18) to solve the dual problem of (14) to a specified tolerance of $\Delta F(\mathbf{y})$ (we assigned 0.01 to it). The division and the *max*(a,b) operations are carried out in an element-wise manner.

$$\begin{cases} \mathbf{y}_{t+1} = \mathbf{y}_t \left[\dfrac{\mathbf{d}_t^+ + \mathbf{Q}_t^-\mathbf{y}_t}{\mathbf{d}_t^- + \mathbf{Q}_t^+\mathbf{y}_t} \right] \\ \mathbf{Q}^+ = \max(\mathbf{Q}, 0) + diag(r) \\ \mathbf{Q}^- = \max(-\mathbf{Q}, 0) + diag(r) \\ \mathbf{d}^+ = \max(\mathbf{d}, 0) \\ \mathbf{d}^- = \max(-\mathbf{d}, 0) \\ diag(r) = \{r_1, ...r_N\}, r_i = \max(\mathbf{Q}_{ii}, \sum_j \mathbf{Q}_{ij}^-) \end{cases} \tag{18}$$

Then, the optimum $(\Delta\mathbf{q}^t)^*$ of the primal problem in (14) can be recovered from the optimum $\mathbf{y}^*$ of the dual problem in (16) using the following equation:

$$(\Delta\mathbf{q}^t)^* = \mathbf{G}^{-1}\cdot\mathbf{J}^T\mathbf{y}^* \tag{19}$$

Finally, taking (19) into (15), we can obtain the desired $\mathbf{q}_g^t$ at current time step t by (20):

$$\mathbf{q}_g^t = \mathbf{q}_h^t + (\Delta\mathbf{q}^t)^* \tag{20}$$

V. EXPERIMENTS

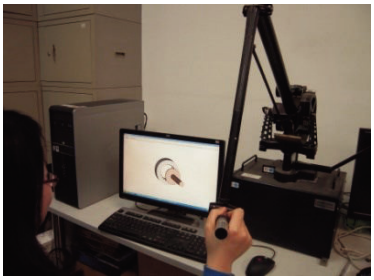


Figure 4. Experimental set up

A Phantom Premium 3.0/6DOF is utilized as the haptic device to provide 6 dimensional forces and torques. The specifications of the computer are: Intel(R) Core(TM) 2，2.20 GHz, 2GB memory, X1550 series radeon graphical card. We have conducted two experiments including a bunny-bunny interaction and a splined peg-hole interaction to validate our proposed method. Fig. 4 shows the experimental set up.

A. Bunny and Bunny

In this experiment, we use the Stanford bunny to compare the efficiency of optimization solving by the two methods: the *active-set* based method and the *PQP* based method. The golden bunny and silver bunny represent the graphic tool and the fixed object. The original triangle mesh is employed for graphic display and its sphere-trees model (level 4 with 4681 spheres for each) is used for haptic computations, including collision detection, constraint modeling and optimization.

We first perform the interaction with *active-set* [13]. In each time step, the number of intersected pairs of sphere and the time costs of both the collision detection and optimization are saved. Also, during the interaction process the six-dimensional configurations of the haptic tool are recorded and reused as the input motion to the new optimization solver with parallel quadratic programming method. Therefore, the inputs for the two optimization methods are maintained the same for the comparing the performance of the two methods.

The results are shown in Fig.5 and Fig.6. In a shallow contact such as sliding on the surface in this test (such as less than 50 intersected sphere pairs), the total time costs of the *active-set* and the *PQP* are less than 1ms, which means both methods can meet the strict requirement of 1kHz update rate of haptic loop. While large area/volume contacts occur (such as more than 100 intersected sphere pairs), the total cost of the *active-set* method increases beyond 1ms. Fig.6 clearly gives more details about their comparison. Because of the same sphere-tree models and the collision detection algorithm, the two methods take the same time on the collision detection. However the time of the optimization with *PQP* occupies only one third times than the one with *active-set*, even less than the

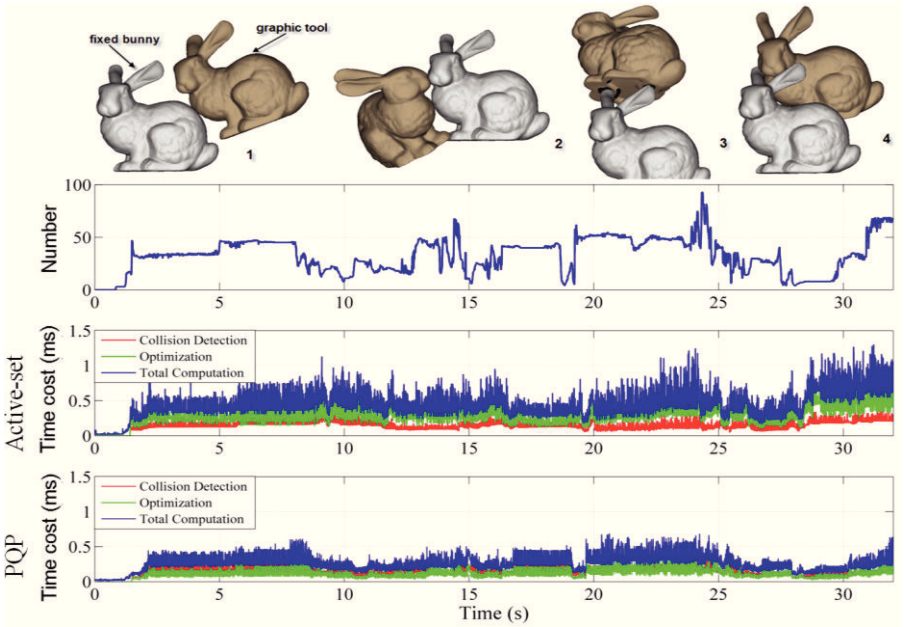


Figure 5. Experiments results of haptic simulation between a pair of bunnies. Top: four steps in contact (as the graphic tool, the golden bunny moved to map the motion of device handle and the silver bunny is a fixed one to be interacted). (a): the number of intersected pairs of sphere in each time step. (b): time cost with the active-set optimization. (c): time cost with parallel quadratic programming method.

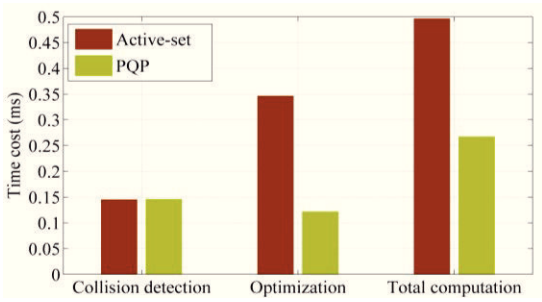


Figure 6. The average time costs of collision detection, optimization and the total computation with the two methods.

collision detection. One of the most important reasons for the acceleration comes from the simplicity in the form of the dual problem for the optimization solver, such as avoiding multiple matrix-matrix arithmetic in *active-set*. Although the speed-up of total computation is not as significant as in the optimization, the *PQP* method still makes a two times speed acceleration than *active-set* method.

The average and maximal iterations for solving (18) is six and ten to achieve the optimal solution in the proposed *PQP* based optimization. The total time cost depends more on the constraints matrix **J** rather than the initial variables.

B. Splined Shaft and Hole

In this experiment, we perform splined shaft and hole interaction using our method in the large contact scenario. To begin with, the splined shaft and hole are modeled as sphere-trees with 5 levels, and more or less 200 pairs of spheres form the constraints in optimization. The force, torque and the time cost from collision detection and optimization could be found in Fig. 7 which goes through the separate, rotating and sliding processes. When user first begin moving the shaft into the hole, the shaft usually stop in front of the hole

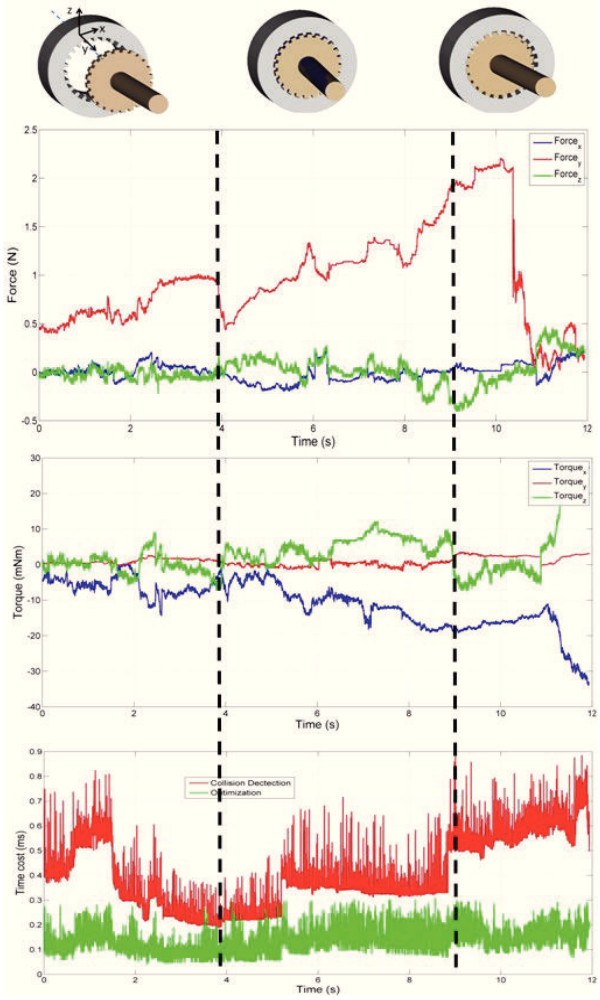


Figure 7. Force, torque and time cost of collision detection and optimization during three interactive statuses (separate, rotating and inserting)

surface first because of the existence of sharp keyways. Then, in the second stage, the user will keep rotating the splined shaft in order to perfectly align the axis of hole by subtle feel. From Fig. 6 we can see in the rotating state, the Y-directional axial force is the major one. The user will feel the axial force decrease remarkably when she/he rotates the splined peg to some angle, which means the sharp keys of splined peg and keyways of hole are totally matched. At this moment, the peg begins to insert into the hole.

In the inserting process, the peg is constrained by the sharp keyways along circular and radial direction, therefore, it can only move along the axial direction only and a litter departure from it will cause the torque constraint from x or z axis. A large contact occurs between all the keys on the shaft and the keyways on the hold at this moment. The scenario in Fig. 7 shows that no interpenetration occurs between the graphical peg and hole even at sharp keys area. Also, the total time cost remain keeping below 1kHz for a stable force and torque feedback during interaction. A noticeable thing is that the time cost of the collision detection fluctuates when the contact state between the tool and the object changes while the optimization always keep low time cost, even the collision detection overtakes more time than optimization solution sometimes. These results illustrate our proposed method is very time-efficient to solve a fast and stable 6-DoF haptic feedback in large contact scenarios.

VI. Conclusion

In this paper, we have proposed an accelerated optimization-based haptic rendering by *PQP* method allowing large contacts between two rigid bodies. There are two key components in this method. One is to formulate the constraints in 6-D configuration-space of the graphic tool from non-penetration constraints between the graphic tool and the object and the other is to introduce *PQP* method to solve the optimization by its dual solution forms.

The benefits of this paper can be summarized as follows:

1) Our proposed method forms a new 6-DoF haptic rendering solution as an alternative choice for *active-set* based QP solver to accelerate optimization. It shows to be capable to meet both requirements of accuracy and fast in multi-region contact constraints.

2) Experiment results demonstrate our method can offer a two to three times speed improvement compared with the *active-set* based optimization. With an appropriate initial value or tolerance to end iteration loop, the efficiency of our method might be further accelerated. With an increasing number of the contact pairs, a promoted efficiency of the proposed method could be more significant than the *active-set* method.

In the future, we will try to implement our method with GPU for better performance. Moreover, we consider about extending the accelerated optimization-based haptic rendering by removing redundant contact constraints based on the geometric topology features of the two contact objects, which is hopefully to further reduce the amount of contact constraints for realizing faster optimization.

Acknowledgment

The authors wish to thank Xin Zhang and Yu Zou for the contributions in this paper. This work is supported by the National Natural Science Foundation of China under the grant No. 61170187 and 61190125.

References

[1] M. A. Otaduy, C. Garre and M. C. Lin, "Representations and Algorithms for Force-Feedback Display," *Proc. IEEE.*, to be published.

[2] D. Wang, Y. Zhang, "Haptic rendering for dental training system," *Science in China Series F: Information Sciences*, vol. 52, no. 3, pp. 529-546, Mar. 2009.

[3] M. A. Otaduy and M. C. Lin, "A Modular Haptic Rendering Algorithm for Stable and Transparent 6-DoF Manipulation," *IEEE Trans. Robotics*, vol. 22, no. 4, pp. 751–762, Aug. 2006.

[4] W. McNeely, K. Puterbaugh, and J. Troy, "Voxel-Based 6-DOF Haptic Rendering Improvements," *Haptics-e*, vol. 3, no. 7, 2006.

[5] E. David, Johnson, P. Willemsen, and E. Cohen, "Six Degree-of-Freedom Haptic Rendering Using Spatialized Normal Cone Search," *IEEE Trans. Visualization and Computer Graphics*, vol. 11, no. 6, pp. 661-670, Nov-Dec. 2005.

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

[7] M. Wan, and W. McNeely, "Quasi-static approximation for 6 degrees-of-freedom haptic rendering", in *Proc. IEEE Visualization Conf.*, Seattle, Washington, 2003, pp. 257-262.

[8] E. J. Colgate, M. C. Stanley, and M. J. Brown, "Issues in the haptic display of tool use," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS'95)*, Pittsburgh, 1995, pp.140-145.

[9] C. Duriez, F. Dubois, A. Kheddar, and C. Androit, "Realistic haptic rendering of interacting deformable objects in virtual environments," *IEEE Trans. Visualization and Computer Graphics*, vol.12, no.1, pp.36 -47, Jan-Feb. 2006.

[10] C. Syllebranque, and C. Duriez, "Six degree-of freedom haptic rendering for dental implantology simulation," in *Proc. ISMBS Conf.*, 2010, pp.139-149.

[11] M. Ortega, S. Redon, and S. Coquillart, "A six degree-of-freedom god-object method for haptic display of rigid bodies with surface properties," *IEEE Trans. Visualization and Computer Graphics*, vol.13, no. 3, pp. 458-469, May-Jun. 2007.

[12] G. Yu, D. Wang, Y. Zhang and X. Zhang, "Six Degree-of-Freedom Haptic Simulation of Sharp Geometric Features using a Hybrid Sphere-tree Model," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS' 12)*, Faro, Portugal, 2012, pp.3314-3319.

[13] D. Wang, S. Liu, X. Zhang, Y. Zhang and J. Xiao, "Configuration-based Optimization for Six Degree-of-Freedom Haptic Rendering for Fine Manipulation," *IEEE Trans. Haptics.*, vol.6, no.2, pp.167-180, 2013.

[14] M. Brand, V. Shilpiekandula and C. Yao, "A Parallel Quadratic Programming Algorithm for Model Predictive Control," in *Proc. IFAC World Congress*, 2011, pp.1013-1039.

[15] G. Bradshaw. Sphere-Tree Construction Toolkit. http://isg.cs.tcd.ie/spheretree/, February 2003.

[16] P. M. Hubbard, "Approximating polyhedral with spheres for time-critical collision detection," *ACM Trans. Graphics,* vol.15, pp. 179–210, Jul. 1996.

[17] G. Bradshaw and C. O'Sullivan, "Adaptive Medial-Axis Approximation for Sphere-Tree Construction," *ACM Trans. Graphics*, vol.23, no.1, pp.1-26, Jan. 2004.

[18] R. Milman and E. J. Davison, "A fast mpc algorithm using nonfeasible active set methods," *Journal of optimization theory and applications*, vol.139, pp.591-616, 2008.

[19] S. Richter, C. Jones, and M. Morari, "Real-time input-constrained mpc using fast gradient methods," in Proc. IEEE Conf. Decision and Control, Shanghai, China, 2009, pp. 7387-7393.