2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

GPU-based Proximity Query Processing on Unstructured Triangular Mesh Model

Kit-Hang Lee, Ziyan Guo, Gary C.T. Chow, Yue Chen,
Wayne Luk, *Fellow, IEEE* and Ka-Wai Kwok, *Member, IEEE*

***Abstract*—This paper presents a novel proximity query (PQ) approach capable to detect the collision and calculate the minimal Euclidean distance between two non-convex objects in 3D, namely the robot and the environment. Such approaches are often considered as computationally demanding problems, but are of importance to many applications such as online simulation of haptic feedback and robot collision-free trajectory. Our approach enables to preserve the representation of unstructured environment in the form of triangular meshes. The proposed PQ algorithm is computationally parallel so that it can be effectively implemented on graphics processing units (GPUs). A GPU-based computation scheme is also developed and customized, which shows >200 times faster than an optimized CPU with single core. Comprehensive validation is also conducted on two simulated scenarios in order to demonstrate the practical values of its potential application in image-guided surgical robotics and humanoid robotic control.**

Index Terms – Graphics processing units (GPUs), haptic feedback, proximity queries (PQs), robot motion planning

I. Introduction

Proximity Query (PQ) is a process to request for the relative configuration or placement among 3D objects. This PQ computational problem has been widely investigated and is fundamental to many applications in fields of robot motion planning, virtual prototyping, haptics rendering, computer graphics and animation. The demand of efficient and fast PQ is mainly driven by the trend to having online simulations of high-fidelity haptic feedback at rate of >1kHz [1], or real-time generation of continuous and collision-free trajectory for safe robotic manipulation [2]. Poor PQ performance can be the major bottleneck for developing various robotic control schemes even with sufficient sensing data available [3]. One of the typical examples can be found in image-guided surgical robotics, e.g. Virtual Fixtures [4] and Active Constraints [5, 6], of which the control concept is to impose force feedback based on anatomical model acquired by imaging data.

Broad-phase PQ involves checking whether two objects are potentially touched or collided with each other. Bounding volumes in primitive shapes, such as box (e.g. AABB [7] or OBB), sphere [8] and torus [9], are commonly used to tightly enclose the object for further detection of object collision. These techniques have been extended to narrow-phase PQ, which refers computation of the minimal Euclidean separation or penetration depth when they are intersected; however, the non-convex objects that are complex in shape may neither readily be bounded nor partitioned by the boxes and spheres. Convex decomposition of the non-convex object is not efficient and also known as a NP-hard problem [10].

Triangular mesh is commonly used in representing 3D objects. Lin-Canny [11, 12] and Voronoi-Clip (V-Clip) [13] algorithms are typical narrow-phase PQ approaches. They exploited Voronoi regions to determine the closest features pair such as vertices, edges and faces between two convex polyhedral meshes. The generation of Voronoi regions of both object features is computationally expensive process due to the complex data structure required. These approaches are particularly not appropriate for objects of which their geometric structure is rapidly changed or deformed time-by-time. Gilbert-Johnson-Keerthi Algorithm (GJK) [14, 15] is a rather efficient method without having to perform any data pre-processing. The overall real-time computational performance of this iterative minimization method is sensitive to the initial guess of the closest feature pair. Chakraborty *et al*. [16] adopted Interior Point Algorithm [17] of which the iterative process is sufficiently fast and efficient for providing high-frequent PQs; however, it required the objects to be represented by implicit surface functions that act as a set of inequality constraints subject to the minimization search.

To guarantee the global convergence, the aforementioned approaches can only be applied on convex objects, thus hampering their practical values in dynamic scenarios where unstructured and irregular geometry details are involved. Although some of these approaches (e.g. [13-16]) may ensure low Big-O complexity in coping with large sampling, their computational parallelism is forgone in the algorithmic design. The inherent limitation is that these iterative minimization approaches (e.g. GJK) cannot take advantage of using parallel computing architecture which has become a trending solution to processing large and complicated date set[2]. In our previous work, we have demonstrated the superior computational performance of PQ on large amount of cloud points using GPU [18] and FPGA [19, 20].

In this paper, we derive a PQ formulation which allows for real-time computation of the shortest distance between contour segment and irregular-shape mesh model. We suggest that multiple series of contour segments enclosing the robotic manipulator can be readily updated and moved based on its kinematics chain. Moreover, the form of triangular meshes modelling the unstructured and dynamic environment can still be preserved. Furthermore, we exploit GPU-based parallel

K.H. Lee, Z. Guo, Y. Chen, and K.W. Kwok are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong; (e-mail: brianlkh@hku.hk).

G.C.T. Chow, and W. Luk are with the Department of Computing, Imperial College London, London, SW7 2AZ, U.K.

computing techniques in support high-frequency and low-latency PQ processing, thereby fulfilling many standard real-time requirement (>1kHz). Detailed validation is conducted on two scenarios of robotic tasks in extremely different natures of application.

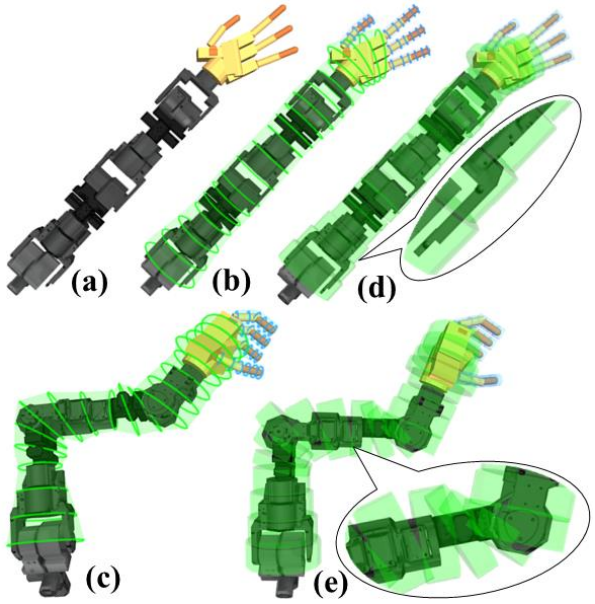

Fig.1. (a) CAD model of the ATLAS robot arm and the Sandia Robotic Hand; (b) Volumetric pathways comprising segments (in green) tightly enclosing along the arm and four fingers; (c) The corresponding twenty series of segments, of which the contours can be posed flexibly along with the subsequencially updated robot configuration; (d) Collision model approximated by the union of same number of standard bounding cylinders; (e) Such collision model fails to fully enclose the robot arm under alternative configuration.

II. Background

The motivation of addressing the aforementioned technical challenges has led to the development of novel PQ algorithm, which aims to determine the shortest distance between the robot manipulator and its surrounding environment constraints with high speed and efficiency. This algorithm is the logical progression based on the work proposed by Kwok *et al* [18], which calculates the shortest distance between an arbitrary point $x \in \Re^{3\times 1}$ and the enclosing segment. In this study, the PQ formulation for analytical calculation of shortest distance between contour segments (robot) and triangular meshes (environment) is explored.

Instead of performing PQ with an exact robot model (**Fig.1a**), our approach can greatly improve the computational efficiency by establishing the tight enclosure [21] as **Fig.1b-c** as compared to the standard bound cylinders (**Fig.1d-e**). Such approach also enables parallelization by discretizing robot segment and mesh. The pathway of robot segment, which has been applied to various robotic kinematic configurations, can be approximately defined as a centerline along the manipulator consisting of a finite chain of line segments $\overrightarrow{P_j P_{j+1}}$ $(j = 1,2,\cdots,N_c - 1)$, N_c is the total number of nodes) with their node at point P_j and tangent of M_j. A single enclosing segment Ω_j (**Fig.2a**) of the pathway comprises two adjacent circles C_j and C_{j+1}. Each circle C_j has its center at point P_j and lies on the plane normal to tangent M_j. The radius R_j of circle C_j is chosen such that the segments can be tightly fitted along the model of robotic segments. The resolution of segments is related to the interval of nodes Δ, where $\Delta = \left|\overrightarrow{P_j P_{j+1}}\right|$. The lower the interval Δ is, the higher the resolution increases with the expense of extra computation effort.

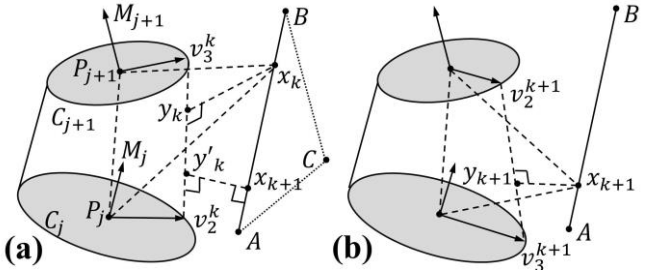

Fig. 2. (a) Basic structure of a single segment Ω_j enclosed by two adjacent circular contours. Mathematics variables introduced for geometric analysis; (b) New variables are estimated after the previous iterative result.

Triangle mesh is one of the most common representations of 3D irregular geometry in computer graphics and virtual reality [22]. The mesh could be either constructed by using robot sensors or based on the synthetic model. Many tessellation algorithms (e.g. [22]) have been proposed to efficiently transform the non-uniform 3D sensing data into triangle mesh, which comprises a large set of triangles with common edges and vertices T_i^l. Collision occurs when triangle is in contact with the robot segments, which implies that the shortest distance between the surround constrains and robot segment is zero ($d = 0$).

The PQ between an enclosing segment and a triangle can be regarded as a 4-dimensional constrained optimization problem. To date, the optimal value can be achieved via an iterative numerical technique, which is at a greater computational expense. In this study, a new algorithm is presented to eliminate this barrier by regarding each edge of the triangle mesh as an independent entity. Thus, the estimation of edge-segment shortest distance will be conducted firstly; thereby the closest point of each triangle will be searched by evaluating its three edges. This method is computationally efficient compared to the traditional approach, which processes triangles independently. It greatly reduces the computational workload in the way of averting the repetitive computation of edges shared by adjacent triangles. Besides, parallel computation with multiple independent threads on GPU can achieve significant acceleration. All these improvements implemented have led to the performance evaluation of novel PQ algorithm and the exploration of its application.

III. Formulation of Proximity Queries for Irregular Mesh Model

In order to realize the objectives stated in Section II, detailed PQ algorithm will be presented in this section. It more specifically, consists of the three steps: 1) analytical formulation of the edge-segment shortest distance; 2) approximation of its shortest distance; 3) identification of triangle-segment shortest distance.

A. Analytical Formulation of Edge-Segment Closest Point Pairs

To calculate the shortest distance between a triangle and segment, the primary step is to calculate the shortest distances between each edge of triangle and segment[23]. As an example, $\overrightarrow{AB}$ is the edge of triangle ΔABC (**Fig.2**), point x and point y are the parameterized points representing edge $\overrightarrow{AB}$ and surface of segment Ω_j respectively, which are defined as follows:

$$\begin{cases} x = A + \alpha \cdot \overrightarrow{AB} \\ y = (1-\eta)\cdot C_j(\theta) + \eta \cdot C_{j+1}(\theta) \end{cases} \quad (1)$$

where parameters $\alpha, \eta \in [0,1]$, $\theta \in [0, 2\pi)$.

With these parameterizations, the problem is transformed into the searching for a closest point pair $\overrightarrow{x_c y_c}$ between edge $\overrightarrow{AB}$ and segment Ω_j, as to which the corresponding shortest distance can be expressed as $d_c = \min\{|\overrightarrow{xy}(\alpha,\eta,\theta)|\}$. The closest point pair $\overrightarrow{x_c y_c}$ should meet the following three criteria in sense of geometrical analysis:

i. **Point y_c should lie on the cross section containing P_j, P_{j+1}, x_c;**

ii. **The closest point pair $\overrightarrow{x_c y_c}$ should be perpendicular to the edge $\overrightarrow{AB}$, which is $\overrightarrow{x_c y_c} \perp \overrightarrow{AB}$;**

iii. **The closest point pair $\overrightarrow{x_c y_c}$ should also be perpendicular to segment edge $\overrightarrow{v_2 v_3}$, which is $\overrightarrow{x_c y_c} \perp \overrightarrow{v_2 v_3}$.**

Referred to the above criteria, the closest point set $\overrightarrow{x_c y_c}(\alpha,\eta,\theta)$ can be determined with solution set (α,η,θ) of the formula below:

$$\begin{cases} \overrightarrow{x_c y_c} \times n_j = 0 \\ \overrightarrow{x_c y_c} \cdot \overrightarrow{AB} = 0 \\ \overrightarrow{x_c y_c} \cdot \overrightarrow{v_2 v_3} = 0 \end{cases} \quad (2)$$

where n_j is the normal of a plane containing points P_j, P_{j+1}, x_c, such that:

$$n_j = (x_c - P_j) \times (x_c - P_{j+1}) \quad (3)$$

By substitution of parameter vectors $u(\eta)$, $w(\theta)$ based on (1), (2) can be rearranged as follows:

$$\begin{cases} u(\eta) w(\theta)^T F_1 w(\theta) = \alpha \\ u(\eta) w(\theta)^T F_2 w(\theta) = 0 \\ u(\eta) w(\theta)^T F_3 w(\theta) = 0 \end{cases} \quad (4)$$

where coefficient matrix $F_{1,2,3} \in \Re^{3\times 3}$, and parameter vectors $u(\eta) = [1 \quad \eta]^T$, $w(\theta) = [1 \quad \sin\theta \quad \cos\theta]^T$

As a simplified version for this problem of distance between edge $\overrightarrow{AB}$ and contour segment Ω_j, the analytical space distance between an edge and a circle is already difficult to obtain [23]. In our problem, the analytical formula (i.e. (2)) is derived from a direct mathematical method and it can be regarded as a necessary reflection of the high geometrical complexity. In addition, this analytical formulation requires extensive computational burdens, making real-time calculation impossible. Instead, a novel algorithm costing light computational resources, but also with sufficient accuracy, might be the promising approach in searching for the closest point pair.

B. Optimization-based Edge-Segment Closest Point Pairs Estimation

Theoretically, an accurate distance d_1 can be obtained by solution set (α,η,θ) in (2), but along with the expensive computation. This is a trade-off that acquiring approximated closest point pair is relatively practical, rather than aiming to solving a closed form solution of (α,η,θ). By regarding the line as a cylinder of zero radius, then this problem is transformed into the case of calculation of the distance between two cylinders. Referring to the algorithm of cylinders intersection test proposed by Eberly *et al.* [23], it can be deduced that the minimization cost function that searches the shortest distance between segment and point along the line is strictly convex. In addition, the object function will have a global minimum that is either at the endpoints of line or the point in-between line where its partial derivatives for each variable are zero or undefined. In the collision detecting system, it works almost exclusively with convex objects (e.g. cylinder) due to the fact that convex objects can force the algorithm converge faster [24]. The properties of convexity and convergence guarantee the direction of the iterative method, which will be presented below, moving towards the closest pair as long as each step achieves point pair with shorter distance than the previous step.

It is worth noting that the last two criteria can be used to evaluate the accuracy of this solution. Referred to **Fig.3**, the proposed optimization-based method can be described as follows:

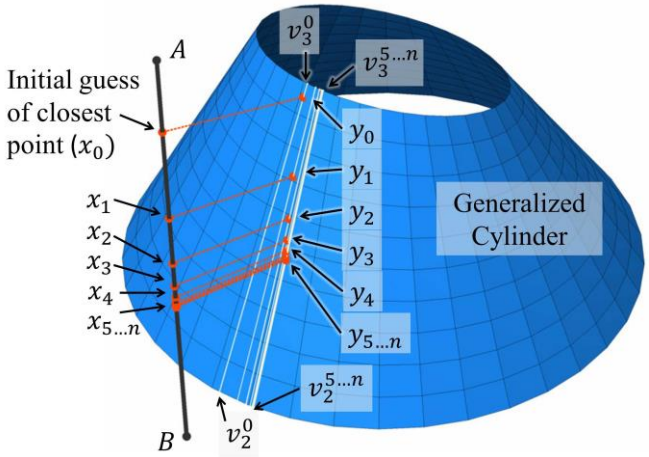

Fig. 3. An example configuration showing the temporary closest point pairs of each step processed by the optimization-based estimation. This configuration can be considered as an more extreme case requiring for more number of steps to reach convergence.

***Step 1*:** Find an initial point x_0 on edge $\overrightarrow{AB}$, which is the closest point to centerline $\overrightarrow{P_j P_{j+1}}$. This step provides an initial value of closest point, which only takes the distance between

centerline $\overrightarrow{P_jP_{j+1}}$ and edge $\overrightarrow{AB}$ into account. However, due to the fact that $\overrightarrow{AB}$ is not necessarily perpendicular to the centerline $\overrightarrow{P_jP_{j+1}}$, further steps will be provided.

***Step 2*:** Referring to the methods proposed in [18], corresponding vertices v_2^0 and v_3^0 on the circle contours can be found, which lie on the plane containing points P_j, P_{j+1}, x_0. The segment edge $v_2^0v_3^0$ contains the initial point y_0 on the enclosing segment and the initial distance is defined as $d_0 = \left|\overrightarrow{x_0y_0}\right|$.

***Step 3*:** Upon setting the initial guess, the iteration begins with finding the point x_{k+1} on edge $\overrightarrow{AB}$ that is closest to segment edge $v_2^kv_3^k$, where k is the index of iteration and $k = 0,1,2...$ (**Fig.2a**). This step, in essence, applies the third criterion to the current value x_k and comes up with a closer point pair estimate $x_{k+1}y'_k$.

***Step 4*:** Compute the corresponding closest point y_{k+1} and edge $v_2^{k+1}v_3^{k+1}$ on the enclosing segment by applying Kwok's PQ on x_{k+1} and the distance of this step is $d_{k+1} = \left|\overrightarrow{x_{k+1}y_{k+1}}\right|$. If x_{k+1} is found to locate inside the segment, return the shortest distance of the segment as zero ($d_c = 0$). The first and third criterions are applied again in this step to give a new estimation.

***Step 5*:** Test whether the resultant point pair $\overrightarrow{x_{k+1}y_{k+1}}$ fulfills the criteria 2 and 3 for the closest point pair. In this step, two vectors are considered as perpendicular if the dot product is smaller than a threshold ε, where ε is a small positive number.

$$\begin{cases} \left|\overrightarrow{x_{k+1}y_{k+1}} \cdot \overrightarrow{AB}\right| < \varepsilon \\ \left|\overrightarrow{x_{k+1}y_{k+1}} \cdot \overrightarrow{v_2^{k+1}v_3^{k+1}}\right| < \varepsilon \end{cases} \tag{5}$$

If Eq. (5) is not satisfied, increase k by 1 and repeat Step 3 to 5 until a closest point pair is found. For GPU implementation, repeating these steps for a fixed iteration is more efficient than imposing an ending criterion. If Eq. (5) is satisfied, the shortest distance d_c is found by $d_c = d_{k+1}$.

Between the two edges $\overrightarrow{AB}$ and $\overrightarrow{v_2^kv_3^k}$, point pair $\overrightarrow{x_{k+1}y'_k}$ is defined as the distance vector such that,

$$\left|\overrightarrow{x_{k+1}y'_k}\right| \le \left|\overrightarrow{x_ky_k}\right|, \text{ i.e. } d'_k \le d_k \tag{6}$$

And considering the two points y'_k, y_{k+1} on segment surface, $\overrightarrow{x_{k+1}y_{k+1}}$ is defined as the distance point pair between point x_{k+1} and segment Ω_j. Thus,

$$\left|\overrightarrow{x_{k+1}y_{k+1}}\right| \le \left|\overrightarrow{x_{k+1}y'_k}\right|, \text{ i.e. } d_{k+1} \le d'_k \tag{7}$$

The inequalities (6) & (7) ensure that a vector with shorter distance will be found in each loop, i.e. $d_{k+1} \le d_k$, i.e. straightly decreasing, in addition to the convexity of problem as mentioned above, the shortest distance will converge to global minimum after a certain number of iterations n.

In summary, the estimation of the shortest distance between edge $\overrightarrow{AB}$ and segment Ω_j is,

$$d_c = \min\{d_k, k = 0,1,2,\cdots,n\} \tag{8}$$

C. Identification of Shortest Distance from Triangle to Segment

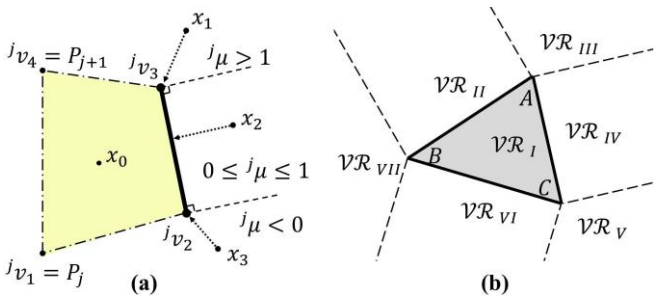

Fig. 4. (a) Portion of cross-sectional region extracted from a single segment. It passes through the corresponding contour centers P_j and P_{j+1}. The potential closest points $x_{0...3}$ always lay inside the Voronoi Region of the coresponding closest feature F_Y; (b) Voronoi Regions of a triangle in 2D space.

Upon computing the closest point pairs for each triangle edge, identification of the closest feature (vertex, edge, or face) on the triangle is of paramount importance. The traditional method (e.g. V-Clip [13]) operates iteratively by searching the closest points on the whole polyhedra, which is confined between convex shapes. The major disadvantage is its slow calculation speed for larger mesh and at the same time doesn't allow parallelization. Our identification method is realized based on Theorem 1, while the parallelization is allowed so as to increase its process speed.

Theorem 1 Assume that F_X and F_Y is a pair of features from each of two disjoint convex polyhedra, containing a pair of closest points for the polyhedra. Let $VR(F_X)$ and $VR(F_Y)$ denote their Voronoi Regions, $x \in F_X$ and $y \in F_Y$ be the closest points between F_X and F_Y. If $x \in VR(F_Y)$ and $y \in VR(F_X)$, then x and y are a globally closest pair of points between the polyhedra.

In the proposed PQ algorithm, F_X and F_Y represent the features (vertex, edge or surface) of triangle ΔABC and segment Ω_j respectively. As shown in **Fig.4a**, upon extracting a cross-sectional region confined by ${}^jv_{i=1,2,3,4}$, the closest feature F_Y (i.e. segment edge ${}^jv_2{}^jv_3$) is determined by ${}^j\mu$ such that the Voronoi Region $VR(F_Y)$ always contains x. Thus $x \in VR(F_Y)$ is always true by definition. Therefore, only the correlation between $VR(F_X)$ and F_Y need to be evaluated in order to find out the closest feature F_X.

A triangle possesses 7 features with mutually exclusive Voronoi Regions: 3 vertices, 3 edges and 1 surface. As shown in **Fig.4b**, Voronoi Regions VR_{III}, VR_V, VR_{VII} are for the vertex features A, B, C respectively, VR_{II}, VR_{IV}, VR_{VI} are for the edge features $\overrightarrow{AB}$, $\overrightarrow{AC}$, $\overrightarrow{BC}$ respectively and VR_I is for the face feature.

To identify the closest feature, the boundary features of a triangle comprising 3 edges and 3 vertices except for the face, are considered. The corresponding shortest distance will then be computed, thus finding the closest one to the contour segment. The shortest distance of the 3 bounding edges ($\overrightarrow{AB}$, $\overrightarrow{AC}$, $\overrightarrow{BC}$) is calculated using the proposed PQ formulation in Section B. This process naturally covers the PQ of vertex features as they are the endpoints of edges. Hence, the closest feature $F_{X\min}$ could be either laid on an edge or exactly at a

vertex, depending whether the resultant closest point is located at a vertex or on an edge on the triangle boundary. The closest feature will then be determined as the minimum one among the resultant shortest distances:

$$d_{\min} = \min\{d_{c,AB}, d_{c,BC}, d_{c,AC}\}$$

At this stage, the only remaining potential closest features on the triangle are $F_{X\min}$ and the face. Referring to Theorem 1, if the corresponding closest point on the segment side y_c lies within $VR(F_{X\min})$, we can confirm that the $F_{X\min}$ is the closest feature to the segment. Otherwise, the closest point appears on the surface of the triangle. This situation is relatively *rare* because it only happens when the closest point on the segment side lies on its circular contours. In this case, the corresponding closest point can be estimated intuitively with the centroid of the closest points of the three edges.

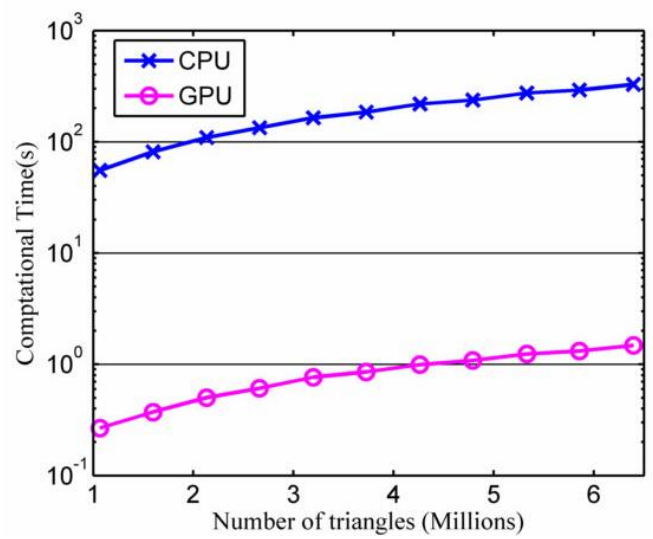

Fig. 5. Illustrating GPU and CPU computational performance of PQ process with 30 segments and millions of meshes

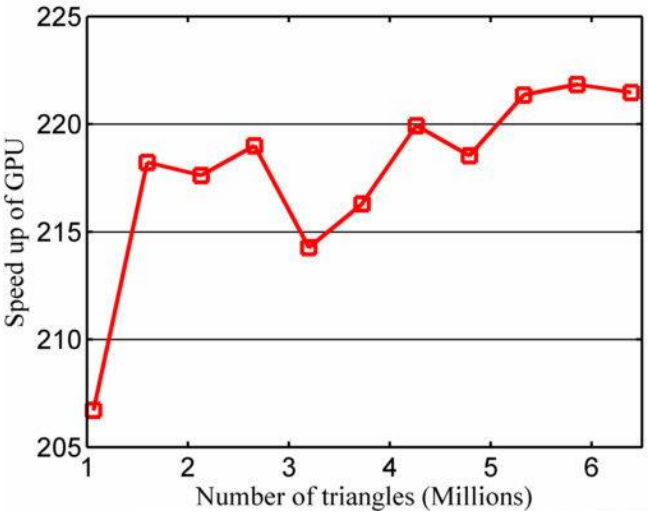

Fig. 6. Speedup of PQ computation implemented on GPU with respect to CPU

IV. Implementation and Results

The overall performance of the proposed PQ algorithm is evaluated on two different types of processor: multi-core CPUs and GPUs, of which their speedup and real-time response are investigated. The CPU-based PQs act as the baseline or reference for comparison with the implementation on GPUs. The CPU-based reference is also designed with detailed kernel source code (CUDA), and compiled using Visual Studio 2012 on an AMD Phenom™ II X4 955 Processor@3.20GHz. A standard GPU platform, nVidia GTX 770, is adopted, which contains 1536 CUDA cores.

Modern GPUs normally have 10-100 times higher computation power in floating-point precision, compared to the modern CPUs. In **Fig.6**, our GPU speedup can achieve two orders, around 200 times, faster than the single-core CPU; Thirty segments are involved, and it allows for PQs on around 4M triangle meshes at rate of >1Hz (**Fig.5**). It is worth noting that their main memory bandwidth does not scale up accordingly and is usually 5-10 times higher than CPUs only. This is also the reason why the number of times speedup cannot be maintained necessarily at constant level, as illustrated in **Fig.6**, but showing that the speedup can be even increased with the number of triangular mesh because of its further efficient memory access.

In our GPU-based computational scheme, we propose to transfer mesh data into small local memory within each CUDA processor; therefore, only the triangle data are loaded from the main memory. This prevents from degrading the performance due to the insufficient bandwidth of the main memory access.

A. Broad-phase culling with Mesh Segmentation and Level of Detail Techniques

A broad phase technique that efficiently prune away unnecessary pair-wised PQ test in the search of the shortest distance is employed. This further enhance the overall performance of the proposed PQ algorithm:

i. A set of sub-meshes is created by bisecting the input mesh successively into smaller pieces, as shown in **Fig.7**.

ii. Each sub-mesh is simplified into a lower level-of-detail (LOD) version while preserving their geometric feature.

iii. To cull out the closest sub-mesh to the contour segments, the PQ of every simplified sub-meshes are computed, which can be completed quickly because the number of mesh is substantially reduced after simplification.

iv. Finally, the exact closest point pair is determined by performing our proposed PQ on only the closest piece of mesh in high resolution.

The advantage of this technique is that it does neither require a convex input mesh nor convex bounding volume representation of the input mesh, resulting in high computationally efficiency and easy implementation. The experimental result (**Table 1.**) demonstrated that this culling technique further speedup the overall PQ performance on both CPU and GPU processors by at least 70%.

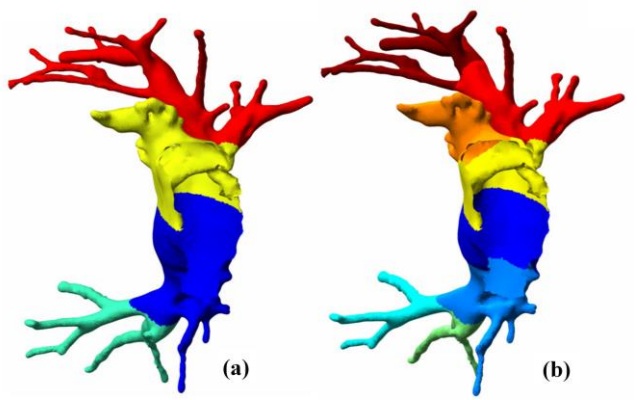

Fig. 7. The segmented sub-mesh of the left atrium model with (a) 8 segments (b) 4 segments for broad-phase culling

Processor	Layer(s) of Bisection	Culling Time (s)	Narrow-phase Time (s)	Total Time (s)
CPU	Brute	-	46.04	46.04
	1	9.15	1.73	10.89
	2	9.22	0.70	9.9
	3	9.22	0.33	9.5
GPU	Brute	-	0.946	0.946
	1	0.193	0.035	0.227
	2	0.206	0.014	0.220
	3	0.213	0.006	0.219

Table 1. The computation time of PQs on 1 million of mesh. PQs with culling are also applied with different numbers of model bisections.

Two scenarios are simulated and designed according to practical robotic tasks in two extremely different nature of size and manipulation accuracy required. Both simulated tasks provide sufficient information to analyze the efficiency and accuracy of the proposed PQ under the software framework of Robot Operating System (ROS).

B. Intra-cardiovascular Catheter Navigation

Fig.8 shows the human left atrial model constructed by pre-operative cardiac MR images for cardiovascular electrophysiology (EP) intervention – a minimally-invasive surgical treatment of heart rhythm disorders [25]. This mesh surface model comprises 5,372 vertices, 31,974 edges and 10658 triangular meshes, which is a part of the EP roadmap acting as the primary reference for electrophysiologist to maneuver the catheter tip to the lesion target for tissue ablation [26]. The EP catheter tip is fitted with 7 contour segments, and its position is measured and tracked continuously [27]. Its instantaneous PQ distances relative to the roadmap surface are indicated with the RGB colour codes. Such PQ information can be foreseen to give strong hints for safe and precise catheter navigation, since the robotic control of catheter for EP has already adopted in real clinical practice (e.g. Hansen Sensei® robotic catheter system). Given the EP roadmap modelled by the number of mesh below 10K, the PQ can be mainland at rate >1Hz; thereby, image-based haptic feedback can be made possible, and it could provide electrophysiologist with navigation guidance to access route to lesion regions safely and accurately [28].

C. Motion Planning for Anthropomorphic Manipulator

The other scenarios [29] (**Fig.9**) simulates an interactive robotic task involving high degree-of-freedom (DoF) manipulation carried out by an advanced humanoid robot, ATLAS (Boston Dynamics, MA, USA). The active sensory system, Light Detection and Ranging (LIDAR), mounted on its head, enables accurate geometry perception. It also facilitates high-quality 3D representation of the unstructured environment in the form of cloud points. The mesh surface (**Fig.9**) is constructed online with fast tessellation of the point-cloud data. This is a sub task in the DARPA Challenge, which is required to accomplish autonomously with minimal human intervention. The anthropomorphic arm and hand are controlled to reach the valve wheel behind a thin wall. The warped surface throughout the hole is also detailed by the 4308 meshes. High-frequency PQ process is applied to deduce a collision-free trajectory in high-dimension joint space so that the hand posture can be optimized so as to turn the valve wheel behind the wall ergonomically.

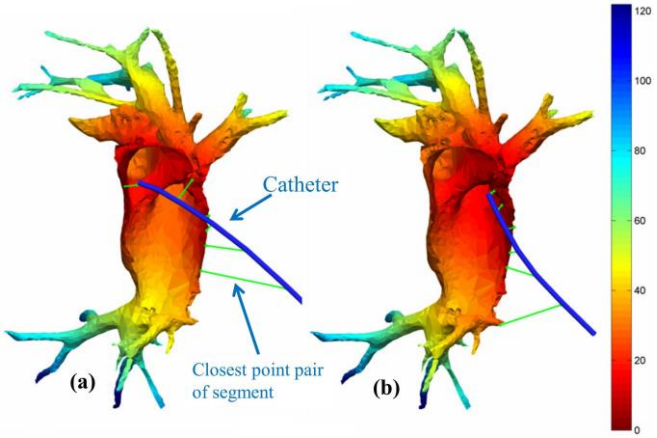

Fig. 8. (a-b) Two catheter configurations with different degree of steering curvature. The catheter (with 7 segmentes) is advanced inside the left atrium model (with 10,658 mesh) for radiofrequency ablation during the cardiovascular electrophysiology procedure. The PQ sampling rate is maintained at rate of 1kHz.

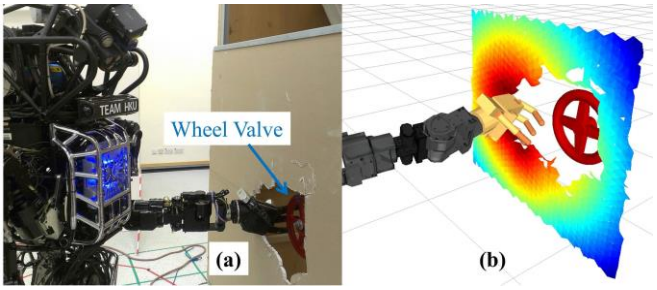

Fig. 9. (a) ATLAS reaching the valve wheel through a hole created previously by the hand-held cutting tool. (b) PQ processed between the robot hand (with 39 segments) and the wall with hole (with 4,308 meshes). An optimal collision-free trajectory is estimated online with the aim to grasp the wheel. The warmer the colour on wall indicates, the closer distance to the robot arm and fingers is.

V. Discussion and Conclusion

We have summarized the challenges and difficulties of implementing conventional PQ approaches. It is well recognized that the performance of many real-time

applications, such as haptic rendering and robot motion planning, is coupled with an increasing demand on fast and effective PQs. A versatile PQ formulation is proposed that the unstructured environment involved in robotic tasks can be flexibly represented by irregular triangular meshes, without having to preprocess the input geometric data. Moreover, the fitting of the contours along the robot kinematic chain needs only to be conducted once. No specific constraint is required for re-modeling the meshes, thus facilitating its use for generic geometric structures. The input geometric data is also not necessary to be temporal coherent. The number and configuration of meshes and contour segments can be varying time-by-time; therefore, the unstructured environment can be kept rapidly changing in terms of its morphology and topology.

Although the BIG-O complexity of this PQ algorithm is proportional to the number of meshes and contour segments, its computation structure is highly parallel and anticipated to be further advanced with the trendy development of GPUs or FPGAs. The GPU-based PQ computation has been demonstrated to achieve >200 times speedup over single-core CPU. Two scenarios have also simulated for validating its practical values for real-time applications.

In our future, this work will be packaged as an open-source library. It can be incorporated under ROS in support of many real-time applications. We will also investigate how this PQ algorithm can take further advantage of exploiting mixed precision technologies [8] with larger single-precision FPU resources on FPGAs.

ACKNOWLEDGMENT

This work is supported in parts by EPSRC in the UK, the Croucher Foundation and the Research Grants Council (RGC) in Hong Kong. We sincerely thank to the team in Advanced Robotics Laboratory at The University of Hong Kong for the access to their equipment.

REFERENCES

[1] N. Chakraborty, J. Peng, S. Akella, and J. Mitchell, "Proximity Queries between Convex Objects: An Interior Point Approach for Implicit Surfaces," in *IEEE International Conference on Robotics and Automation*, 2006, pp. 1910-1916.

[2] J. Pan, L. Zhang, and D. Manocha, "Collision-free and Smooth Trajectory Computation in Cluttered Environments," *International Journal of Robotics Research,* vol. 31, pp. 1155-1175, 2012.

[3] M. Li and R. H. Taylor, "Spatial motion constraints in medical robot using virtual fixtures generated by anatomy," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 1270-1275.

[4] M. Li, M. Ishii, and R. H. Taylor, "Spatial motion constraints using virtual fixtures generated by anatomy," *IEEE Trans. on Robotics,* vol. 23, pp. 4-19, 2007.

[5] B. Davies, M. Jakopec, S. J. Harris, F. R. Y. Baena, *et al.*, "Active-constraint robotics for surgery," *Proceedings of the IEEE,* vol. 94, pp. 1696-1704, 2006.

[6] K. W. Kwok, V. Vitiello, and G. Z. Yang, "Control of Articulated Snake Robot under Dynamic Active Constraints," *Medical Image Computing and Computer-Assisted Intervention - Miccai 2010, Pt Iii,* vol. 6363, pp. 229-236, 2010.

[7] X. Zhang and Y. J. Kim, "Interactive Collision Detection for Deformable Models Using Streaming AABBs," *IEEE Trans. on Visualization and Computer Graphics,* vol. 13, pp. 318-329 2007.

[8] G. C. T. Chow, K. W. Kwok, W. Luk, and P. Leong, "Mixed Precision Comparison in Reconfigurable Systems," in *IEEE International Symposium on Field-Programmable Custom Computing Machines*, 2011, pp. 17-24.

[9] A. Escande, S. Miossec, and A. Kheddar, "Continuous gradient proximity distance for humanoids free-collision optimized-postures," in *IEEE-RAS International Conference on Humanoid Robots*, 2007, pp. 188-195.

[10] C. L. Bajaj and T. K. Dey, "Convex Decomposition of Polyhedra and Robustness," *SIAM Journal on Computing,* vol. 21, pp. 339-364, 1992.

[11] M. C. Lin and J. F. Canny, "A fast algorithm for incremental distance calculation," in *IEEE International Conference on Robotics and Automation*, 1991, pp. 1008-1014.

[12] J. F. Canny and M. C. Lin, "An opportunistic global path planner " in *IEEE International Conference on Robotics and Automation*, 1990, pp. 1554-1559.

[13] B. Mirtich, "V-Clip: Fast and robust polyhedral collision detection," *ACM Trans. on Graphics,* vol. 17, pp. 177-208, 1998.

[14] E. Gilert, D. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three dimensional space," *IEEE Journal of Robotics and Automation,* vol. 4, pp. 193-203, 1988.

[15] S. Cameron, "A Comparison of Two Fast Algorithms for Computing the Distance between Convex Polyhedra," *IEEE Trans. on Robotics,* vol. 13, pp. 915-920, 1997.

[16] N. Chakraborty, J. Peng, S. Akella, and J. E. Mitchell, "Proximity Queries Between Convex Objects: An Interior Point Approach for Implicit Surfaces," *IEEE Trans. on Robotics,* vol. 24, pp. 211-220, 2008.

[17] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An Interior Point Algorithm for Large Scale Nonlinear Programming," *SIAM Journal on Optimization,* vol. 9, pp. 35-59, 1999.

[18] K. W. Kwok, K. H. Tsoi, V. Vitiello, J. Clark, *et al.*, "Dimensionality Reduction in Controlling Articulated Snake Robot for Endoscopy Under Dynamic Active Constraints," *IEEE Transactions on Robotics,* vol. 29, pp. 15-31, Feb 2013.

[19] K. W. Kwok, G. C. T. Chow, T. C. P. Chau, Y. Chen, *et al.*, "FPGA-based acceleration of MRI registration: an enabling technique for improving MRI-guided cardiac therapy," *Journal of Cardiovascular Magnetic Resonance,* vol. 16(Suppl 1):W11, 2014.

[20] T. C. P. Chau, K. W. Kwok, G. C. T. Chow, K. H. Tsoi, *et al.*, "Acceleration of real-time Proximity Query for dynamic active constraints," in *Field-Programmable Technology (FPT), 2013 International Conference on*, 2013, pp. 206-213.

[21] K. W. Kwok, G. P. Mylonas, L. W. Sun, M. Lerotic, *et al.*, "Dynamic Active Constraints for Hyper-Redundant Flexible Robots," *Medical Image Computing and Computer-Assisted Intervention - Miccai 2009, Pt I, Proceedings,* vol. 5761, pp. 410-417, 2009.

[22] S. Ilic and P. Fua, "Implicit meshes for surface reconstruction," *IEEE Trans on Pattern Analysis and Machine Intelligence,* vol. 28, pp. 328-333, Feb 2006.

[23] D. Eberly, "Intersection of cylinders," 2010.

[24] C. Fares and Y. Hamam, "Collision detection for rigid bodies: A state of the art review," *GraphiCon 2005,* 2005.

[25] Y. Chen, K. W. Kwok, J. Ge, Y. Hu, *et al.*, "Augmented Reality for Improving Catheterization in MRI-guided Cardiac Electrophysiology," *Journal of Medical Devices - Transactions of the ASME,* vol. 8, p. 020917, 2014.

[26] Y. Chen, J. Ge, K. W. Kwok, K. R. Nilsson, *et al.*, "MRI-conditional catheter sensor for contact force and temperature monitoring during cardiac electrophysiological procedures," *Journal of Cardiovascular Magnetic Resonance,* vol. 16(Suppl 1):P150.

[27] K. W. Kwok, K. H. Lee, Y. Chen, W. Wang, *et al.*, "Interfacing Fast Multi-phase Cardiac Image Registration with MRI-based Catheter Tracking for MRI-guided Electrophysiological Ablative Procedures," *Circulation,* vol. 130, A18568, 2014.

[28] K. W. Kwok, Y. Chen, T. C. P. Chau, W. Luk, *et al.*, "MRI-based visual and haptic catheter feedback: simulating a novel system's contribution to efficient and safe MRI-guided cardiac electrophysiology procedures," *Journal of Cardiovascular Magnetic Resonance,* vol. 16, p. O50, 2014.

[29] K. H. Lee and W. S. Newman, "Natural Admittance Control of an Electro-Hydraulic Humanoid Robot," in *IEEE International Conference on Robotics and Biomimetics*, 2014 (accepted)