PhongPD: Gradient-continuous Penetration Metric for Polygonal Models using Phong Projection

(http://graphics.ewha.ac.kr/PhongPD)

Youngeun Lee and Young J. Kim

***Abstract*— We present a novel algorithm to compute a gradient-continuous penetration depth (PhongPD) between two interpenetrated polygonal models. Our penetration depth (PD) formulation ensures separating the intersected models by translation, and the amount of such translation is close to an optimal motion to resolve interpenetration in most cases. In order to achieve the gradient-continuity in our algorithm, we interpolate tangent planes continuously over the contact space and then perform a projection along a normal direction defined by the interpolated tangent planes; this projection scheme is known as Phong projection. We have implemented our PhongPD algorithm and certifies its continuity using three benchmarks consisting of diverse combinatorial complexities, and show that our algorithm shows smoother PD results than a conventional Euclidean-projection-based PD method.**

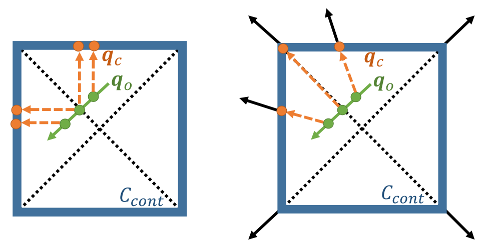
I. Introduction

Measuring the amount of interpenetration between overlapping models is an important problem in robotics. A well-known distance measure for inter-penetration is penetration depth (PD), which is defined as a minimum translation to separate two intersecting models [1], [2]. Also, it is known that the PD is equivalent to finding the minimum distance from the origin of a translational configuration space obstacle (equivalently, Minkowski sums) to its boundary. This PD definition is widely used in robotic and engineering application. For instance, in optimization-based robot motion planning, PD is used to formulate non-penetrating constraints [3], [4]. Retraction-based robot motion planning heavily relies on PD to identify collision-free configurations in the presence of narrow passages [5]. Furthermore, PD can be used to determine path existence for robot motion planning [6].

However, the original definition of PD can easily yield a discontinuity in the direction (or gradient) of PD, which often causes an instability problem in the aforementioned robotic applications. Fig. 1(a) illustrates a PD discontinuity example. In the figure, the PD result ($\mathbf{q}_c$) corresponds to an Euclidean projection, displayed as orange arrows, from the origin ($\mathbf{q}_o$) to the contact space (C_{cont}), and as $\mathbf{q}_o$ moves along the green arrow, $\mathbf{q}_c$ jumps from one configuration to another configuration even when q_o slightly changes. This discontinuity can lead to unstable force response in penalty-based dynamics.

Note that known practical PD algorithms such as optimization-based PD methods [7], [8] tend to compute

Y. Lee and Y. J. Kim are with the Department of Computer Science and Engineering at Ewha Womans University in Seoul, Korea. youngeunlee@ewhain.net, kimy@ewha.ac.kr



(a) Euclidean projection (b) Phong projection

Fig. 1. The blue rectangle is C_{cont} and the black doted lines represent the medial axis of C_{obj}. The green circles are $\mathbf{q}_o$ and $\mathbf{q}_o$ moves along the green arrow. The orange circles show projection results. (a) $\mathbf{q}_c$ are computed by Euclidean projection. Note that the direction of PD is not continuous when $\mathbf{q}_o$ crosses the media axis of C_{cont}. (b) $\mathbf{q}_o$ is projected by Phong projection. The projection results are continuous even if $\mathbf{q}_o$ is on the medial axis because the projection direction (the orange arrows) also continuously changes along the corresponding surface normals (or tangents).

discontinuous PD. However, finding a gradient-continuous PD is a difficult problem to solve since the original PD definition relies on Euclidean projection and it naturally incurs a discontinuity as demonstrated in Fig. 1(a). To the best of our knowledge, [9] is the only known work to handle this problem. However, this technique can deal with a configuration space only when it is homeomorphic to a sphere with no holes, which is not satisfied for general robotic applications and also is difficult to determine a priori.

Main Results We present a novel algorithm, called PhongPD, to compute a gradient-continuous penetration depth between two intersected models. Just like the conventional PD definition, the PhongPD can be computed as a minimum distance from the relative configuration of two models to their contact space, but the minimum distance is determined using Phong projection [10] instead of Euclidean projection. This guarantees continuous projection results. In order to perform Phong projection, we define tangent planes continuously over the boundary of contact space, similarly to [11], and then project the relative configuration along the normal direction defined by the correspond tangent plane. To guarantee the gradient-continuity for a moving model, we cache the PhongPD result from the previous motion frame, and use it to find the proper projection direction. As a result, our algorithm performs Phong projection quickly and continuously. To demonstrate the efficiency of our PhongPD algorithm, we tested our algorithm with three benchmarks generating contact spaces whose complexities range from 440 to 4K triangles. In these cases, the results of our method are continuous and smoother than the conventional PD.

II. Previous Work

We briefly survey various PD algorithms using different distance metrics and formulations.

A. Translational Penetration Depth

The translational PD (or simply PD) can be computed using the Minkowski sums and is defined as the shortest distance from the origin to the boundary of Minkoswksi sums [1]. It is well known that the complexity of exact PD computation for convex and non-convex rigid models are $O(n^2)$ and $O(n^6)$, respectively, where n is the number of polygons in the models. For convex models, [2] and [12] suggested exact computational algorithms but no implementations exist. [13] and [14] approximated PD tightly and were implemented successfully showing a good runtime behavior.

For non-convex models, [15] presented an exact PD computation method using convex decomposition but its implementation does not exist. Approximate PD algorithms such as [16], [17] are based on Minkowski sums, but are rather slow in terms of runtime performance. [8] suggested a real-time algorithm based on iterative local optimization on a local contact space.

B. Generalized Penetration Depth

The generalized PD is defined as a minimal rigid motion to separate overlapping objects [7]. Here, the "minimality" is defined under some distance metric in configuration space such as DISP [18], object norm [7], S or geodesic [19]. The computational complexity of exact generalized PD computation can be as high as $O(n^{12})$, where n is the number of polygons in the models. These method are rather slow for real-time applications. More recently, [20] presented an interactive-rate generalized PD computation method for rigid and articulated models. Machine learning techniques were also used to compute PD [21], but they suffer from an accuracy problem when the penetration depth is small.

C. Other Penetration Measures

The translational and generalized PD's are not the only measures of interpenetration for overlapping models. The growth distance [22], [23] is an alternative measure to quantify the interpenetration between convex models. It is defined as a maximum scaling factor of two models in order to just touch each other. [24], [25] and [26] used an intersection volume and distance fields, respectively, to measure the model interpenetration. [27] suggested point-wise PD which is defined as the Hausdorff distance of the intersection volume between two overlapping models, and was applied to physically-based animation.

No algorithm exist to compute the gradient-continuous PD except [9], which they call continuous PD, but this method works only when the contact space is spherical with no holes. In our case, we do not have such a restriction.

III. Preliminaries

In this section, we introduce our notations and define the penetration depth (PD). And we describe our approach to compute gradient-continuous penetration depth using Phong projection.

A. Problem Formulation

The configuration space is a set of all possible configurations for a given robot. Suppose that we have two polygonal models $\mathcal{A}$ and $\mathcal{B}$ in $\mathbb{R}^3$. Then each point or configuration in configuration space corresponds to a relative configuration of $\mathcal{A}$ with respect to $\mathcal{B}$. Without loss of generality, we assume that $\mathcal{A}$ is a robot movable only by translation and $\mathcal{B}$ is a fixed obstacle. Then the associated configuration space has three degrees of freedom (DoFs) in our problem.

A configuration space can be further subdivided into two subsets: collision-free space $C_{free} = \{\mathbf{q} | \mathcal{A}(\mathbf{q}) \cup \mathcal{B} = \emptyset\}$ and obstacle space $C_{obj} = \{\mathbf{q} | \mathcal{A}(\mathbf{q}) \cup \mathcal{B} \neq \emptyset\}$ where $\mathcal{A}(\mathbf{q})$ corresponds to $\mathcal{A}$ located at the configuration $\mathbf{q}$. The contact space C_{cont} is the boundary of the obstacle space, i.e. $C_{cont} = \partial C_{obj}$, in which $\mathcal{A}$ and $\mathcal{B}$ just touch each other. When $\mathcal{A}$ and $\mathcal{B}$ are polygonal models, C_{cont} is the boundary of the Minkowski sums $\mathcal{A} \oplus -\mathcal{B}$.

The penetration depth is defined as the minimum translational motion to separate two overlapping models. In other words, it is a translational motion to move from the current configuration $\mathbf{q}_o$ to the closest configuration in C_{cont}:

$$\mathbf{q}_c = \underset{\mathbf{q} \in C_{cont}}{\operatorname{argmin}}\ \delta(\mathbf{q}_o, \mathbf{q}) \tag{1}$$

where δ is the Euclidean distance function. Finally we get PD as:

$$PD(\mathcal{A}(q_o), \mathcal{B}) = \mathbf{q}_c - \mathbf{q}_o \tag{2}$$

Penetration depth is a vector function and has both magnitude and direction. The magnitude of PD is continuous as long as C_{cont} is closed and continuous. However the direction of PD (i.e. its gradient) may not be continuous for polygonal models due to the Euclidean projection of $\mathbf{q}_o$ to C_{cont}. For instance, as illustrated in Fig. 1(a), when $\mathbf{q}_o$ crosses the medial axis [29] of C_{obj}, $\mathbf{q}_c$ can jump from one configuration to another abruptly.

B. Algorithm Overview

The goal of our work is projecting $\mathbf{q}_o$ onto C_{cont} to obtain a gradient-continuous penetration depth like Fig. 1(b). Our algorithm uses Phong projection [10] instead of conventional Euclidean projection, where the PD vector between $\mathbf{q}_o$ and $\mathbf{q}_c$ is collinear with the surface normal $\mathbf{n}_c$ at $\mathbf{q}_c$:

$$(\mathbf{q}_c - \mathbf{q}_o) \times \mathbf{n}_c = \mathbf{0}. \tag{3}$$

Euclidean projection can be considered as a special case of Phong projection because Eq. (3) is still satisfied in that case. The main difference between Euclidean and Phong projections lies in defining surface normals (equivalently, the projection direction). While all points on the same planar surface have a same surface normal (or projection direction) in Euclidean projection, the surface normals (or projection directions) may vary in Phong projection. In the latter case, the results of projection are continuous under some condition [11].

Our PhongPD algorithm has two phases: offline and online phases. The offline phase interpolates tangent planes over C_{cont}. The online query phase corresponds to Phong projection to yield a gradient-continuous penetration depth.

IV. Phong Projection

C_{cont} for two polygonal models in $\mathbb{R}^3$ with three DoFs is also polygonal [17]. In order to perform Phong projection onto a polygonal surface, the Phong projection for a polygonal model as well as continuous tangent planes need to be defined properly. In our case, we rely on a technique proposed by [11].

A. Projection

The original Phong projection [30] uses normals like in Eq. (3) and interpolate the normals to get continuous projection. In our case, however, we use tangents instead of normals because the normal interpolation can induce a singular point; the normal is still used eventually for projection, but derived from a tangent vector. For example, when the directions of two normal are collinear but have an opposite direction, then their interpolation is $\mathbf{0}$. However, a tangent plane interpolation simply rotates its basis vectors with no singular interpolation.

Assuming that a polygonal surface is triangulated without loss of generality, let a triangle $\triangle\mathbf{t} \subset C_{cont}$ consisting of three vertices $\{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3\}$ and each vertex has a tangent plane spanned by two basis vectors $T_i \in \mathbb{R}^{2\times 3}$. In order to define a surface normal for a vertex, we set the vertex normal as an average of the surface normals of adjacent triangles and compute a tangent plane orthogonal to the normal. Let us denote a point in $\triangle\mathbf{t}$ as $\mathbf{q} = \lambda_1\mathbf{q}_1 + \lambda_2\mathbf{q}_2 + \lambda_3\mathbf{q}_3$, $\lambda_i \geq 0$, $\sum\lambda_i = 1$ using the Barycentric coordinate $(\lambda_1, \lambda_2, \lambda_3)$. Let $\Psi(\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}^{2\times 3}$ be a continuous interpolation of tangent planes over the interior of $\triangle\mathbf{t}$. The Phong projection from a point onto a triangle is defined as follows:

Definition 1 *A point* $\mathbf{q}_c = (\lambda_1, \lambda_2, \lambda_3)$ *on a triangle* $\triangle\mathbf{t}(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$ *is a Phong projection of* $\mathbf{q}_o$ *on* $\mathbf{t}$ *if:*

$$\Psi(\lambda_1, \lambda_2, \lambda_3)(\lambda_1\mathbf{q}_1 + \lambda_2\mathbf{q}_2 + \lambda_3\mathbf{q}_3 - \mathbf{q}_o) = \mathbf{0} \quad (4)$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \quad (5)$$

$$\lambda_i \geq 0 \quad (6)$$

Eq. (4) means that the tangent plane of $\mathbf{q}_c$ is perpendicular to $\mathbf{q}_c - \mathbf{q}_o$, and it equivalent to Eq. (3).

The Phong projection onto a triangulated surface is equivalent to the closest Phong projection from $\mathbf{q}_o$ to all triangles comprising the surface. Note that Phong projection may not exist in some cases. For instance, Fig. 2 illustrates different results of Phong projections using different normal assignments. Thus, defining normals or tangents properly is crucial for valid Phong projection.

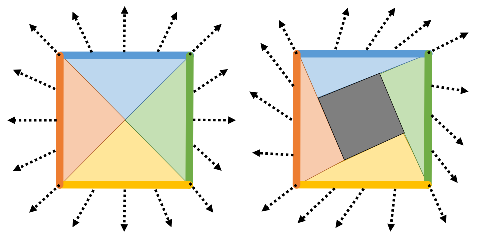


Fig. 2. Phong projection for rectangles using different normal assignments. The black doted lines denote normals, and the color-coded areas are a set of points that are Phong-projected to the edges with the same color. Phong projection is undefined for the gray area.

B. Continuous Tangent Computation

We first define the distance between two tangent planes, T and K, before defining Ψ as follows:

$$d(T, K) = \min_{A\in O(2)} \|\mathrm{Ort}(T) - A\mathrm{Ort}(K)\| \quad (7)$$

where $\mathrm{Ort}(\cdot)$ denotes the nearest orthonormal basis using Frobenius norm as in Eq. (8), which can be computed using polar decomposition.

$$\mathrm{Ort}(T) = \underset{B\in\mathbb{R}^{2\times 3}:BB^T=I}{\mathrm{argmin}} \|T - B\|_F \quad (8)$$

We use tangent planes defined as in Definition 2 [11]. The interpolation here is continuous under some condition and independent of choosing bases representing the tangent planes.

Definition 2 *Continuous tangent planes for a point* $(\lambda_1, \lambda_2, \lambda_3)$ *in* $\triangle(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$ *is computed as below:*

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \frac{\lambda_1\lambda_2\lambda_3}{\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1}\Big(\frac{1}{\lambda_3}\Psi_{12}(\lambda_1, \lambda_2, \lambda_3) + \frac{1}{\lambda_1}\Psi_{23}(\lambda_1, \lambda_2, \lambda_3) + \frac{1}{\lambda_2}\Psi_{31}(\lambda_1, \lambda_2, \lambda_3)\Big) \quad (9)$$

where

$$\Psi_{12}(\lambda_1, \lambda_2, \lambda_3) = \lambda_1 E_{12}R_{12}T_1 + \lambda_2 E_{12}T_2 + \lambda_3\frac{1}{2}(E_{23} + E_{31}R_{31})T_3 \quad (10)$$

Here, Ψ_{ij} is a interpolation function of the tangent plane for edge $\overline{\mathbf{q}_i\mathbf{q}_j}$. Ψ_{23} and Ψ_{31} are computed by cyclic permutation of indices. $R_{ij} = \mathrm{Ort}(T_jT_i^T)$ is an orthogonal matrix to minimize the difference between the bases T_i and T_j. $E_{ij} \in \mathbb{R}^{2\times 2}$ is computed as follows:

$$E_{12}, E_{23}, E_{31} = \underset{E_{12},E_{23},E_{31}\in O(2)}{\mathrm{argmin}} \sum_{1\leq i<j\leq 6} \|A_i - A_j\|_F^2 \quad (11)$$

where $A_1 = E_{12}R_{12}T_1$, $A_2 = E_{12}T_2$, $A_3 = E_{23}R_{23}T_2$, $A_4 = E_{23}T_3$, $A_5 = E_{31}R_{31}T_3$, and $A_6 = E_{31}T_1$. The E_{ij} is iteratively optimized by fixing two values in E_{ij} and solving the Procrustes problem for unfixed ones. For a well-tessellated model, the existence of Phong projection is guaranteed for a point close to the surface of a model [11].

V. Continuous Phong Projection on Contact Space

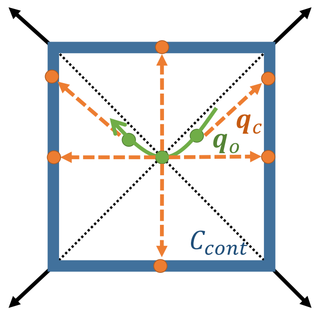
In the previous section, we explained how to interpolate tangent planes, and now we explain how to perform Phong projection to the contact space efficiently and robustly, which in turn corresponds to the PD result.

A. Phong Projection on a Triangle

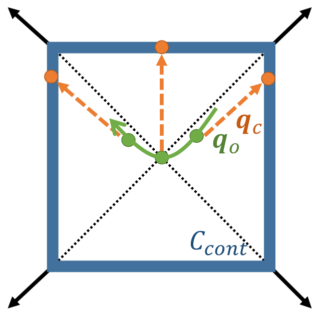
The Phong projection for a triangle is equivalent to finding λ_i in Definition 1. λ_i is computed using Newton's method starting from $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ until Eqs. (4) and (5) are satisfied. If no result is found to satisfy Eq. (6), we conclude that Phong projection does not exist for this triangle.

B. Phong Penetration Depth

A Phong projection on a triangulated surface is equivalent to the shortest Phong projection with respect to every triangle in the surface. As illustrated in Fig. 3(a), however, the result of Phong projection may not be unique. In other words, Def. 1 may have multiple solutions. In this case, we choose the one closest from the previous computation result, and this projection operator (i.e. PhongPD) shows reliable results.



(a) Phong Projection



(b) Phong PD

Fig. 3. The blue rectangle denotes C_{cont}. The black solid arrows show vertex normals. As $\mathbf{q}_o$ moves along the green curve, (a) has four Phong-projected results (the two vertical and two horizontal orange arrows) when $\mathbf{q}_o$ is located at the center of C_{cont}, but (b) has a unique PD (the upper arrow), which is closest to the previous result.

C. Efficient Projection

Repeating projection $\mathbf{q}_o$ for all triangles to find the best projection is inefficient. Panozzo *et al.* [11] project a point based on the result of Euclidean projection because the Euclidean projection is a special case of Phong projection and their results could be similar. However, in our case, to make this process run more efficiently and reliably, we start to project from the triangle which was obtained from the previous frame of motion like Fig. 4. This "warm-start" searching method reduces the search space. To effectively implement this, we construct K-ring neighbors for each triangle as a list like in Fig. 4 before the projection starts, and this search starts using this list. In case that, results from the previous frame are not available (e.g. the first frame), we simply return the shortest Phong projection from $\mathbf{q}_o$. Moreover, sometimes Phong projection does not exist as illustrated in Fig. 2. This can happen when $\mathbf{q}_o$ is located deeply inside C_{cont}.

Algorithm 1 PhongPD

Input: Input collision configuration $\mathbf{q}_o$, a triangle containing the result from the previous frame $\triangle\mathbf{t}_p$, contact space C_{cont}, k-ring neighbor $Neighbor$

Output: Result of Phong projection $\mathbf{q}_c$

```
1: if !$\triangle\mathbf{t}_p$ then
2:     {There is no previous result}
3:     $MinLength$ := Maximum value;
4:     for $i = 1$ to Number of Triangles in $C_{cont}$ do
5:         $\triangle\mathbf{t}$ :=$C_{cont}.(i)$;
6:         $(\lambda_1, \lambda_2, \lambda_3)$ := PhongProjection($\mathbf{q}_o$, $\triangle\mathbf{t}$);
7:         if $(\lambda_i \geq 0, i = 1,2,3)\&MinLength > \|\triangle\mathbf{t}.\text{barycentric}(\lambda_1, \lambda_2, \lambda_3) - \mathbf{q}_o\|$ then
8:             {Phong Projection exists by Eq.(6)}
9:             $MinLength$ := $\|\triangle\mathbf{t}.\text{barycentric}(\lambda_1, \lambda_2, \lambda_3) - \mathbf{q}_o\|$;
10:            $\mathbf{q}_c$ := $\triangle\mathbf{t}$.barycentric($\lambda_1, \lambda_2, \lambda_3$);
11:        end if
12:    end for
13:    if $MinLength \neq$ Maximum value then
14:        return $\mathbf{q}_c$;
15:    end if
16: else
17:    {Store neighbors of $\triangle\mathbf{t}_p$ to $N$}
18:    $N := Neighbor(\triangle\mathbf{t}_p)$;
19:    for $i = 0$ to length of $N$ do
20:        $\triangle\mathbf{t}$ :=$N(i)$;
21:        $(\lambda_1, \lambda_2, \lambda_3)$ := PhongProjection($\mathbf{q}_o$, $\triangle\mathbf{t}$);
22:        if $(\lambda_i \geq 0, i = 1, 2, 3)$ then
23:            {Phong Projection exists by Eq.(6)}
24:            $\mathbf{q}_c$ := $\triangle\mathbf{t}$.barycentric($\lambda_1, \lambda_2, \lambda_3$);
25:            return $\mathbf{q}_c$
26:        end if
27:    end for
28: end if
29: return NO_RESULT
```

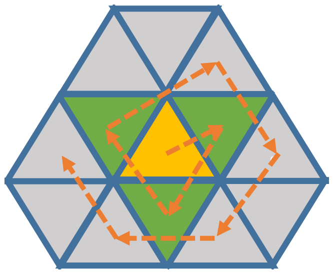


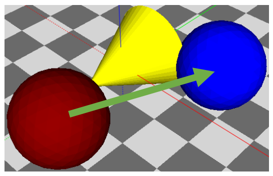
Fig. 4. The blue triangles shows a subset of C_{cont}. The yellow triangle is the result of Phong projection from the previous frame. Thus, we begin performing Phong-projection from the yellow triangle. The orange arrows denote the sequence of triangle-search. First, we perform Phong projection for the yellow triangle, and then move to the green triangles (i.e. 1-ring neighbors) which neighbor the yellow triangle, and then the gray triangles (i.e. 2-ring neighbors).

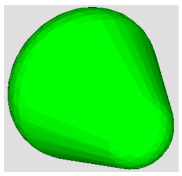
VI. Results and Discussion

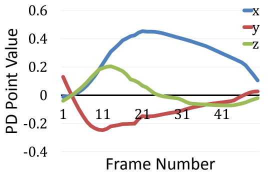
We now show our implementation results under various benchmarking scenarios and analyze the results

A. Implementation and Benchmarks

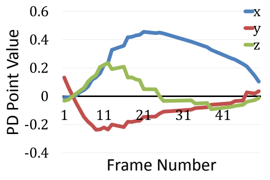
We have implemented our PhongPD algorithm using C++ programming language (Visual Studio 2012) under Windows 7, 32 bit operating system equipped with an Intel Core i7 2.67Ghz CPU and 3GB of main memory. We use the m+3d


(a) Ball and cone

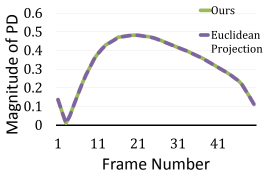

(b) Minkowski sum



(c) PhongPD direction

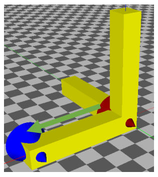


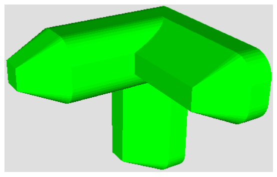
(d) Euclidean PD direction

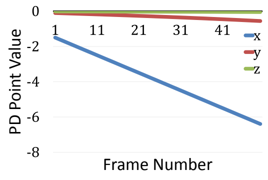


(e) PD magnitude

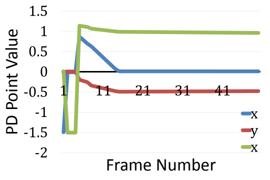
Fig. 5. Benchmark1: ball and cone. (a) The ball moves from the red to the blue position. And the ball collides with the yellow cone. (b) Minkowski sums of the ball and the cone (i.e. contact space). (c) x,y, and z components of PhongPD (d) x,y, and z components of Euclidean PD (e) PD magnitudes of PhongPD (i.e. $\sqrt{x^2+y^2+z^2}$)


(a) Cone and axes

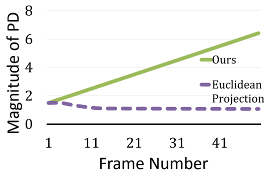

(b) Minkowski sum



(c) PhongPD direction

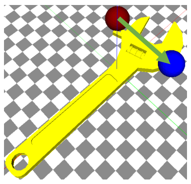


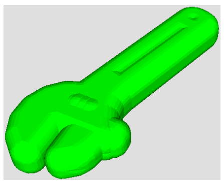
(d) Euclidean PD direction

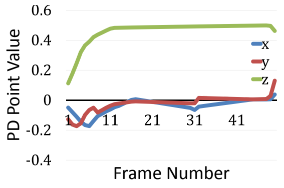


(e) PD magnitude

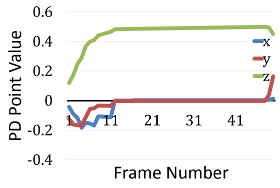
Fig. 6. Benchmark2: cone and axes. The cone moves along the green arrow in (a). The conventions in (b), (c), (d), (e) are identical to Benchmark1


(a) Ball and wrench

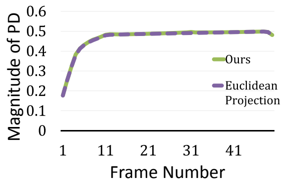

(b) Minkowski sum



(c) PhongPD direction



(d) Euclidean PD direction



(e) PD magnitude

Fig. 7. Benchmark3: ball and wrench. The ball moves along the green arrow in (a). The conventions in (b), (c), (d), (e) are identical to Benchmark1

library [17] to get C_{cont} of the general polygonal models and use [11] to perform Phong projection.

Our PhongPD algorithm was tested using three sets of benchmarks using different types of Minkowski sums whose complexity ranges from 440 to 4K triangles, as shown in Figs. 5, 6, and 7. We place two models $\mathcal{A}$, $\mathcal{B}$ in $\mathbb{R}^3$ and only $\mathcal{A}$ translates from the red-colored position to the blue one along the green arrow. And we track the history of the x, y, z components of PhongPD results. The Euclidean projection results (i.e. conventional PD) are also presented to compare with our algorithm. In these graphs, the x-axis denotes the frame number during the motion, and the y-axis denotes the magnitude of PhongPD .

- **Ball and Cone** (Fig. 5) The combinational complexity of the ball (movable) and the cone (fixed) are 1K and 78 triangles, respectively. And their Minkowski sums consist of 1035 triangles. The magnitude of PhongPD is similar to that of conventional PD, in this case.
- **Cone and Axes** (Fig. 6) The axes model (fixed) has 35 triangles. The Minkowski sums of the cone (movable) and the axes have 440 triangles. The half of the cone is penetrated into the axes. The magnitude of PhongPD is slightly greater than that of conventional PD, in this case.
- **Ball and Wrench** (Fig. 7) The wrench model (fixed) consists of 772 and the Minkowski sums with the ball (movable) has 4K triangles. The direction and magnitude of PhongPD are both similar to those of conventional PD.

B. Discussion

All the benchmarks in Figs. 5, 6, and 7 show that our PhongPD algorithms generates a smoother PD result than the Euclidean projection, and its PD direction is guaranteed to be continuous. Moreover, the magnitude of PhongPD is nearly the same as that of conventional PD for the ball/cone and ball/wrench benchmarks (Figs. 5(e) and 7(e)). However, it is not the case with the cone/axes benchmark (Figs. 6(e)), since the curvature of the Minkowski sums in this case is higher than others; in this case, the Phong-projected result traces out the high-curvature region to guarantee the gradient-continuity, while the Euclidean projection simply chooses the shortest one. For instance, Fig. 8 shows quite different Euclidean PD and PhongPD results.

Our algorithm has a few limitations. We assume that the orientation of $\mathcal{A}$ does not change and has only translational motion, which was also case with [9]. But unlike [9], our PhongPD can handle any type of translational configuration space. Our method does not always show gradient-continuous results when $\mathbf{q}_o$ is deep inside of C_{cont}, where Phong projection is undefined.

VII. Conclusion

We have presented a new algorithm for computing gradient-continuous penetration depth, called PhongPD.

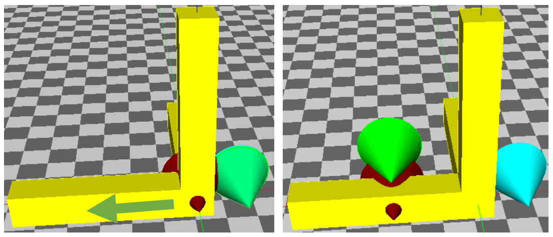


Fig. 8. A snapshot of the cone/axes benchmark. The red cone is initially located at a given configuration $\mathbf{q}_o$, and moves along the green arrow. The green cone is the result of conventional PD (Euclidean projection) while the cyan cone is that of PhongPD. Initially the green cone coincides with the cyan cone (left). However, as the red cone moves along the green arrow, the green cone jumps over the vertical bar and the cyan cone stays the same to guarantee the gradient-continuity (right).

Given a C_{cont}, we interpolate continuous tangent planes over C_{cont} and perform Phong projection to yield a gradient-continuous PD. We have also applied our algorithm to various benchmarks, and the experimental results show that PhongPD generates a smoother result than Euclidean PD. For future work, we would like to compute PhongPD that can handle deep penetration, as well as an orientation change of a model. Extending PhongPD to 6DoF will be also a challenging and interesting task. We also would like to apply our smooth PD results to penalty-based robot dynamics and haptic rendering to generate reliable simulation results.

Acknowledgment

This work was supported in part by NRF in Korea (2012R1A2A2A01046246, 2012R1A2A2A06047007, 2014K1A3A1A17073365) and MCST/KOCCA in the CT R&D program 2014 (R2014060011).

References

[1] S. Cameron and R. Culley, “Determining the minimum translational distance between two convex polyhedra,” in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, Apr 1986, pp. 591–596.

[2] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri, “Computing the intersection-depth of polyhedra,” *Algorithmica*, vol. 9, pp. 518–533, 1993.

[3] Y. Lee, S. Lengagne, A. Kheddar, and Y. J. Kim, “Accurate evaluation of a distance function for optimization-based motion planning,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[4] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, “Generation of whole-body optimal dynamic multi-contact motions,” *Int. J. Rob. Res.*, vol. 32, no. 9-10, pp. 1104–1119, Aug. 2013.

[5] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, “On finding narrow passages with probabilistic roadmap planners,” Natick, MA, USA, pp. 141–153, 1998.

[6] L. Zhang and D. Manocha, “An efficient retraction-based rrt planner,” in *In Proceedings IEEE International Conference on Robotics and Automation*, May 2008, pp. 3743–3750.

[7] L. Zhang, Y. J. Kim, and D. Manocha, “A fast and practical algorithm for generalized penetration depth computation,” in *In Proceedings of Robotics: Science and Systems Conference*, 2007.

[8] C. Je, M. Tang, Y. Lee, M. Lee, and Y. J. Kim, “Polydepth: Real-time penetration depth computation using iterative contact-space projection,” *ACM Transaction on Graphics*, vol. 31, no. 1, pp. 5:1–5:14, Feb. 2012.

[9] X. Zhang, Y. J. Kim, and D. Manocha, “Continuous penetration depth,” *SIAM Geometric and Physics Modeling*, 2013.

[10] B. T. Phong, “Illumination for computer generated pictures,” *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, 1975.

[11] D. Panozzo, I. Baran, O. Diamanti, and O. Sorkine-Hornung, “Weighted averages on surfaces,” *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, vol. 32, no. 4, pp. 60:1–60:12, 2013.

[12] P. K. Agarwal, L. J. Guibas, S. Har-peled, A. Rabinovitch, and M. Sharir, “Penetration depth of two convex polytopes in 3d,” *Nordic J. Computing*, vol. 7, pp. 227–240, 2000.

[13] G. Bergen, “Proximity queries and penetration depth computation on 3d game objects,” *Game Developers Conference*, 2001.

[14] Y. J. Kim, M. C. Lin, and D. Manocha, “DEEP: an incremental algorithm for penetration depth computation between convex polytopes,” *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 921–926, 2002.

[15] P. Hachenberger, “Exact minkowksi sums of polyhedra and exact and efficient decomposition of polyhedra into convex pieces,” *Algorithmica*, vol. 55, no. 2, pp. 329–345, 2009.

[16] Y. J. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha, “Fast penetration depth computation for physically-based animation,” in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. New York, NY, USA: ACM, 2002, pp. 23–31.

[17] J.-M. Lien, “A simple method for computing minkowski sum boundary in 3d using collision detection,” in *Algorithmic Foundation of Robotics VIII*, ser. Springer Tracts in Advanced Robotics, G. Chirikjian, H. Choset, M. Morales, and T. Murphey, Eds. Springer Berlin / Heidelberg, 2009, vol. 57, pp. 401–415.

[18] L. Zhang, Y. J. Kim, and D. Manocha, “C-dist: Efficient distance computation for rigid and articulated models in configuration space,” in *Proceedings of ACM symposium on Solid and physical modeling*. ACM, 2007, pp. 159–169.

[19] G. Nawratil, H. Pottmann, and B. Ravani, “Generalized penetration depth computation based on kinematical geometry,” *Computer Aided Geometric Design*, vol. 26, no. 4, pp. 425–443, May 2009.

[20] M. Tang and Y. J. Kim, “Interactive generalized penetration depth computation for rigid and articulated models using object norm,” *ACM Transactions on Graphics*, vol. 33, no. 1, pp. 1:1–1:15, Feb. 2014.

[21] J. Pan, X. Zhang, and D. Manocha, “Efficient penetration depth approximation using active learning,” *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2013)*, vol. 32, no. 6, pp. 191:1–191:12, Nov. 2013.

[22] E. Gilbert and C.-J. Ong, “New distances for the separation and penetration of objects,” in *In Proceedings IEEE International Conference on Robotics and Automation*, May 1994, pp. 579–586.

[23] C.-J. Ong and E. Gilbert, “Growth distances: new measures for object separation and penetration,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 888–903, Dec 1996.

[24] R. Weller and G. Zachmann, “Inner sphere trees for proximity and penetration queries.” in *Robotics: Science and Systems*, vol. 2, 2009.

[25] J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, and P. G. Kry, “Volume contact constraints at arbitrary resolution,” *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*, vol. 29, no. 3, Aug. 2010.

[26] A. Sud, N. Govindaraju, R. Gayle, I. Kabul, and D. Manocha, “Fast proximity computation among deformable models using discrete voronoi diagrams,” *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)*, vol. 25, pp. 1144–1153, 2006.

[27] M. Tang, M. Lee, and Y. J. Kim, “Interactive hausdorff distance computation for general polygonal models,” *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2009)*, vol. 28, no. 3, 2009.

[28] S. Redon and M. C. Lin, “A fast method for local penetration depth computation,” *Journal of graphics tools*, vol. 11, no. 2, pp. 37–50, 2006.

[29] D. Attali, J.-D. Boissonnat, and H. Edelsbrunner, “Stability and computation of medial axes - a state-of-the-art report,” in *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, ser. Mathematics and Visualization. Springer Berlin Heidelberg, 2009, pp. 109–125.

[30] L. Kobbelt, J. Vorsatz, and H.-P. Seidel, “Multiresolution hierarchies on unstructured triangle meshes,” *Computational Geometry: Theory and Applications (Special issue on multi-resolution modelling and 3D geometry compression)*, vol. 14, no. 1-3, pp. 5–24, Nov. 1999.