Continuous Penetration Depth

Xinyu Zhang [a,b] Young J. Kim [b] Dinesh Manocha [a]

[a] *The University of North Carolina at Chapel Hill, U.S.A.*

[b] *Ewha Womans University, Seoul, Korea*

Abstract

We present a new measure for computing *continuous penetration depth* between two intersecting rigid objects. We generate a set of samples in the configuration space, precompute an approximation of the *contact space* for two intersecting objects using binary classification techniques, and construct a bijective mapping between the spherical space and the precomputed contact space. For a given in-collision configuration, we search the spherical space for the nearest neighbor and find the corresponding image in the contact space based on the predefined spherical parameterization. The resulting image is a witness equivalent to the nearest configuration and is used to formulate the penetration depth direction based on our measure. Unlike prior algorithms, our algorithm guarantees that both the penetration depth magnitude and direction are continuous with respect to the motion parameters. Our algorithm is approximate in a sense that we approximate the exact contact space, and we have applied our algorithm to complex rigid models composed of tens or hundreds of thousands of triangles and the runtime query takes only around 0.01 milliseconds.

Key words: penetration depth, discontinuity, spherical parameterization

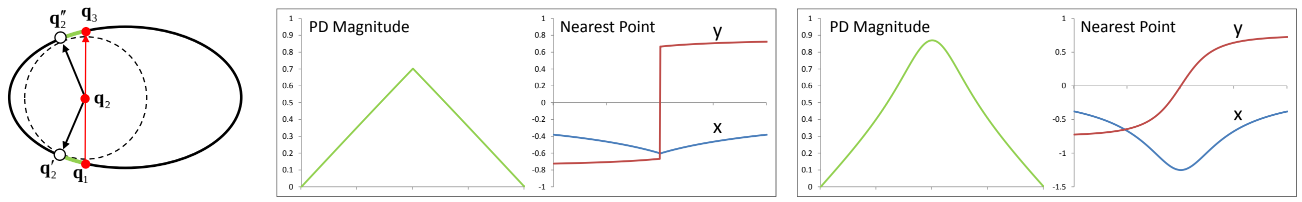
1 Introduction

Measuring the extent of inter-penetration between two intersecting objects is an important problem in physically-based simulation, haptic rendering, geometric computing, and robotics. Among these applications, a key computation is find a measure to separate two overlapping objects. It is necessary to change the configuration variables which describe the translational or even rotational motion of the objects to eliminate their intersections. The problem has been well studied in computer-aided design and robotics for more than three decades [1]. Perhaps the most natural measure for inter-penetration is *penetration depth* (PD), which is defined as the minimum rigid transformation required to separate two intersecting objects. This definition is widely used for contact resolution in dynamic simulation [2], tolerance verification for virtual prototyping [3], force computation in haptic rendering [4,5], motion planning in robotics [6], etc.

In many applications, the penetration depth query is often performed repeatedly as the objects move continuously along a path. The query result includes the magnitude and direction separating the intersecting objects [7]. However, the penetration depth computation can result in large discontinuities in terms of direction or even magnitude. These discontinuities, for instance, can significantly degrade the stability of the penalty response/force, which is computed as a function of penetration depth. The effects can include noticeable jittering and sudden local jumps in the computed force. Fig. 1 shows an example and illustrates this problem. For the optimization-based approaches [8,9,5], the penetration depth computation relies upon initial guesses and a series of projections on the local contact space. A discontinuity can occur in terms of both the magnitude and direction of penetration depth results. During the optimization, the local minimum can discontinuously vary with respect to the change of initial guesses. The discontinuity can become more severe when a contact space has many small surface features or local minima.

Main Results: In this paper, we present a new measure of inter-penetration which allows to efficiently computing continuous penetration depth between two intersecting rigid objects. Our new formulation of continuous penetration depth can guarantee that both the penetration depth magnitude and direction are continuous with respect to the motion parameters. This formulation relies on an implicit distance metric defined using a space transformation and a bijective mapping between the spherical space and the

Email addresses: zhangxy@cs.unc.edu (Xinyu Zhang), kimy@ewha.ac.kr (Young J. Kim), dm@cs.unc.edu (Dinesh Manocha).



(a) Contact Space (b) Discontinuous Penetration Depth (c) Continuous Penetration Depth

Fig. 1. The penetration depth is discontinuous based on a widely-used prior penetration depth definition, which corresponds to a nearest point from the given in-collision configuration to the contact space. (a) An ellipse has a major radius of 1.25 and a minor radius of 0.76, illustrating a contact space. A point (i.e., configuration) undergoes a continuous motion, while moving inside the ellipse. When the point passes through $\mathbf{q}_2$, its nearest point on the contact space has a sudden "jump" (i.e., discontinuity) from $\mathbf{q}_2'$ to $\mathbf{q}_2''$. (b) The penetration depth magnitude is not smooth and the nearest point is discontinuous at $\mathbf{q}_2$. The nearest point with its x and y coordinates are plotted in blue and red, respectively. (c) The penetration depth magnitude and nearest points resulted from our continuous penetration depth definition.

contact space of two intersecting objects. Given the high complexity of an exact contact space, we approximate the contact space using support vector machines and precompute the space transformation between the spherical space and the contact space using spherical parameterization, which can be precomputed. Our runtime algorithm uses a nearest neighbor query in the spherical space and the space transformation to approximate the continuous penetration depth. Our approach has been tested using complex models with tens or hundreds of thousands of triangles. The computational overhead for the performance of runtime query is almost constant and takes about 0.01 milliseconds to compute continuous penetration depth for these complex models.

Compared to prior approaches, our new measure of inter-penetration and the derived approximation algorithm have the following advantages:

(i) Continuity with respect to object motions; the penetration depth magnitude and direction continuously change for a continuous relative motion in configuration space.
(ii) Differentiability with respect to the motion parameters. The derivatives of continuous penetration depth are well-defined if the contact space is smooth and the motion is continuous.
(iii) Applicability to convex and non-convex contact spaces that are homeomorphic to a sphere.

Organization: The rest of the paper is organized in the following manner. We survey related work on penetration depth formulations and techniques in Section 2. We introduce our terminology and derive the formulation of our continuous penetration depth in Section 3. Our methods for approximating a contact space using binary classification and for precomputing spherical parameterization are given in Section 4. Our runtime penetration depth query is given in Section 5. Some experimental results and comparisons are given in Section 6.

2 Related Work

There exists an extensive amount of work on the penetration depth computation in computer-aided design and robotics, and a few measures and algorithms have been proposed. Some authors used an intersecting volume as a measure of inter-penetration [10–12]. However, this measure may not be able to correctly reflect the proximity situation when one object is completely inside in the other [10]; the measure remains constant regardless of the amount of inter-penetration. Moreover, a relatively small intersection between two complex objects may correspond to a deep penetration. Thus the volume of intersection may not provide an accurate measure of inter-penetration.

The growth distance [13,14] is a measure of inter-penetration between convex objects and is based on the idea of "growing" objects with respect to an interior seed point. The measure corresponds to appropriate scaling of the two objects to be just touching. This measure is continuous with respect to an object motion. However, this formulation is only applicable to convex objects. A similar measure was proposed based on shrinking one of the objects toward a reference point on the object until the objects are touching [15]. However, their measure is not continuous with respect to a continuous motion.

When the objects are in motion relative to one another, a motion trajectory-dependent penetration depth was proposed in [15]. Their measure of inter-penetration is defined as the distance that the object must move backward along the same path it used to approach before the objects are no longer inter-penetrating, but just "touching".

For convex polytopes, the penetration depth can be computed using the Minkowski sum [16,17,3] and the penetration depth is defined as the shortest distance between the origin and the boundary of Minkowski sum of two convex objects [1,10]. For non-convex objects, there are various algorithms to compute their penetration depth. The penetration depth can be computed by decomposing each object into convex primitives, computing pairwise Minkowski sums of the convex primitives, and computing the union of all convex Minkowski sums [18]. However, the union computation has a high computational complexity. Other acceleration techniques use rasterization hardware [18]. There are some algorithms to compute only local penetration depth, which computes a transformation to separate locally intersecting features [19–24]. Distance fields can also be used for local translational PD computation [7] and can be computed in realtime using GPUs. Point-based Minkowski sum approximation [25] can be used to approximate translational penetration depth for non-convex objects. None of these algorithms can consistently avoid the problems of discontinuity.

Exact generalized PD can be computed by constructing the exact contact space and then searching the contact space to find a nearest point for a given query [6]. However, due to high time and space complexity, most generalized PD algorithms use optimization-based techniques [8,9,5], computing a locally optimal solution based on local approximation of the contact space. Convex decomposition techniques can be used to compute lower and upper bounds on generalized PD [6]. Recently, a new machine learning method was suggested to approximate the contact space for PD computation [26]. None of them takes account of continuity.

3 Continuous PD Formulation

In this section, we first introduce our notation, then introduce the conventional penetration depth formulation in terms of configuration space and analyze its discontinuity. We present our new formulation for computing continuous penetration depth using spherical space transformation, which avoids the problems with conventional discontinuous formulation and local contact space methods.

3.1 Contact Space

Given two objects A and B, we denote their configuration space as $\mathcal{C}$-Space. Each point (i.e., configuration) in $\mathcal{C}$-Space corresponds to the relative configuration of A with respect to B. For the rest of the paper, we assume that A is movable and B is fixed. Moreover, we limit A to having a translational motion. In this case, $\mathcal{C}$-Space has 2 degrees of freedom (DOF) for 2D objects and it has 3-DOF for 3D objects. $\mathcal{C}$-Space is composed of two components: *collision-free* space $\mathcal{C}_{free} = \{\mathbf{q} : A(\mathbf{q}) \cap B = \emptyset\}$ and *in-collision* space $\mathcal{C}_{col} = \{\mathbf{q} : A(\mathbf{q}) \cap B \neq \emptyset\}$, where $A(\mathbf{q})$ corresponds to A located at the configuration $\mathbf{q}$. The *Contact Space* is the boundary of $\mathcal{C}_{col}$ and is denoted as $\mathcal{C}_{cont} = \partial C_{col}$. Intuitively, the contact space corresponds to a set of configurations where A and B just touch each other without any inter-penetration.

3.2 Discontinuous PD Formulation

The conventional penetration depth is defined as a minimum motion or transformation required to separate two intersecting objects A and B [17,3]. The penetration depth consists of two components: magnitude and direction. The underlying penetration depth formulation is defined as

$$|\mathrm{PD}(A(\mathbf{q}_0), B)| = \min_{\mathbf{q} \in \mathcal{C}_{cont}} \mathrm{dist}(\mathbf{q}_0, \mathbf{q}), \tag{1}$$

where $\mathbf{q}_0$ is an in-collision configuration ($\mathbf{q}_0 \in \mathcal{C}_{col}$) and $\mathbf{q}$ is a configuration that lies on the contact space $\mathcal{C}_{cont}$. We use $\mathrm{dist}(\cdot,\cdot)$ to represent a distance metric between two configurations. The contact configuration at which $|\mathrm{PD}(A,B)|$ attains its minimal value is denoted as

$$\mathbf{q}_0^{\mathcal{C}} = \mathrm{argmin}_{\mathbf{q} \in \mathcal{C}_{cont}} \mathrm{dist}(\mathbf{q}_0, \mathbf{q}). \tag{2}$$

$\mathbf{q}_0^{\mathcal{C}}$ is the nearest point from $\mathbf{q}_0$ to $\mathcal{C}_{cont}$, which realizes the $|\mathrm{PD}(A,B)|$. We use the following metric for computing the magnitude of penetration depth

$$\mathrm{dist}(\mathbf{q}_i, \mathbf{q}_j) = q_1^2 + q_2^2 + q_3^2, \tag{3}$$

where (q_1, q_2, q_3) corresponds to a relative translation between $\mathbf{q}_i$ and $\mathbf{q}_j$.

3.3 Discontinuity

It is clear that the magnitude of penetration depth defined by Eq. 1 is continuous because it is a single-valued continuous mapping. However, the direction of penetration depth defined by Eq. 2 is a multi-valued mapping between $\mathcal{C}_{col}$ and $\mathcal{C}_{cont}$. There can be more than one configuration in the contact space corresponding to the same magnitude of penetration depth. The discontinuity is intrinsic in the penetration depth definition corresponding to Eqs. 1 and 2, and the discontinuity is also related to the medial axis [27] of the in-collision space. Whenever a point crosses the medial axis, a discontinuity occurs. Since the medial axis has at least two nearest points on the boundary, there exists a sudden transition from one nearest point on the boundary to another when a query point crosses the medial axis (see Fig.1). In case of separation distance computation, a continuity can be guaranteed by using strictly convex objects such as sphere-torus patches bounding volumes (STPBV) [28] or k-IOS [29], but it does not work for penetration depth computation.

3.4 Continuous PD Formulation

The necessity for an alternative solution is obvious. If we have a new metric $\widetilde{\mathrm{dist}}(\cdot,\cdot)$ and under this metric Eq. 2 becomes a simple-valued mapping, there will be only one configuration in the contact space that realizes the penetration depth, and it is continuous. In order to simplify our derivation, we assume that the contact space is homeomorphic to a sphere. This assumption corresponds to the contact spaces used in our benchmarking models shown in Section 6. Unlike the explicit metric given in Eq. 3, our new metric is implicitly defined via an intermediate spherical space. Moreover, the medial axis of a sphere is reduced to a simple point (its center), which motivates us to choose spherical space as the intermediate space.

Now, we present our continuous penetration depth formulation using the contact space and the spherical space transformation. Let

$$\phi : \mathcal{S} \mapsto \mathcal{C}_{cont} \tag{4}$$

be a bijective mapping that maps a point of the spherical space $\mathcal{S}$ onto the one on the contact space $\mathcal{C}_{cont}$. Assume there exists another bijective mapping

$$\varphi : \mathbf{enc}(\mathcal{S}) \mapsto \mathbf{enc}(\mathcal{C}_{cont}) \tag{5}$$

that maps the space enclosed by $\mathcal{S}$ onto the one enclosed by $\mathcal{C}_{cont}$, where **enc** is the set of all points enclosed by a space including the space boundary. Given the new metric

$$\widetilde{\text{dist}}(\mathbf{q}_0, \mathbf{q}) = \text{dist}(\varphi^{-1}(\mathbf{q}_0), \phi^{-1}(\mathbf{q})), \tag{6}$$

our new measure of penetration depth can be analogously defined as

$$\mathbf{q}_0^{\mathcal{C}} = \phi(\text{argmin}_{\mathbf{q}\in\mathcal{C}_{cont}} \widetilde{\text{dist}}(\mathbf{q}_0, \mathbf{q})) \tag{7}$$

and

$$|\text{PD}(A(\mathbf{q}_0), B)| = \text{dist}(\mathbf{q}_0, \mathbf{q}_0^{\mathcal{C}}), \tag{8}$$

where $\mathbf{q}_0$ is an in-collision configuration and $\mathbf{q}$ is a configuration that lies on the contact space $\mathcal{C}_{cont}$ and dist$(\cdot,\cdot)$ is defined by Eq. 3.

Fig. 2 illustrates our new definition and its geometric interpretation. Intuitively, $\mathbf{q}_0^{\mathcal{S}} = \phi^{-1}(\mathbf{q}_0^{\mathcal{C}})$ is obtained by finding the nearest point in the intermediate space $\mathcal{S}$ from $\mathbf{q}_0$, and its image in $\mathcal{C}_{cont}$ is computed using the bijective mapping ϕ. If the intermediate space is the same as $\mathcal{C}_{cont}$, it is the same as the conventional penetration depth formulation given in Eqs. 1 and 2.

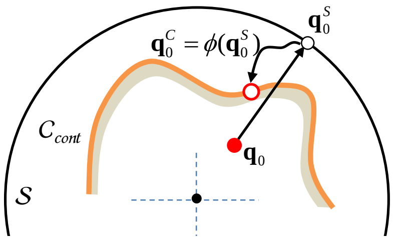


Fig. 2. Continuous penetration depth formulation. $\mathbf{q}_0$ is an in-collision configuration, $\mathbf{q}_0^{\mathcal{S}}$ is a point in the spherical space that attains its minimal Euclidian distance dist$(\cdot,\cdot)$ for $\mathbf{q}_0$. ϕ maps the point $\mathbf{q}_0^{\mathcal{S}}$ of the spherical space to a point $\mathbf{q}_0^{\mathcal{C}}$ in the contact space. We assume the mapping φ^{-1} is self-mapping, $\varphi^{-1}(\mathbf{q}_0) = \mathbf{q}_0$. The magnitude of penetration depth is defined as dist$(\mathbf{q}_0, \mathbf{q}_0^{\mathcal{C}})$.

3.5 Proof of Continuity

Theorem 1 *Our new PD formulation defined in Eqs. 5∼ 8 is continuous for any given in-collision configuration $q \in \mathcal{C}_{col}$ (except the sphere center).*

Proof: Based on an assumption that the bijective mapping $\phi : \mathcal{S} \mapsto \mathcal{C}_{cont}$ is continuous, so is its inverse mapping $\phi^{-1} : \mathcal{C}_{cont} \mapsto \mathcal{S}$. For any in-collision configuration point $\mathbf{q}_0$ undergoing a continuous motion, the Euclidean distance from the configuration to the spherical space is

$$\min_{\mathbf{q}\in\mathcal{S}} \text{dist}(\mathbf{q}_0, \mathbf{q}) = r - \|\mathbf{q}_0\|. \tag{9}$$

where r is the sphere radius. It is obvious this distance is continuous for any $\mathbf{q}_0$ (except the sphere center). The nearest point that realizes this distance is

$$\mathbf{q}_0^{\mathcal{S}} = r\frac{\mathbf{q}_0}{\|\mathbf{q}_0\|}. \tag{10}$$

It is not difficult to understand that, for any given continuous query $\mathbf{q}_0$ except the sphere center, its nearest point $\mathbf{q}_0^{\mathcal{S}}$ is continuous. Since the bijective mapping $\phi : \mathcal{S} \mapsto \mathcal{C}_{cont}$ is continuous, the nearest point in the contact space $\mathcal{C}_{cont}$ is continuous (Eq. 7). Then its distance computed using the nearest point (Eq. 8) is continuous. □

The sphere center is a singular point that has an infinite number of nearest points on the spherical space. However, we can easily handle this problem by avoiding going through the sphere center.

Fig. 3 illustrates the intuitive interpretation of continuity. When the object A moves along a continuous path (red) which connects $\mathbf{q}_1$ and $\mathbf{q}_2$, the witness (nearest) point that realizes the minimum distance in the spherical space continuously moves from $\mathbf{q}_1^{\mathcal{S}}$ to $\mathbf{q}_2^{\mathcal{S}}$ (blue path) while the witness point of contact space that realizes the final PD moves continuously from $\mathbf{q}_1^{\mathcal{C}}$ to $\mathbf{q}_2^{\mathcal{C}}$ (green path).

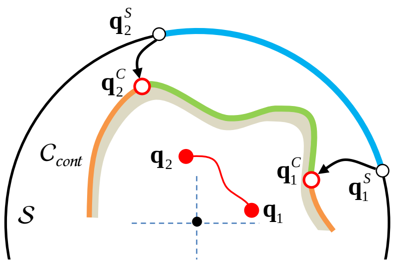


Fig. 3. Intuitive interpretation of continuity. For a continuous in-collision motion, our continuous PD formulation (Eqs. 7 and 8) can guarantee the continuity of PD magnitude and nearest points.

Algorithm 1 CPD(A, B, $\mathbf{q}_0$)
Comment: compute a continuous PD for two objects
Output: [|PD|, $\mathbf{q}_0^{\mathcal{C}}$]
|PD|: return the PD magnitude
$\mathbf{q}_0^{\mathcal{C}}$: return the nearest point on contact space

```
   //precomputation phase
1: $\mathcal{C}_{cont} \leftarrow$ ApproximateContactSpace($A$, $B$)
2: $\phi \leftarrow$ ComputeSphereicalParameterization($\mathcal{C}_{cont}$, $\mathcal{S}$)

   //runtime query phase
3: EmbedAndAlign($\mathcal{S}$, $\mathcal{C}_{cont}$, $\phi$)
4: $\mathbf{q}_0^{\mathcal{S}} \leftarrow$ NearestPoint($\mathcal{S}$, $\mathbf{q}_0$)
5: $\mathbf{q}_0^{\mathcal{C}} \leftarrow$ SphericalParamterization $\phi(\mathbf{q}_0^{\mathcal{S}})$
6: |PD| $\leftarrow$ DIST($\mathbf{q}_0$, $\mathbf{q}_0^{\mathcal{C}}$)
```

3.6 Algorithm Overview

Our algorithm consists of two main phases: a) precomputation, and (b) runtime PD query.

Precomputation: For two given intersecting objects, we precompute their contact space and then compute its spherical space transformation. The time complexity of exact contact space computation in 3D can be very high for two non-convex input models. Thus we use binary classification techniques to efficiently approximate the contact space. Then we use a spherical parameterization to compute the space transformation between the approximate contact space and the spherical space.

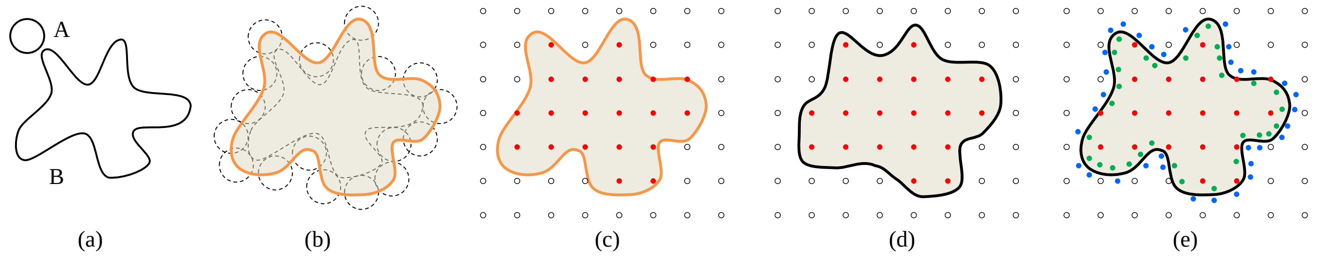


Fig. 4. Approximating a contact space using binary classification. (a) two given objects; (b) their exact contact space (orange); (c) initial uniform sampling, where circles and red solid dots denote collision-free and in-collision samples, respectively; (d) initial approximation; (e) refinement using new samples, where green and blue solid dots denote in-collision and collision-free samples, respectively.

Runtime Query: After computing the contact space and spherical parameterization, we embed the contact space into the space enclosed by the spherical space under appropriate alignment. Given a query $\mathbf{q}_0$, we first search the spherical space for a point $\mathbf{q}_0^{\mathcal{S}}$ that has the minimum distance. An image $\mathbf{q}_0^{\mathcal{C}}$ in the contact space can be uniquely found for the point of the spherical space. Finally, the distance between $\mathbf{q}_0$ and $\mathbf{q}_0^{\mathcal{C}}$ is computed using an appropriate metric dist$(\mathbf{q}_0, \mathbf{q}_0^{\mathcal{C}})$.

The corresponding pseudocode is given in Alg. 1.

4 Precomputing Contact Space and Spherical Parameterization

4.1 Approximating Contact Space

The time complexity of exact contact space computation in 3D can be as high as $O(m^3n^3)$ for translational PD, where m and n are the number of triangles for two non-convex input models. Given the high complexity of exact contact space computation, many approximate algorithms have been proposed [25,30,5]. However, these methods are either too slow or generate only a local contact space approximation. Here, we use a new method suggested in [26] to approximate the contact space. To guarantee the consistent continuity, we assume that the resulting approximation is homeomorphic to the exact contact space. Our method uses a binary classification algorithm to accelerate the computation of the contact space.

The algorithm is illustrated in Fig. 4. For a pair of objects A and B (Fig. 4-(a)), our goal is to compute their contact space (Fig. 4-(b)). We first uniformly generate a small set of configuration samples in a subspace of $\mathcal{C}$-space that covers the entire in-collision space $\mathcal{C}_{col}$ (Fig. 4-(c)). Next, we classify these configurations into two classes, collision-free ($\mathcal{C}_{free}$) or in-collision ($\mathcal{C}_{col}$), by performing exact collision detection at the sampled configurations. Using the classified configuration samples, an initial approximation of contact space (Fig. 4-(d)) is computed using a binary classifier based on support vector machines (SVM). However, since this initial approximation is too coarse to be used immediately, we add more new samples to refine the classification model (Fig. 4-(e)). Then we update the approximation (the black curve in Fig. 4-(e)). This sampling process can be further accelerated using active machine learning techniques such as [26].

4.1.1 SVM-based Classifier

A support vector machine (SVM) is a classification technique. It is straightforward to implement and computationally efficient.

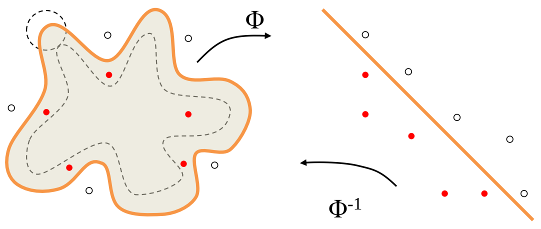


Fig. 5. The original data is transformed to another data in a high dimensional feature space using the mapping Φ. An optimal hyperplane is found in feature space and mapped back to input space.

As shown in Fig. 5, the core of SVM is to use a mapping that transforms the original data in input space to the data in feature space, so that the classification in input space is reduced to a linearly separable problem in feature space. Then the optimal separating hyperplane in feature space is mapped back into input space via its inverse mapping. Though the separating hyperplane is linear in feature space, the resulting surface in input space can have a very high complexity. Let Φ be a mapping function from input $\mathcal{C}$-Space (configuration space) into a feature space $\mathcal{H}$. We assume $\mathcal{C}$ to be $\mathcal{R}^n (n \leq 3)$ and $\mathcal{H}$ to be $\mathcal{R}$. Let $K(\mathbf{q}_i, \mathbf{q}_j) = \langle \Phi(\mathbf{q}_i), \Phi(\mathbf{q}_j) \rangle$ be the kernel function. Essentially, K is a function used to calculate inner products in feature space. In our algorithm, we use the radial basis function (RBF) as the kernel

$$K(\mathbf{q}_i, \mathbf{q}_j) = \exp(\gamma \|\mathbf{q}_i - \mathbf{q}_j\|^2), \tag{11}$$

where γ is a positive parameter. Then the classifier can be modelled using an function

$$f(\mathbf{q}) = \mathbf{w} \cdot \Phi(\mathbf{q}) + b \tag{12}$$

where $\mathbf{w} \in \mathcal{H}$ and $b \in \mathcal{R}$. $f(\mathbf{q}) = 0$ is called a decision boundary. The formulation can be rewritten as

$$f(\mathbf{q}) = \sum_{i=1}^{k} \alpha_i c_i K(\mathbf{q}_i, \mathbf{q}) + b, \tag{13}$$

where $\alpha_i \geq 0$. A few α_i's are non-zero and the corresponding $\mathbf{q}_i$ are the *support vectors*. c_i are collision states (either -1 or $+1$). This SVM problem can be solved using sequential minimal optimization (SMO) algorithm [31,32]. For more details on SVM, we refer readers to [33,34].

4.1.2 Sampling

We propose a tree-based method for efficiently adding new samples in the configuration space and refining the initial contact space approximation. As shown in Fig. 6, an octree is first constructed for the decision boundary of the initial SVM model. For each node in the tree, we predict the collision states of its center and all the neighbor corners using the initial SVM model $f(\mathbf{q})$. If these states do not have the identical signs and the distance from the node center to the decision boundary ($f(\mathbf{q}) = 0$) is larger than a threshold, the node needs to be further split. Since $f(\mathbf{q})$ is a zero set function, it is nontrivial to compute the exact distance to the decision boundary. Therefore, we use the following strategy to compute the lower bound of distance for a given configuration. The idea is based on an observation that it is trivial to compute the distance from $\mathbf{q}$ to $f(\mathbf{q}) = 0$ in the feature space, where $f(\mathbf{q}) = 0$ is in the form of a hyperplane:

$$\begin{aligned} d_{ft} &= \operatorname{dist}(\Phi(\mathbf{q}), \operatorname{HyperPlane}(\mathbf{w}, b)) \\ &= \frac{|\mathbf{w} \cdot \Phi(\mathbf{q}) + b|}{\|\mathbf{w}\|_2} = \frac{|f(\mathbf{q})|}{\sqrt{\sum_{i,j} \alpha_i \alpha_j c_i c_j K(\mathbf{q}_i, \mathbf{q}_j)}}. \end{aligned} \tag{14}$$

Since we use the RBF as the kernel, the distance d_{ft} in the feature space corresponds to a lower bound of the distance d_{in} in input space

$$d_{in} \geq -\frac{1}{\gamma} \ln\left(1 - \frac{d_{ft}^2}{2}\right). \tag{15}$$

After the tree structure is computed, the centers of all the leaf nodes are the candidates for new samples near the decision boundary. We use k-centroid clustering algorithm to select the required number of samples from these candidates.

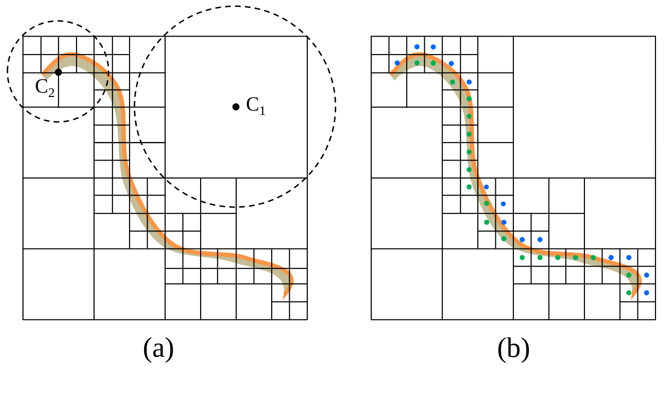


Fig. 6. (a) We approximate the initial decision boundary (the orange curve) using an octree structure. (b) The leaf nodes close to the decision boundary are returned as new samples (green for in-collision samples and blue for collision-free samples).

To further accelerate the computation, we can use active learning and non-uniform sampling techniques [26].

4.2 Precomputing Spherical Parameterization

Given a SVM model and its decision boundary $f(\mathbf{q}) = 0$, we convert it into a triangular mesh and perform mesh parameterization. A parameterization is a bijective mapping between a surface and a parameter domain. If the surface and the parameter domain have the same topology, then such a bijective mapping is guaranteed to exist. In this reason, we assume that the resulting contact space is homeomorphic to a sphere and we apply spherical parameterization to the contact space.

4.2.1 Convex Contact Space

Since any compact convex space with a nonempty interior is homeomorphic to a closed ball [35], a spherical parameterization for a convex contact space can be easily computed by projecting each point of $\mathcal{C}_{cont}$ onto the sphere along the radial direction. $\phi : \mathcal{S} \mapsto \mathcal{C}_{cont}$ can be expressed as follows

$$\phi^{-1}(\mathbf{q}) = r\frac{\mathbf{q}}{\|\mathbf{q}\|_2}, \mathbf{q} \in \mathcal{C}_{cont}. \tag{16}$$

Due to convexity, the simple projection $\phi : \mathcal{S} \mapsto \mathcal{C}_{cont}$ is bicontinuous.

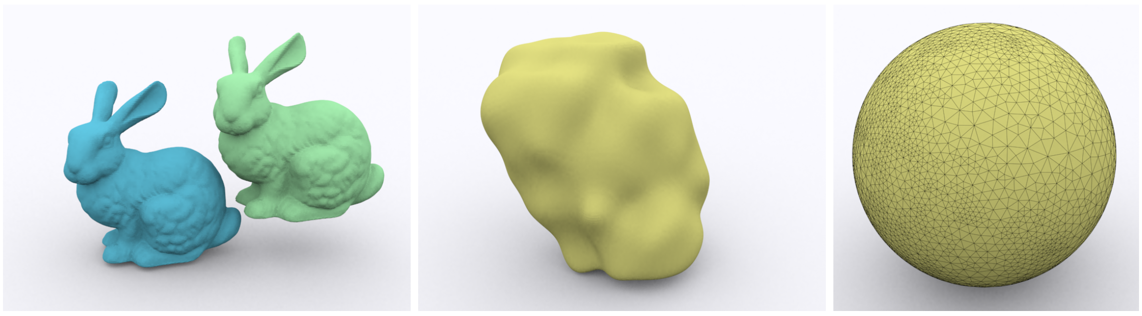
4.2.2 Non-Convex Contact Space

Here, we consider the contact space embedding in $\mathcal{R}^3$ and the intermediate space being a spherical surface in $\mathcal{R}^3$. Given a contact surface $\mathcal{C}_{cont}$, the spherical parameterization is to find a continuous invertible map $\phi : \mathcal{S} \mapsto \mathcal{C}_{cont}$ from a sphere $\mathcal{S}$ to the contact surface $\mathcal{C}_{cont}$. For a given approximate contact space represented by a mesh, the map is specified by assigning each point $\mathbf{q}$ a parameterization $\phi^{-1}(\mathbf{q}) \in \mathcal{S}$. Each mesh edge is mapped to a great circle arc, and each mesh triangle is mapped to a spherical triangle bounded by these arcs. If we assume that contact surface $\mathcal{C}_{cont}$ is homeomorphic to a sphere, then the map is continuous and one-to-one. To guarantee a valid spherical parameterization, each vertex position must be expressed as a convex combination of the positions of its neighbors projected on the sphere [36]. Let $(\mathbf{V}, \mathbf{E})$ be a mesh in $\mathcal{R}^3$ if the contact space is represented as a triangulation, where $\mathbf{V}$ is a set of vertices and $\mathbf{E}$ is a set of edges. The convex combination can be expressed with respect to the neighborhood of a vertex.

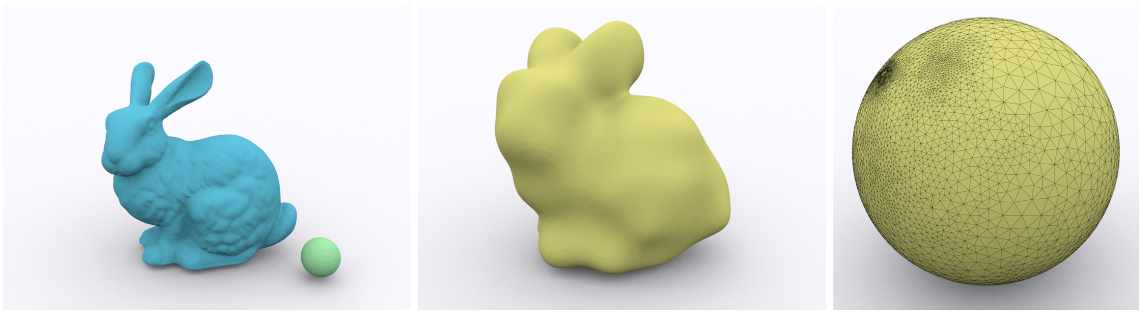
$$\begin{aligned} &\mathbf{v}_i = \sum_{i=1}^{k} \lambda_{ij}\mathbf{v}_j, \quad \mathbf{v}_i, \mathbf{v}_j \in \mathbf{V}, \\ &\lambda_{ij} = 0 \quad \text{if } \mathbf{e}_{ij} \notin \mathbf{E}, \\ &\lambda_{ij} > 0 \quad \text{if } \mathbf{e}_{ij} \in \mathbf{E}, \\ &\sum_{i=1}^{k} \lambda_{ij} = 1. \end{aligned} \tag{17}$$

where $\mathbf{e}_{ij} \in \mathbf{E}$ is the edge connecting $\mathbf{v}_i$ and its neighbor $\mathbf{v}_j$.

Under the above convex condition, none of the triangles are allowed to overlap. In the 2D planar case, once



(a) bunny-bunny models, their contact space and spherical parameterization



(b) bunny-sphere models, their contact space and spherical parameterization

Fig. 7. The contact space approximated by a SVM-based classifier and space transformation using spherical parameterization.

the boundary of the triangulation has been fixed and the barycentric coordinates are chosen, the positions of the interior vertices are uniquely determined by solving a linear system. Uniqueness may not be guaranteed for the spherical case, which requires more constraints. First, the spherical triangulation may be controlled by choosing a proper set of symmetric weights. The above system can be expressed as a set of non-linear equations in terms of the vertices of a mesh after introducing k auxiliary scaling variables a_i and spherical distance constraint $v_{x,i}^2 + v_{y,i}^2 + v_{z,i}^2 = 1$ for any vertex $\mathbf{v}_i = (v_{x,i}, v_{y,i}, v_{z,i})$.

$$\begin{aligned} a_i \mathbf{v}_i &= \sum_{i=1}^{k} \lambda_{ij} \mathbf{v}_j \\ v_{x,i}^2 + v_{y,i}^2 + v_{z,i}^2 &= 1. \end{aligned} \tag{18}$$

These equations can be solved by iteratively minimizing the quadratic spring energy

$$W = \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} \lambda_{ij} \left\| \mathbf{v}_i - \mathbf{v}_j \right\|^2. \tag{19}$$

To avoid degenerate scenarios and have fast convergence, we can anchor one or two vertices to a fixed position on the sphere. In our implementation, we choose the farthest vertex from the origin and map it to the closet point on the sphere.

Theorem 2 $\phi : \mathcal{S} \mapsto \mathcal{C}_{cont}$ *is bicontinuous.*

Proof: This directly follows from the assumption that the genus-zero contact space is homeomorphic to a sphere. Therefore, $\phi : \mathcal{S} \mapsto \mathcal{C}_{cont}$ is a bijection, $\phi : \mathcal{S} \mapsto \mathcal{C}_{cont}$ is continuous and the inverse function $\phi^{-1} : \mathcal{C}_{cont} \mapsto \mathcal{S}$ is continuous ($\phi : \mathcal{S} \mapsto \mathcal{C}_{cont}$ is an open mapping). At the same time, based on our assumption on the homeomorphism between $\mathcal{C}_{cont}$ and its approximation, the mapping between $\mathcal{C}_{cont}$ and its approximation is bicontinuous, too. □

Note that any continuous motion on the spherical space $\mathcal{S}$ corresponds to a continuous trajectory on the approximate contact space $\mathcal{C}_{cont}$ and corresponds to another continuous trajectory on the exact contact space $\mathcal{C}_{cont}$ as long as the homeomorphism between these spaces remains valid.

Fig. 7 shows two examples of approximated contact space and their spherical parameterization. Note that we can use other spherical parameterization methods [37] to compute ϕ as long as bijective mapping (no local and global overlaps) can be guaranteed.

5 Runtime PD Queries

We use the approximated contact space and its corresponding spherical parameterization to perform runtime queries. Based on our new PD formulation (also refer to Fig. 2), a runtime query is very straightforward after space transformation. Given a configuration $\mathbf{q}_0 \in \mathcal{C}_{col}$, we find the nearest point in the spherical space by projecting $\mathbf{q}_0$ along radial direction. Let $\mathbf{q}_0^{\mathcal{S}} \in \mathcal{S}$ be this nearest point. $\mathbf{q}_0^{\mathcal{S}}$ can be computed Eq.10. Next, we search the spherical parameter domain for the spherical triangle containing $\mathbf{q}_0^{\mathcal{S}}$. Let this spherical triangle be $\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$. We compute $\mathbf{q}_0^{\mathcal{S}}$'s barycentric coordinates (w_1, w_2, w_3) with respect to the spherical triangle $\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$. Based on the spherical parameterization ϕ, the corresponding triangle in the contact space is $\phi(\mathbf{v}_1)\phi(\mathbf{v}_2)\phi(\mathbf{v}_3)$. Using $\mathbf{q}_0^{\mathcal{S}}$'s barycentric coordinates, the point of contact space corresponding to the nearest point $\mathbf{q}_0^{\mathcal{S}}$ is

$$\mathbf{q}_0^{\mathcal{C}} = w_1\phi(\mathbf{v}_1) + w_2\phi(\mathbf{v}_2) + w_3\phi(\mathbf{v}_3). \tag{20}$$

Since the spherical parameterization is represented as a triangular mesh, the barycentric coordinates can be computed by

$$\begin{aligned} w_1 &= \triangle\mathbf{q}_0^{\mathcal{S}}\mathbf{v}_2\mathbf{v}_3 / \triangle\,\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3 \\ w_2 &= \triangle\mathbf{q}_0^{\mathcal{S}}\mathbf{v}_3\mathbf{v}_1 / \triangle\,\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3 \\ w_3 &= \triangle\mathbf{q}_0^{\mathcal{S}}\mathbf{v}_1\mathbf{v}_2 / \triangle\,\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3 \end{aligned} \tag{21}$$

We use bounding volume hierarchies and hashing techniques [38] to accelerate the search for the triangle that contains $\mathbf{q}_0^{\mathcal{S}}$. Eventually, the PD magnitude can be computed using Eq. 8 for $\mathbf{q}_0$ and $\mathbf{q}_0^{\mathcal{C}}$.

For now, we only consider the continuity for a configurations inside $\mathcal{C}_{col}$. However, for a configuration $\mathbf{q}_0 \in \mathcal{C}_{cont}$, its PD magnitude may not be a non-zero value. When a configuration moves from the collision-free space into the in-collision space or moves out of the in-collision space to the collision-free space, a discontinuity may occur. We illustrate this problem in Fig. 8. Consider a continuous path from $\mathbf{q}_1$ to $\mathbf{q}_2$ ($\mathbf{q}_1, \mathbf{q}_2 \in \mathcal{C}_{cont}$). For the configurations $\mathbf{q}_1$ and $\mathbf{q}_2$, their nearest points in the spherical space $\mathcal{S}$ are initially $\mathbf{q}_1^{\mathcal{S}}$ and $\mathbf{q}_2^{\mathcal{S}}$. Then the corresponding points in the contact space are $\mathbf{q}_1^{C}$ and $\mathbf{q}_2^{C}$. If $\mathbf{q}_1^{C} \neq \mathbf{q}_1$, its PD magnitude will suddenly jump from zero to non-zero when $\mathbf{q}_1$ moves from $\mathcal{C}_{free}$ into $\mathcal{C}_{col}$. Similarly, when a point $\mathbf{q}_2$ moves out of $\mathcal{C}_{col}$ to $\mathcal{C}_{free}$, the PD magnitude may suddenly jump from non-zero to zero.

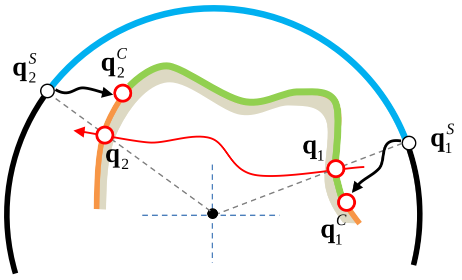


Fig. 8. A discontinuity may happen between the in-collision and collision-free spaces. The PD magnitude may suddenly jump from zero to non-zero when $\mathbf{q}_1$ moves from $\mathcal{C}_{free}$ into $\mathcal{C}_{col}$; when a configuration $\mathbf{q}_2$ moves out of $\mathcal{C}_{col}$ to $\mathcal{C}_{free}$, the PD magnitude may suddenly jump from non-zero to zero.

This discontinuity problem is mainly caused by the distortion of spherical parameterization and by our assumption that the mapping φ^{-1} is self mapping. As a result, for a configuration $\mathbf{q}_0 \in \mathcal{C}_{cont}$, the nearest point is not necessary to be itself and its PD magnitude can be a non-zero value. In order to eliminate such discontinuity, we use the following strategies.

Move in $\mathcal{C}_{col}$: The discontinuity can be avoid by introducing a rotation to the mapping ϕ between the spherical space space and the contact space during runtime query. As shown in Fig. 9, for a move-in configuration $\mathbf{q}_1$, we first find the nearest point $\mathbf{q}_1^{\mathcal{S}}$ in the spherical space. At the same time, for $\mathbf{q}_1$, we search for its corresponding point $\phi^{-1}(\mathbf{q}_1)$ in the contact space using spherical mapping ϕ. Then we rotate the spherical space by $\chi(\theta)$ so that $\phi^{-1}(\mathbf{q}_1) = \mathbf{q}_1^{\mathcal{S}}$. After applying the rotation to the spherical space, the nearest point is identical to the move-in point $\mathbf{q}_1$ and the PD magnitude becomes zero. When a configuration moves deep inside the in-collision space, we gradually and continuously relax the rotation applied to the spherical space so that the point $\phi^{-1}(\mathbf{q}_1)$ is restored to the original position. If we let the restoring region be $[0, \varepsilon]$, the following linear rotation allows us to continuously restore the original spherical space

$$\mathcal{S}' = \chi(\frac{\varepsilon - |\mathrm{PD}|}{\varepsilon}\theta)\mathcal{S}, \ |\mathrm{PD}| \in [0, \varepsilon].$$

Here $\mathcal{S}'$ is the spherical space after rotation.

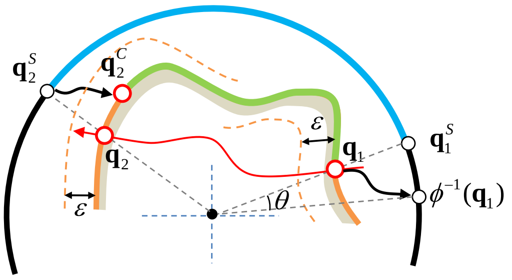


Fig. 9. For a move-in contact configuration $\mathbf{q}_1$, the discontinuity can be eliminated by rotating the spherical parameter space. For a move-out contact configuration $\mathbf{q}_2$, the discontinuity can be avoided by a linearly decreasing function in the collision-free space.

Move out of $\mathcal{C}_{col}$: The above strategy does not work for a configuration moving from the in-collision space to the collision-free space because we have little knowledge about the transitioning contact configuration a priori. Instead, we use a linearly decreasing function to eliminate the discontinuity when the configuration leaves the in-collision space. As shown in Fig. 9, if we let $\mathbf{q}_2$ be a move-out configuration, the PD magnitude at the contact point $\mathbf{q}_2$ is $|PD|$. When two objects separate (i.e. the configuration moves out of the in-collision space), let the distance between them be d. If their distance falls within a distance interval $[0, \varepsilon]$, we use the following formulation to diminish the discontinuity.

$$|\mathrm{PD}|' = \frac{\varepsilon - d}{\varepsilon}|\mathrm{PD}|$$

The new PD magnitude will gradually and continuous reduce to zero and the nearest point will remain the same.

6 Implementation and Performance

In this section, we give some implementation details, highlight the performance of our algorithm on some complex benchmarks and compare our algorithm with prior techniques.

6.1 Implementation

We implemented our algorithm using C++ under Visual Studio 2010 and Windows 7. We use the following algorithms and libraries in our implementation. The performance is measured on a PC with 3.2GHz Intel Core i7 CPU and 6G memory.

SVM-based Classifer: We used the open SVM libary SVMLIB [32] for approximating the contact surface.

Collision Detection: Bounding volume hierarchies (BVHs) are the most popular techniques and data structure to accelerate collision queries. Thus, we used the open library OBB-Tree [39] for exact collision detection between polygonal objects.

Spherical Parameterization: A non-manifold geometric library ARCHMIND [1] was used for spherical parameterization.

Others: We use the library PQP [2] [40] to perform distance computation. This is mainly used for avoiding discontinuity when a point moves from in-collision space to collision-free space. The open PD library PolyDepth [3] [5] based on local optimization and projection is used for comparison.

6.2 Benchmarks

We used the following benchmarks to evaluate the performance of our algorithm. Some of them are shown in Figs. 10∼13. It typically takes 0.5∼3 seconds to approximate the contact space using the binary classification technique, including collision detection and solving SVM classifier. It takes 2-10 seconds for spherical parameterization. The runtime query takes a nearly constant time, around 0.01 milliseconds.

We analyze the continuity in these benchmarking scenarios. In Figs. 10∼13, we use green curves to highlight the PD magnitude computed by our algorithm and gray curves to represent the result computed by PolyDepth. We use individual color to illustrate each coordinate of the nearest point. The x, y and z coordinates are plotted in blue, red and orange, respectively.

Hose vs. Point: The contact space between a model and a point is the model itself. Fig. 10 shows a hose model (4.2K triangles) and its spherical parameterization. When a point undergoes a continuous path (from top to bottom) inside the model, the PD magnitude and nearest points are shown in Fig. 10-(c)∼(d).

Dragon vs. Sphere: Fig. 11 shows a dragon model and a sphere model, where the sphere undergoes a continuous motion. The dragon model consists of 214K triangles and the sphere consists of 1.6K triangles. The intermediate contact surface and its parameterization are shown in Fig. 11-(b) and (c).

Cup vs. Spoon: We demonstrate our algorithm using the cup and spoon models. The cup and spoon consist of 1000 and 1344 triangles, respectively. In the top row of Fig. 12, the spoon is placed inside the cup and moves.

Buddha vs. Buddha: Fig. 13-(a) shows two Buddha models; one model is stationary and the other undergoes a continuous motion in the configuration space. Each Buddha model consists of 20K triangles. The intermediate contact surface and parameterization are shown in Fig. 13-(b) and (c).

6.3 Comparison with Prior Methods

In Section 2, we have highlighted different techniques for PD computation. In order to better understand the difference between our algorithm and the prior methods, we compared our algorithm with optimization-based methods [8,9,5]. In these methods, a sequence of configuration samples on the contact space are iteratively computed until a local minimum configuration is found. The performance of these algorithms relies heavily upon the initial configuration guesses. The continuity is not guaranteed in these algorithms. As shown in Figs. 10∼13, our PD algorithm provide continuous, smooth PD values compared to the most recent optimization-based work, PolyDepth [5].

The intersecting volume-based method can obtain continuous PD magnitude and direction when the normal cones are well defined [11,12]. However, it is not clear whether it can be extended to the cases where normal cones are not defined. For instance, when one object is completely embedded inside the other object, the normal cones cannot be defined using the techniques suggested in [12]. Growth distance [13,14] can achieve continuous penetration depth for convex objects, but the property of continuity may not be preserved for non-convex objects.

7 Limitations and Conclusions

Penetration depth computation is used in various collision response algorithms and many applications. Under the conditions of continuous motion, conventional penetration depth formulations exhibit a discontinuity with respect to the motion parameters, which can cause an instability in collision response. We have presented a new approach to computing continuous penetration depth between polygonal models. The main idea is based on contact space approximation and spherical parameterization.

The overall approach is simple (see Alg. 1). The basic components of our continuous penetration depth algorithm consist of the following routines: SVM-based classification, collision detection, nearest-neighbor computation, and spherical parameterization. Good implementations of these algorithms are easily available in public domain (see Section 6.1).

Our approach has a few limitations. The algorithm only handles translational penetration depth. Moreover, our algorithm currently computes an approximation to the continuous PD formulation in Eqs. 7 and 8 and its accuracy depends almost completely on the sufficiency of samples during classification. Without sufficient sampling, the assumption of homeomorphism between the exact

[1] http://www.cs.uoi.gr/˜fudos/smi2011.html
[2] http://gamma.cs.unc.edu/SSV/
[3] https://code.google.com/p/polydepth/

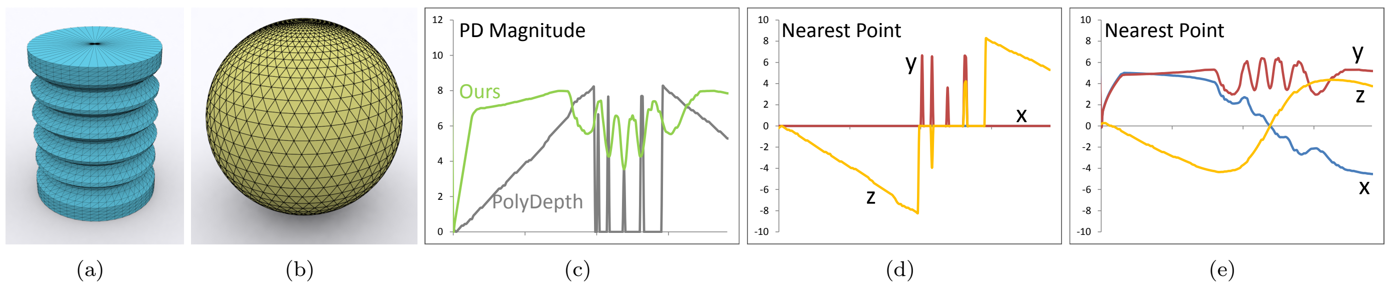


Fig. 10. (a) hose model (b) spherical parameterization (c) PD magnitude: Our algorithm vs. PolyDepth. (d) nearest points computed by PolyDepth; (e) nearest points computed by our continuous PD algorithm.

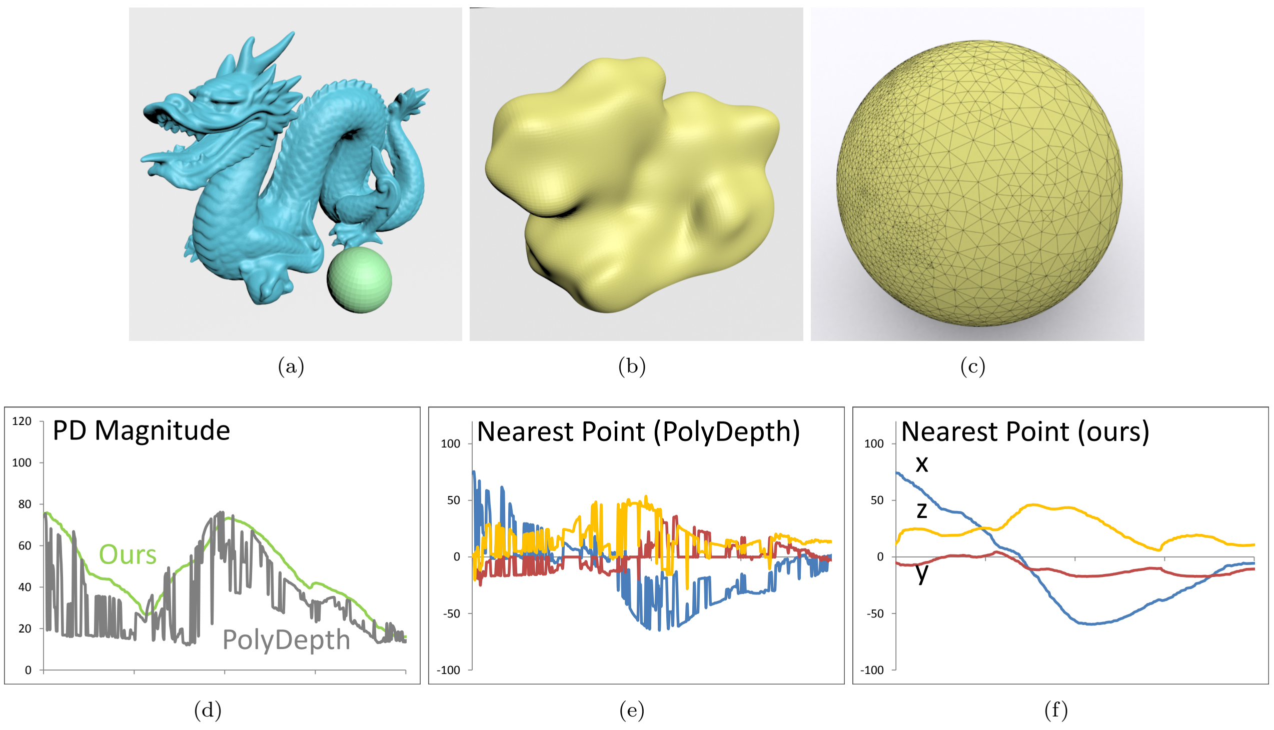


Fig. 11. (a) dragon and sphere; (b) contact space; (c) spherical parameterization (d) PD magnitude: Our algorithm vs. PolyDepth. (e) nearest points computed by PolyDepth; (f) nearest points computed by our continuous PD algorithm.

contact space and its approximation may not be valid. For example, a dumbbell shape contact space with a very thin neck may require very dense samples to preserve the shape using a SVM-classifier; otherwise, the approximate contact space may contain two disconnected components. In this case, the sampling techniques using adaptive and active strategies such as [26] can be a very promising alternative. Our algorithm is based on an assumption that there is homeomorphism between the contact space and spherical space. This simplification brings about a few benefits in terms of reducing the problem complexity, but it can be challenging to extend to high-genus contact spaces. It is not very desirable for the queries that require frequent recomputation of the contact space and the spherical parameterization since these two precomputation steps largely dominate the overall computation.

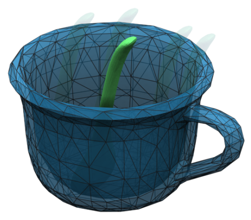
There are many avenues for future work. We would like to extend our algorithm to generalized penetration depth computation. The basic components of our precomputation and run-time phases (SVM-based classifier, collision detection, nearest-neighbor computation, and spherical parameterization) can be accelerated using GPU parallelism. It would be interesting to investigate the discontinuity with other PD formulations, such as intersecting volume [11] and growth distance [14] for non-convex models.

Acknowledgement

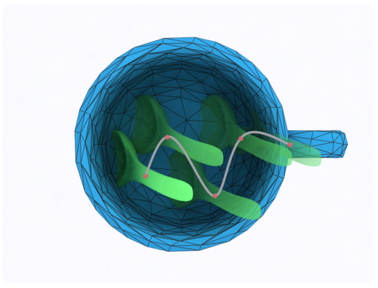
This research was supported in part by ARO under contract W911NF-10-1-0506, by NSF awards 1000579 and 1117127. This research was supported in part by NRF in Korea (No. 2012R1A2A2A01046246, No. 2012R1A2A2A06047007).

References

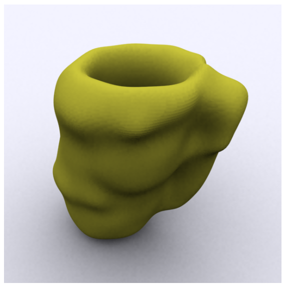
[1] S. Cameron, R. Culley, Determining the minimum translational distance between two convex polyhedra, in: Proceedings of IEEE



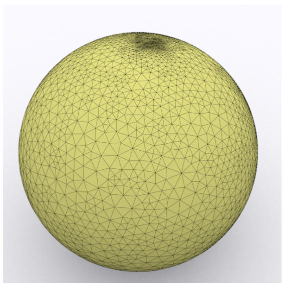
(a)



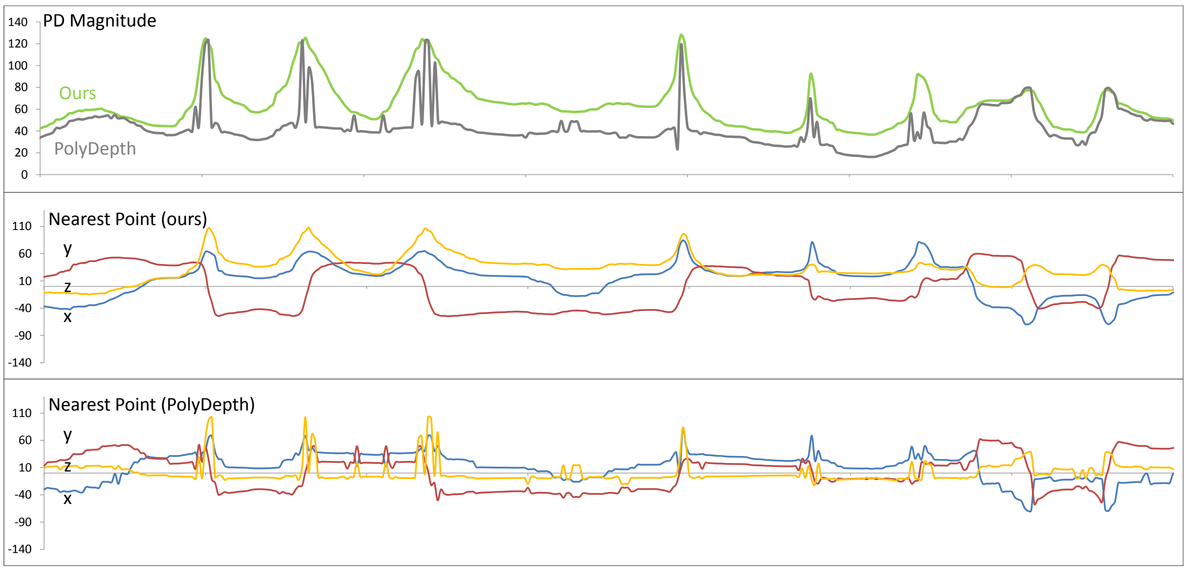
(b)



(c)



(d)



(e)

Fig. 12. (a) cup and spoon models; (b) different positions of the spoon along a continuous path; (c) the contact space approximation of cup and spoon. (d) the spherical parameterization of the approximate contact space. (e) the PD magnitude and nearest points. x, y and z coordinates are plotted in blue, red and orange, respectively.

International Conference on Robotics and Automation, 1986, pp. 591–596, volume 3.

[2] D. Baraff, A. Witkin, Physically Based Modeling, ACM SIGGRAPH Course Notes, 2001.

[3] Y. J. Kim, M. C. Lin, D. Manocha, DEEP: Dual-space expansion for estimating penetration depth between convex polytopes, in: Proceedings of International Conference on Robotics and Automation, 2002, pp. 921–926.

[4] D. Wang, S. Liu, X. Zhang, J. Xiao, Configuration-based optimization for six degree-of-freedom haptic rendering for fine manipulation, Transactions on Haptics.

[5] C. Je, M. Tang, Y. Lee, M. Lee, Y. J. Kim, Polydepth: Real-time penetration depth computation using iterative contact-space projection, ACM Transactions on Graphics 31 (1) (2012) 5:1–5:14.

[6] L. Zhang, Y. J. Kim, G. Varadhan, D. Manocha, Generalized penetration depth computation, Computer-Aided Design 39 (8) (2007) 625–638.

[7] B. Heidelberger, M. Teschner, R. Keiser, M. Mller, M. H. Gross, Consistent penetration depth estimation for deformable collision response, in: International Fall Workshop on vision, modeling and visualization, 2004, pp. 339–346.

[8] G. Nawratil, H. Pottmann, B. Ravani, Generalized penetration depth computation based on kinematical geometry, Computer Aided Geometric Design 26 (4) (2009) 425–443.

[9] L. Zhang, Y. J. Kim, D. Manocha, A fast and practical algorithm for generalized penetration depth computation, in: Robotics: Science and Systems, 2007.

[10] S. S. Keerthi, K. Shidharan, Measures of intensity of collision between convex objects and their efficient computation, in: Proc. of Int. Conf. on Intelligent Robotics, 1991.

[11] R. Weller, G. Zachmann, Inner sphere trees for proximity and penetration queries, in: Proceedings of Robotics: Science and Systems, Seattle, USA, 2009.

[12] R. Weller, G. Zachmann, Inner sphere trees for proximity and penetration queries, in: International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE), San Diego, USA, 2009.

[13] E. G. Gilbert, C. J. Ong, New distances for the separation and penetration of objects, in: ICRA, 1994, pp. 579–586.

[14] C. J. Ong, E. G. Gilbert, Growth distances: New measures for object separation and penetration, IEEE Transactions on Robotics and Automation 12 (6) (1996) 888–903.

[15] C. Y. Liu, R. W. Mayne, Distance calculations in motion planning problems with interference situations, in: ASME DETC 16th Design Autom. Conf, 1990, p. 145152.

[16] G. van den Bergen, Proximity queries and penetration depth computation on 3D game objects, in: Game Developers Conference, 2001.

[17] P. K. Agarwal, L. J. Guibas, S. Har-Peled, A. Rabinovitch, M. Sharir, Computing the penetration depth of two convex polytopes in 3d, in: Proceedings of Scandinavian Workshop on Algorithm Theory, 2000, pp. 328–338.

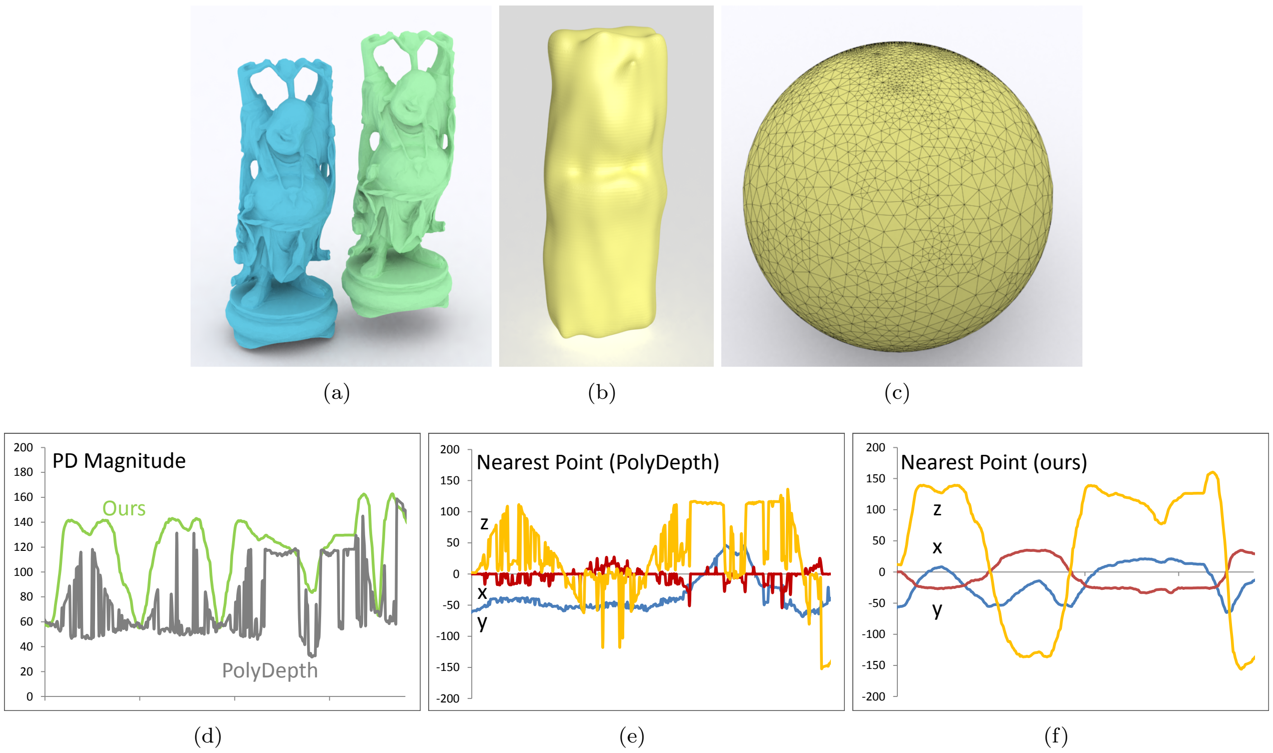


Fig. 13. (a) two Buddha models; (b) precomputed contact space approximation; (c) spherical parameterization of the approximate contact space; (d) PD magnitude: Our algorithm vs. PolyDepth, in which PD magnitudes computed by PolyDepth are not continuous; (e) nearest points computed by PolyDepth are not continuous; (f) nearest points computed by our continuous PD algorithm are continuous.

[18] Y. J. Kim, M. A. Otaduy, M. C. Lin, D. Manocha, Fast penetration depth computation for physically-based animation, in: Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation, 2002, pp. 23–31.

[19] E. Guendelman, R. Bridson, R. Fedkiw, Nonconvex rigid bodies with stacking, ACM Transactions on Graphics 22 (3) (2003) 871–878.

[20] S. Redon, M. C. Lin, A fast method for local penetration depth computation, Graphical Tools 8 (1) (2006) 63–70.

[21] J.-M. Lien, A simple method for computing minkowski sum boundary in 3d using collision detection, in: Algorithmic Foundation of Robotics VIII, Vol. 57, 2009, pp. 401–415.

[22] M. Tang, M. Lee, Y. J. Kim, Interactive hausdorff distance computation for general polygonal models, ACM Transactions on Graphics 28 (3) (2009) 74:1–74:9.

[23] M. Tang, D. Manocha, M. A. Otaduy, R. Tong, Continuous penalty forces, ACM Transactions on Graphics 31 (4) (2012) 107:1–107:9.

[24] B. Wang, F. Faure, D. K. Pai, Adaptive image-based intersection volume, in: Proceedings of SIGGRAPH, 2012, pp. 97:1–97:9.

[25] J.-M. Lien, Covering minkowski sum boundary using points with applications, Computer Aided Geometric Design 25 (8) (2008) 652–666.

[26] Anonymous, Efficient penetration depth computation using active learning, Tech. rep. (Jan. 2013).

[27] D. Attali, J.-D. Boissonnat, H. Edelsbrunner, Stability and Computation of Medial Axes: a State of the Art Report, Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration, Springer-Verlag, 2009, pp. 109–125.

[28] A. Escande, S. Miossec, A. Kheddar, Continuous gradient proximity distance for humanoids free-collision optimized-postures, in: Proc. 7th IEEE-RAS International Conference on Humanoid Robots, 2007, pp. 188–195.

[29] X. Zhang, Y. J. Kim, *k-IOS: Intersection of spheres for efficient proximity query, in: ICRA'12, 2012, pp. 354–359.*

[30] M. G. Choi, E. Ju, J.-W. Chang, J. Lee, Y. J. Kim, Linkless octree using multi-level perfect hashing, Computer Graphics Forum 28 (7) (2009) 1773–1780.

[31] J. Platt, Advances in Kernel Methods - Support Vector Learning, MIT Press, 1999, Ch. Fast Training of Support Vector Machines using Sequential Minimal Optimization, pp. 185–208.

[32] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27, software available at `http://www.csie.ntu.edu.tw/ cjlin/libsvm`.

[33] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge Universimity Press, 2000.

[34] F. Steinke, B. Scholkopf, V. Blanz, Support vector machines for 3d shape processing, Computer Graphics Forum 24 (2005) 285294.

[35] G. E. Bredon, Topology and Geometry, Springer, 1993.

[36] C. Gotsman, X. Gu, A. Sheffer, Fundamentals of spherical parameterization for 3d meshes, in: ACM SIGGRAPH, 2003, pp. 358–363.

[37] A. Sheffer, E. Praun, K. Rose, Mesh parameterization methods and their applications, Found. Trends. Comput. Graph. Vis. 2 (2) (2006) 105–171.

[38] S. Ehmann, M. Lin, Accurate and fast proximity queries between polyhedra using surface decomposition, Computer Graphics Forum 20 (3) (2001) 500–510.

[39] S. Gottschalk, M. C. Lin, D. Manocha, OBBTree: a hierarchical structure for rapid interference detection, in: Proceedings of SIGGRAPH, 1996, pp. 171–180.

[40] E. Larsen, S. Gottschalk, M. C. Lin, D. Manocha, Fast proximity queries with swept sphere volumes, in: International Conference on Robotics and Automation, 2000, pp. 3719–3726.