Robotics: Science and Systems 2007
Atlanta, GA, USA, June 27-30, 2007

A Fast and Practical Algorithm for Generalized Penetration Depth Computation

Liangjun Zhang [1] Young J. Kim [2] Dinesh Manocha [1]

[1] *Dept. of Computer Science, University of North Carolina at Chapel Hill, USA, {zlj,dm}@cs.unc.edu*

[2] *Dept. of Computer Science and Engineering, Ewha Womans University, Korea, kimy@ewha.ac.kr*

http://gamma.cs.unc.edu/PDG

***Abstract*— We present an efficient algorithm to compute the generalized penetration depth (PD^g) between rigid models. Given two overlapping objects, our algorithm attempts to compute the minimal translational and rotational motion that separates the two objects. We formulate the PD^g computation based on model-dependent distance metrics using displacement vectors. As a result, our formulation is independent of the choice of inertial and body-fixed reference frames, as well as specific representation of the configuration space. Furthermore, we show that the optimum answer lies on the boundary of the contact space and pose the computation as a constrained optimization problem. We use global approaches to find an initial guess and present efficient techniques to compute a local approximation of the contact space for iterative refinement. We highlight the performance of our algorithm on many complex models.**

I. Introduction

Penetration depth (PD) is a distance measure that quantifies the amount of interpenetration between two overlapping objects. Along with collision detection and separation distance, PD is one of the proximity queries that is useful for many applications including dynamics simulation, haptics, motion planning, and CAD/CAM. Specifically, PD is important for computing collision response [1], estimating the time of contact in dynamics simulation [2], sampling for narrow passages in retraction-based motion planing [3], [4], and C-obstacle query in motion planning [5].

There has been considerable work on PD computation, and good algorithms are known for convex polytopes. As for non-convex models, prior approaches on PD computation can be classified into local or global algorithms. The local algorithms only take into account the translational motion, i.e. *translational PD* (PD^t), and the results may be overly conservative. In many applications, including torque computation for 6-DOF haptic rendering or motion planning for articulated models, it is important to compute a penetration measure that also takes into account the rotational motion, i.e. *generalized penetration depth* (PD^g). However, the computational complexity of global PD between non-convex models is high. For PD^t, it can be computed using *Minkowski sum* formulation with the combinatorial complexity $O(n^6)$, where n is the number of features in the models [6]. For PD^g, it can be formulated by computing the arrangement of contact surfaces, and the combinatorial complexity of the arrangement is $O(n^{12})$ [7]. As a result, prior algorithms for global PD only compute an approximate solution [5], [8]. Moreover, these algorithms perform convex decomposition on non-convex models and can be rather slow for interactive applications. Overall, there are no good and practical solutions to compute the PD between non-convex models, thereby limiting their applications [4], [9], [10].

A key issue in PD^g computation is the choice of an appropriate distance metric. It is non-trivial to define a distance metric that can naturally combine the translational and rotational motion for an undergoing model, such that the resulting distance metric is *bi-invariant* with the choice of inertial and body-fixed reference frames, as well as of specific representations of the configuration space [11]. Specifically, it is well-known that for the spatial rigid body motion group SE(3), it is impossible to define a bi-invariant distance metric unless the shape of the model is known a priori [12], [13]. Finally, the distance metric should be easy to evaluate in order to devise an efficient PD^g computation algorithm.

A. Main Results

We present an efficient algorithm for computing PD^g for rigid, non-convex models. We formulate PD^g computation as a constrained optimization problem that minimizes an objective function defined by any proper distance metric that combines both translational and rotation motions, such as DISP [14] and *object norm* [15]. We use global approaches, based on motion coherence and random sampling, to compute an initial guess and incrementally walk on the *contact space* along the maximally-decreasing direction of the objective function to refine the solution. The algorithm computes a local approximation of the contact space, and we present culling techniques to accelerate the computation. As compared to the prior approaches, our algorithm offers the following benefits:

- **Generality:** Our approach is general and applicable to both convex and non-convex rigid models. The algorithm can be also extended to articulated or deformable models.
- **Practicality:** Unlike the prior approaches, our algorithm is relatively simple to implement and useful for many applications requiring both translational and rotation measures for inter-penetration.
- **Efficiency:** We use a local optimization algorithm and reduce the problem of PD^g computation to multiple collision detection and contact queries. As a result, our algorithm is efficient and can be used for interactive applications with high motion coherence.

We have implemented our PD^g algorithm and applied it to many non-convex polyhedra. In practice, our algorithm takes about a few hundred milli-seconds on models composed of a few thousand triangles.

B. Organization

The rest of our paper is organized as follows. We provide a brief survey of related work on PD^g computations in Sec. 2. In Sec. 3, we present a formulation of PD^g and give an overview of distance metrics. In Sec. 4, we provide our optimization-based algorithm to compute PD^g. We present its implementation and highlight its performance in Sec 5.

II. Previous Work

There has been considerable research work done on proximity queries including collision detection, separation distance, and PD computation [16], [17]. In this section, we briefly discuss prior approaches to PD computation and distance metrics.

A. PD Computation

Most of the work in PD computation has been restricted to PD^t, and these algorithms are based on Minkowski sums [6], [18]. A few good algorithms are known for convex polytopes [19], [20] and general polygonal models [8]. Due to the difficulty of computing a global PD^t between non-convex models, some local PD^t algorithms have been proposed [9], [10], [21].

A few authors have addressed the problem of PD^g computation. Ong's work [22], [23] can be considered as one of the earliest attempts. The optimization-based method using a quadratic objective function can be regarded as implicitly computing PD^g [24]. Ortega et al. [25] presented a method to locally minimize the *kinetic distance* between the configurations of a haptic probe and its proxy using constraint-based dynamics and continuous collision detection. Zhang et al. [5] proposed the first rigorous formulation of computing PD^g. They presented an efficient algorithm to compute PD^g for convex polytopes, and provide bounds on PD^g of non-convex polyhedra. The problem of PD^g computation is closely related to the containment problem [26]. The notion of *growth distance* has been introduced to unify separation and penetration distances [22]. Recently, Nawratil et al. [27] have also described a constrained optimization based algorithm for PD^g computation.

B. Distance Metrics in Configuration Space

The distance metric in configuration space is used to measure the distance between two configurations in the space. It is well-known that *model-independent* metrics are not bi-invariant, and thus most approaches use *model-dependent metrics* for proximity computations [11], [14], [28].

1) Distance Metrics in SE(3): The spatial rigid body displacements form a group of rigid body motion, SE(3). Throughout the rest of the paper, we will refer to a model-independent distance metric in SE(3) as a distance metric in SE(3). In theory, there is no natural choice for distance metrics in SE(3) [12], [13]. Loncaric [29] showed that there is no bi-invariant Riemannian metric in SE(3).

2) Model-dependent Distance Metrics: Using the notion of a *displacement vector* for each point in the model, the DISP distance metric is defined as the maximum length over all the displacement vectors [14], [28], [30]. The object norm, proposed by [15], is defined as an average squared length of all displacement vectors. Hofer and Pottmann [31] proposed a similar metric, but consider only a set of feature points in the model. All of these displacement vector-based metrics can be efficiently evaluated. The length of a trajectory travelled by a point on a moving model can be also used to define model-dependent metrics [5], [32]. However, it is difficult to compute the exact value of these metrics.

III. Generalized Penetration Depth and Distance Metrics

In this section, we introduce our notation and highlight issues in choosing an appropriate distance metric for defining PD^g for polyhedral models. We then show that our metrics can naturally combine translational and rotational motions, have invariance properties, and can be rapidly calculated. We also show that the optimal solution for PD^g computation with respect to each metric exists on the contact space.

A. Notation and Definitions

We first introduce some terms and notation used throughout the rest of the paper. We define the *contact space*, $\mathscr{C}_{contact}$, as a subset of the configuration space, $\mathscr{C}$, that consists of the configurations at which a robot A only touches one or more obstacles without any penetration. The union of free space $\mathscr{F}$ and contact space constitutes the valid space, $\mathscr{C}_{valid}$, of the robot, and any configuration in $\mathscr{C}_{valid}$ is a *valid* configuration. The complement of $\mathscr{F}$ in $\mathscr{C}$ is the C-obstacle space or $\mathscr{O}$.

PD^g is a measure to quantify the amount of interpenetration between two overlapping models. Given a distance metric δ in configuration space, PD^g between two polyhedral models A and B can be defined as:

$$PD^g_\delta(A,B) = \{min\{\delta(\mathbf{q_0},\mathbf{q})\} \,|\, \text{interior}(A(\mathbf{q})) \cap B = \emptyset, \mathbf{q} \in \mathscr{C}\}, \quad (1)$$

where $\mathbf{q_0}$ is the initial configuration of A, and $\mathbf{q}$ is any configuration in $\mathscr{C}$.

PD^g can be formulated as an optimization problem under non-penetration constraints (Fig. 1(a)), where the optimization objective is described by some distance metric to measure the extent of a model transformed from one configuration to another. Therefore, the computation of PD^g is directly governed by the underlying distance metric.

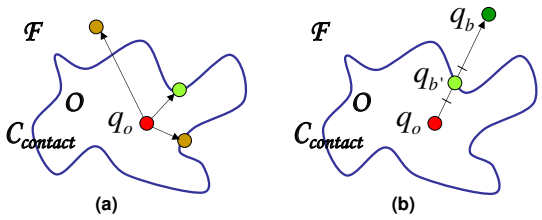

Fig. 1. PD^g **Definition and Contact Space Realization**: *(a)* PD^g *is defined as the minimal distance between the initial collision configuration* $\mathbf{q_o}$ *and any free or contact configuration, with respect to some distance metric. (b) The optimal configuration* $\mathbf{q_b}$*, which realizes* $\text{PD}^g{}_{\text{DISP}}$ *or* $\text{PD}^g{}_\sigma$*, must be on the contact space* $\mathscr{C}_{contact}$*; otherwise, one can compute another contact configuration* $\mathbf{q_b}'$*, which further reduces the objective function.* $\mathbf{q_b}'$ *is computed by applying the* **bisection** *method on the* screw motion *that interpolates* $\mathbf{q_o}$ *and* $\mathbf{q_b}$.

B. Distance Metric

We address the issue of choosing an appropriate distance metric to define PD^g. In principle, any distance metric in C-space can be used to define PD^g. We mainly use two distance metrics for rigid models, *displacement distance metric* DISP [28], [30] and *object norm* [15].

1) Displacement distance metric: Given a model A at two different configurations $\mathbf{q_a}$ and $\mathbf{q_b}$, the *displacement distance metric* is defined as the longest length of the displacement vectors of all the points on A [28], [30]:

$$\text{DISP}_A(\mathbf{q_a},\mathbf{q_b}) = \max_{\mathbf{x}\in A} ||\mathbf{x}(\mathbf{q_b}) - \mathbf{x}(\mathbf{q_a})||_2. \qquad (2)$$

2) Object norm: Also based on displacement vectors, Kazerounian and Rastegar [15] make use of an integral operator to define the *object norm*:

$$\sigma_A(\mathbf{q_a},\mathbf{q_b}) = \frac{1}{V}\int_A \rho(\mathbf{x})||\mathbf{x}(\mathbf{q_b}) - \mathbf{x}(\mathbf{q_a})||^2\, dV, \qquad (3)$$

where V and $\rho(\mathbf{x})$ are the volume and mass distribution of A, respectively.

3) Properties of DISP and σ: Both metrics can combine the translational and rotational components of SE(3) without relying on the choice of any weighting factor to define PD^g. Since both metrics are defined by using displacement vectors, they have some *invariance properties*; they are independent of the choice of inertial reference frame and body-fixed reference frame [11], and also independent of the representation of $\mathscr{C}$.

Moreover, DISP and σ metrics can be computed efficiently. In [14], we show that for a rigid model, the DISP distance is realized by a vertex on its convex hull. This leads to an efficient algorithm, C-DIST, to compute DISP. For σ, by using a quaternion representation, we can further simplify the formula originally derived by Kazerounian and Rastegar [15] into:

$$\sigma_A(\mathbf{q_a},\mathbf{q_b}) = \frac{4}{V}(I_{xx}q_1^2 + I_{yy}q_2^2 + I_{zz}q_3^2) + q_4^2 + q_5^2 + q_6^2, \qquad (4)$$

where $diag(I_{xx},I_{yy},I_{zz})$ forms a diagonal matrix computed by diagonalizing the inertia matrix I of A. (q_0,q_1,q_2,q_3) is the quaternion for the **relative** orientation of A between $\mathbf{q_a}$ and $\mathbf{q_b}$, and (q_4,q_5,q_6) is the relative translation.

C. Properties of PD^g_{DISP} and PD^g_σ

Geometrically speaking, the generalized penetration depth under DISP, $\text{PD}^g{}_{\text{DISP}}$, can be interpreted as the minimum of the maximum lengths of the displacement vectors for all the points on A, when A is placed at any collision-free or contact configuration. Also, the generalized penetration depth under σ, $\text{PD}^g{}_\sigma$, can be interpreted as the minimum cost to separate A from B, where the cost is related to the kinetic energy of A.

Due to the underlying distance metric, both $\text{PD}^g{}_{\text{DISP}}$ and $\text{PD}^g{}_\sigma$ are independent of the choice of inertial and body-fixed reference frames. In practice, these invariance properties are quite useful since one can choose any arbitrary reference frame and representation of the configuration space to compute $\text{PD}^g{}_{\text{DISP}}$ and $\text{PD}^g{}_\sigma$.

D. Contact Space Realization

For rigid models, $\text{PD}^g{}_{\text{DISP}}$ (or $\text{PD}^g{}_\sigma$) has a contact space realization property. This property implies that any valid configuration $\mathbf{q_b}$ that minimizes the objective DISP (or σ) for PD^g must lie on the contact space of A and B, or equivalently, at this configuration $\mathbf{q_b}$, A and B just touch with each other.

Theorem 1 (Contact Space Realization) *For a rigid model A placed at $\mathbf{q_o}$, and a rigid model B, if $\mathbf{q_b} \in \mathscr{C}_{valid}$ and $\text{DISP}_A(\mathbf{q_o},\mathbf{q_b}) = \text{PD}^g{}_{\text{DISP}}(A,B)$, then $\mathbf{q_b} \in \mathscr{C}_{contact}$. A similar property holds for $\text{PD}^g{}_\sigma$.*

Proof: We prove it by contradiction. Suppose the configuration $\mathbf{q_b}$ realizing $\text{PD}^g{}_{\text{DISP}}$ does not lie on the contact space $\mathscr{C}_{contact}$. Then, $\mathbf{q_b}$ must lie in the free space $\mathscr{F}$ ((Fig. 1(b)).

We use Chasles' theorem in Screw theory [33], which states that a rigid body transformation between any two configurations can be realized by rotation about an axis followed by translation parallel to that axis, where the amount of rotation is within $[0,\pi]$. The screw motion is a curve in C-space, and we denote that curve between $\mathbf{q_o}$ to $\mathbf{q_b}$ as $s(t)$, where $s(0) = \mathbf{q_o}$ and $s(1) = \mathbf{q_b}$. Since $\mathbf{q_o}$ is in $\mathscr{O}$, and $\mathbf{q_b}$ is in $\mathscr{F}$, there is at least one intersection between the curve $\{s(t)|t \in [0,1]\}$ and the contact space (Fig. 1). We denote the intersection point as $\mathbf{q_b}'$.

Based on Chasles theorem, we can compute the length of the displacement vector for any point $\mathbf{x}$ on A between $\mathbf{q_o}$ and any configuration on the screw motion $s(t)$. Furthermore, we can show that this length strictly increases with the parameter t. Therefore, for each point on A, the length of the displacement vector between $\mathbf{q_o}$ and $\mathbf{q_b}$ is less than the one between $\mathbf{q_o}$ and $\mathbf{q_b}'$. Since DISP metric uses the maximum operator for the length of the displacement vector over all points on A, we can

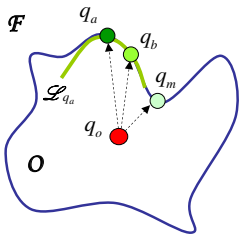

Fig. 2. **Optimization-based** PD^g **Algorithm**: *our algorithm walks on the contact space* $\mathscr{C}_{contact}$*, i.e. from* $\mathbf{q_a}$ *to* $\mathbf{q_b}$*, to find a local minimum* $\mathbf{q}_m$ *under any distance metric.*

infer that $\mathrm{DISP}_A(\mathbf{q_o},\mathbf{q_b}') < \mathrm{DISP}_A(\mathbf{q_o},\mathbf{q_b})$. This contradicts our assumption that $\mathbf{q_b}$ is the realization for $\mathrm{PD}^g{}_{\mathrm{DISP}}$.

Similarly, we can infer $\sigma_A(\mathbf{q_o},\mathbf{q_b}') < \sigma_A(\mathbf{q_o},\mathbf{q_b})$, and thus prove the property for $\mathrm{PD}^g{}_\sigma$. ■

According to Thm. 1, in order to compute PD^g, it is sufficient to search only the contact space $\mathscr{C}_{contact}$, which is one dimension lower than that of $\mathscr{C}$. Our optimization-based algorithm for PD^g uses this property.

IV. *PD^g* Computation Algorithm

In this section, we present our PD^g computation algorithm. Our algorithm can optimize any distance metric (or objective) presented in Sec. 3 by performing incremental refinement on the contact space. As Fig. 2 illustrates, our iterative optimization algorithm consists of three major steps:

1) Given an initial contact configuration $\mathbf{q_a}$, the algorithm first computes a local approximation $\mathscr{L}_{\mathbf{q_a}}$ of the contact space around $\mathbf{q_a}$.
2) The algorithm searches over the local approximation to find a new configuration $\mathbf{q_b}$ that minimizes the objective function.
3) The algorithm assigns $\mathbf{q_b}$ as a starting point for the next iteration (i.e. *walk* from $\mathbf{q_a}$ to $\mathbf{q_b}$) if $\mathbf{q_b}$ is on the contact space with smaller value of the objective function as compared to $\mathbf{q_a}$'s. Otherwise, we compute a new contact configuration $\mathbf{q_b}'$ based on $\mathbf{q_b}$.

These steps are iterated until a local minimum configuration $\mathbf{q_m}$ is found or the maximum number of iterations is reached. Next, we discuss each of these steps in more detail. Finally, we address the issue of computing an initial guess.

A. Local Contact Space Approximation

Since it is computationally prohibitive to compute a global representation of the contact space $\mathscr{C}_{contact}$, our algorithm computes a local approximation. Given a configuration $\mathbf{q_a}$, where A is in contact with B, we enumerate all contact constraints according to the pairs of contact features [28], [34]. We further decompose each contact constraint into *primitive contact constraints*, i.e. vertex/face ($v-f$), face/vertex($f-v$) or edge/edge ($e-e$). Conceptually, each primitive contact constraint represents a halfspace, and the set of all primitive constraints are used to characterize the local non-penetration

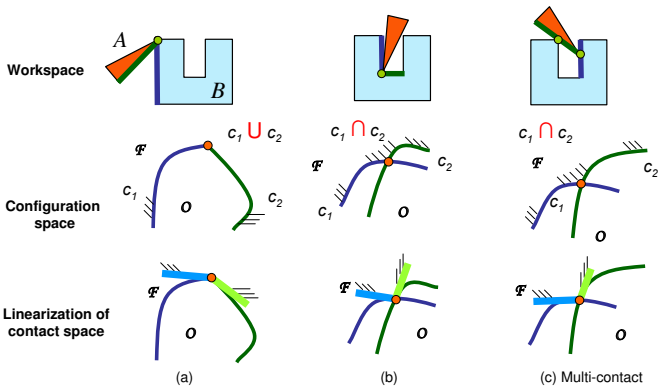

Fig. 3. **Local Contact Space Approximation**: *the local contact space is algebraically represented as a set of contact constraints concatenated with intersection or union operators (Eq. 5). Columns (a) and (b) explain how to obtain proper operators when decomposing a constraint into primitive contact constraints using 2D examples (cf. Sec IV.A). Column (c) shows a multiple contact situation. The last row illustrates the corresponding linearization for each local contact space.*

computation between A and B. Therefore, we obtain a local approximation $\mathscr{L}_{\mathbf{q_a}}$ of $\mathscr{C}_{contact}$ around the contact configuration $\mathbf{q_a}$ after concatenating all these primitive constraints $\{C_i\}$ using proper intersection or union operators $\{\circ_i\}$:

$$\mathscr{L}_{\mathbf{q_a}} = \{C_1 \circ_1 C_2 \cdots \circ_{n-1} C_n\}. \tag{5}$$

It should be noted that we do not explicitly compute a geometric representation of $\mathscr{L}_{\mathbf{q_a}}$. Instead, it is algebraically represented, and each primitive constraint is simply recorded as a pair of IDs, identifying the contact features from A and B, respectively.

When decomposing each constraint into primitive constraints, we need to choose proper Boolean operators to concatenate the resulting primitive constraints. This issue has been addressed in the area of dynamics simulation [35] and we address it in a similar manner for PD^g computation. Fig. 3 shows a 2D example with a triangle-shaped robot A touching a notch-shaped obstacle B. When decomposing a $v-v$ contact constraint into two $v-e$ constraints C_1 and C_2, if both of the contact vertices of A and B are convex (Fig. 3(a)), we use a union operator, because if either constraint C_1 or C_2 is enforced, there is no local penetration. Otherwise, if one contact vertex is non-convex (Fig. 3(b)), the intersection operation is used. For 3D models, a similar analysis is performed by identifying the convexity of edges based on their dihedral angles. In case of multiple contacts, one can first use intersection operations to concatenate all the constraints. Each individual constraint is then further decomposed into primitive constraints.

B. Searching over Local Contact Space

Given a local contact space approximation $\mathscr{L}$ of the contact configuration $\mathbf{q_a}$, we search over $\mathscr{L}$ to find $\mathbf{q_b}$ that minimizes the objective function. Since the contact space is a non-linear subspace of $\mathscr{C}$, we use two different search methods: random sampling in $\mathscr{L}$ and optimization over a first-order approximation of $\mathscr{L}$. Each of them can be performed independently.

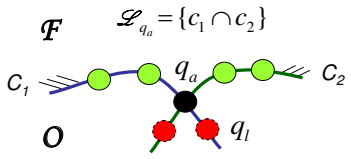

Fig. 4. **Sampling in Local Contact Space**: *$\mathscr{L}_{\mathbf{q_a}}$ is a local approximation of contact space around $\mathbf{q_a}$, represented by the intersection of its contact constraints C_1 and C_2. Our algorithm randomly generates samples on C_1 and C_2. Many potentially infeasible samples, such as $\mathbf{q}_l$, can be discarded since they are lying outside the halfspace of $\mathscr{L}_{\mathbf{q_a}}$.*

1) Sampling in Local Contact Space: Our algorithm randomly generates samples on the local contact approximation $\mathscr{L}_{\mathbf{q_a}}$ around $\mathbf{q_a}$ (Fig. 4), by placing samples on each primitive contact constraint C_i as well as on their intersections [36]. We discard any generated sample $\mathbf{q}$ if it lies outside of the halfspace formulated by $\mathscr{L}_{\mathbf{q_a}}$ by simply checking the sign of $\mathscr{L}_{\mathbf{q_a}}(\mathbf{q})$. Since $\mathscr{L}_{\mathbf{q_a}}$ is a local contact space approximation built from all contact constraints, this checking of $\mathscr{L}$ allows us to cull potentially many infeasible colliding configurations. For the rest of the configuration samples, we evaluate their distances δ to the initial configuration $\mathbf{q_o}$, and compute the minimum.

These samples are efficiently generated for each non-linear contact constraint C_i. First, we generate random values for the rotation parameters. By plugging these values into a non-linear contact constraint, we formulate a linear constraint for the additional translation parameters. Under the formulated linear constraint, random values are generated for these translation parameters.

In practice, an optimal solution for PD^g may correspond to multiple contacts, suggesting that one needs to generate more samples on the boundary formed by multiple contact constraints. As a result, we set up a system of non-linear equations for each combination of these constraints, generate random values for the rotation parameters in the system (thereby making the system linear), and sample the resulting linear system for the translation parameters.

2) Linearizing the Local Contact Space: We search for a configuration with smaller distance to the contact space by linearly approximating the contact space. For each basic contact constraint C_i, we compute its Jacobian, which is the normal of the corresponding parameterized configuration space. Using this normal, we obtain a half-plane, which is a linearization of the contact surface [21], [37]. By concatenating the half-planes using Boolean operators $\circ_i$, we generate a non-convex polyhedral cone, which serves as a local linear approximation of $\mathscr{C}_{contact}$.

3) Local Search: The sampling-based method is general for any distance metric. Moreover, we can generate samples on each non-linear contact constraint efficiently. Finally, using the local contact space approximation, our method can cull many potentially infeasible samples.

On the other hand, the method of linearizing the contact space is suitable for optimizing PD^g, if the underlying objective has a closed form. For example, for the *object norm*, we transform the coordinate in the quadratic function in Eq. (4), from an elliptic form to a circular one. Now, the problem of searching over $\mathscr{L}$ reduces to finding the closest point in the Euclidean space from $\mathbf{q_a}$ to the non-convex polyhedral cone, formulated using the linearization of $\mathscr{L}$. Since the polyhedral cone is formulated as a local approximation of $\mathscr{C}_{contact}$, it typically has a small size. Therefore, the closest point query can be performed by explicitly computing the non-convex polyhedral cone.

Algorithm 1 Optimization-based Local PD^g Algorithm
Input: two intersecting polyhedra: A - movable, B - static.
$\mathbf{q_o}$:= the initial collision configuration of A, $\mathbf{q_o} \in \mathscr{O}$.
$\mathbf{q_a}$:= a seed contact configuration of A, $\mathbf{q_a} \in \mathscr{C}_{contact}$.
Output: PD^g(A, B)

```
1:  repeat
2:    i++;
3:    L_qa := Local contact space approximation at qa;
4:    qb := argmin{δ(qo, q), q ∈ L_qa};
5:    if δ(qo, qb) == δ(qo, qa) then
6:      return δ(qo, qa);
7:    else if qb ∈ C_contact then
8:      qa := qb;
9:    else if qb ∈ F then
10:     qa := CCD_Bisection(qo, qb);
11:   else
12:     qb' := CCD(qa, qb);
13:     L_qa := L_qa ⋂ L'_qb;
14:     goto 3;
15:   end if
16: until i < MAX_ITERATION
```

C. Refinement

Although searching over the local contact space $\mathscr{L}$ around $\mathbf{q_a}$ can yield a new configuration $\mathbf{q_b}$ that improves the optimization objective of $\mathbf{q_a}$, we still need to check whether $\mathbf{q_b}$ is a valid contact configuration before advancing to it because $\mathbf{q_b}$ is computed based upon a local approximation of contact space and $\mathbf{q_b}$ may not be on the contact space.

For instance, the new configuration $\mathbf{q_b}$ may be a collision-free configuration due to the first-order approximation. To handle this case, we project $\mathbf{q_b}$ back to $\mathscr{C}_{contact}$ by computing the intersection $\mathbf{q_b}'$ between the contact space and a curve interpolating from $\mathbf{q_o}$ to $\mathbf{q_b}$ using screw motion (Fig. 1). Since $\mathbf{q_o}$ is in $\mathscr{O}$ and $\mathbf{q_b}$ is free, the intersection $\mathbf{q_b}'$ can be efficiently computed by bisection (*CCD_Bisection* in Alg. 1). Also, according to the contact space realization theorem in Sec. III.D, $\delta(\mathbf{q_o}, \mathbf{q_b}') < \delta(\mathbf{q_o}, \mathbf{q_b})$. Therefore, we are guaranteed to obtain a new configuration $\mathbf{q_b}'$, which is closer to $\mathbf{q_o}$, and thus it can be used for successive iterations.

It is also possible the new configuration $\mathbf{q_b}$ may be a colliding configuration. As Fig. 5 on the left shows, when moving from

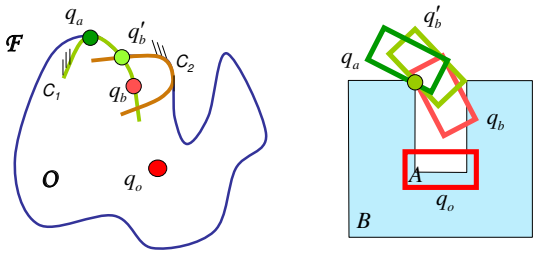

Fig. 5. **Refinement**. *Left: using the local contact space representation of* $\mathbf{q_a}$*, which includes only one constraint* C_1*, we obtain new configuration* $\mathbf{q_b}$*. Though* $\mathbf{q_b}$ *is still on* C_1*, it may not be on the contact space any more, since it will violate other constraint, such as* C_2 *here. The right figure shows a dual example happening in the workspace. When A slides on B, i.e. from* $\mathbf{q_a}$ *to* $\mathbf{q_b}$*, a collision can be created by other portions of the models. Our algorithm uses CCD to compute a correct, new contact configuration* $\mathbf{q_b}'$*.*

$\mathbf{q_a}$ to $\mathbf{q_b}$, the contact constraint C_1 is maintained. However, $\mathbf{q_b}$ is a colliding configuration as it does not satisfy the new constraint C_2. The figure on the right highlights this scenario in the workspace. When A moves from $\mathbf{q_a}$ to $\mathbf{q_b}$, the contact is still maintained. In order to handle this case, we use *continuous collision detection* (CCD) to detect the time of first collision when an object continuously moves from one configuration to another using a linearly interpolating motion in $\mathscr{C}$ [38]. In our case, when A moves from $\mathbf{q_a}$ to $\mathbf{q_b}$, we ignore the sliding contact of $\mathbf{q_a}$, and use CCD to report the first contact $\mathbf{q_b}'$ before the collision [39]. The new configuration $\mathbf{q_b}'$ can be used to update the local approximation of $\mathbf{q_a}$. This yields a more accurate contact space approximation and consequently improves the local search, e.g. culling away additional invalid samples.

D. Initial Guess

The performance of the PD^g algorithm depends on a good initial guess. For many applications, including dynamic simulation and haptic rendering, the motion coherence can be used to compute a good initial guess. Since no such motion coherence could be exploited in some other applications (e.g. sample-based motion planning), we propose a heuristic. Our method generates a set of samples on the contact space as a preprocess. At runtime, given a query configuration $\mathbf{q_o}$, our algorithm searches for the K nearest neighbors from the set of precomputed samples, and imposes the inter-distance between any pair of these K samples should be greater than some threshold. The distance metric used for nearest neighbor search is the same as the one to define PD^g. The resulting K samples serve as initial guesses for our PD^g algorithms. To generate samples on the contact space, we randomly sample the configuration space and enumerate all pairs of free and collision samples. For each pair, a contact configuration can be computed by a bisection method (Fig. 1(b)).

V. Implementation and Performance

We have implemented our PD^g algorithm using local contact space sampling for general non-convex polyhedra. In this section, we discuss some important implementation issues and highlight the performance of our algorithm on a set of complex polyhedral models. All the timeings reported here were taken on a Windows PC, with 2.8GHZ of CPU and 2GB of memory.

A. Implementation

Since our PD^g formulation is independent of the representation of the configuration space, we use a *quaternion* to represent the rotation because of its simplicity and efficiency. In our PD^g algorithm, any proximity query package supporting collision detection or contact determination can be employed. In our current implementation, we use the SWIFT++ collision detection library, because of its efficiency and it provides both these proximity queries [40]. Based on SWIFT++, our algorithm computes all the contacts between A at a contact configuration $\mathbf{q_a}$ with B. We sample the contact space locally around $\mathbf{q_a}$. For each primitive contact constraint C_i, we derive its implicit equation with respect to the parameters of a rotation component (a quaternion) and a translation component (a 3-vector). In order to sample on a constraint C_i, we first slightly perturb its rotational component by multiplying a random quaternion with a small rotational angle. The resulting rotational component is plugged back into the constraint C_i. This yields a linear constraint with only translational components, and therefore can be used to generate additional samples. To linearize C_i, we compute the Jacobian of its implicit equation for C_i . For other types of contacts, we decompose them into primitive contact constraints. Proper operators to concatenate them are identified by computing the dihedral angle of contacting edges, thereby determining whether the contact features are convex or not.

In the refinement step of the algorithm, we perform collision detection using SWIFT++ to check whether $\mathbf{q_b}$ from the local search step still lies on the contact space. When $\mathbf{q_b}$ is on contact space, our algorithm proceeds to the next iteration. Otherwise, when $\mathbf{q_b}$ is free, a new contact configuration $\mathbf{q_b}'$ is computed for the next iteration by performing recursive bisections (Fig. 1(b)) on the screw motion interpolating between $\mathbf{q_o}$ and $\mathbf{q_b}$. Finally, when $\mathbf{q_b}$ is in C-obstacle space, we compute a new contact configuration $\mathbf{q_b}'$ by using CCD. In our current implementation, we check for collision detection on a set of discrete samples on a linear motion between $\mathbf{q_a}$ and $\mathbf{q_b}$. In order to ignore the old contact during CCD query, the idea of *security distance* is used [39]. After computing a new contact configuration $\mathbf{q_b}'$ from the CCD query, our algorithm updates the local approximation around $\mathbf{q_a}$ and resumes a local search again.

B. Performance

We use different benchmarks to test the performance of our algorithm. Fig. 6(a) shows a typical setup of our experiment including two overlapping models, where A ('Pawn') is movable and B ('CAD Part') is stationary. In (b), our algorithm computes PD$^g{}_{\text{DISP}}$ or PD$^g{}_{\sigma}$ to separate the model A, initially placed at A_0, from the model B. The three images on the right highlight the intermediate configurations of A_1 and A_2 and a PD$^g{}_{\text{DISP}}$ solution A_3 with yellow color. The sequence of images (b,c,d,e) illustrates that our algorithm successfully finds

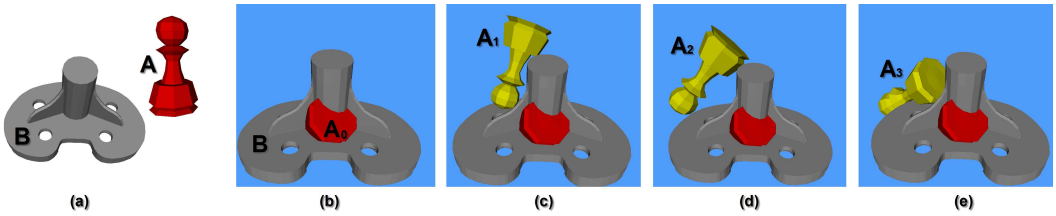

Fig. 6. **The 'CAD Part' example**: *(a) shows the models A - 'pawn' and B - 'CAD Part' used in this test. (b) illustrates a typical* PD^g *query scenario where the model A at* A_0 *overlaps with B.* A_1 *and* A_2 *are intermediate placements of A during the optimization for* $PD^g{}_{DISP}$. A_3 *is the solution for an upper bound of* $PD^g{}_{DISP}$. *The sequence of images (c,d,e) illustrates that our algorithm incrementally slides the model 'pawn' on the model 'CAD Part' to minimize* DISP *distance.*

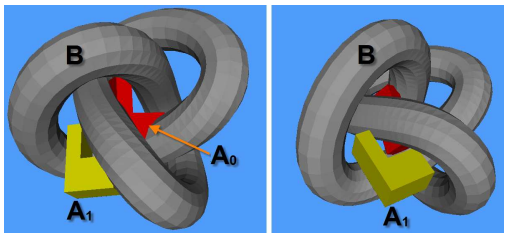

Fig. 7. **The 'torus-with-knot' example**: *the left image highlights a* PD^g *query between a model 'torus-with-knot' B intersecting with a model 'L-shaped box' at* A_0 *(red).* A_1 *is a collision-free placement of the 'L-shaped box' model as a result of* $PD^g{}_\sigma$; *the right image shows the same result but from another viewpoint.*

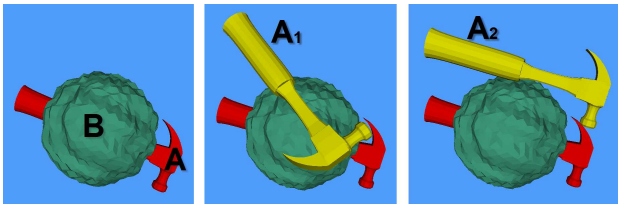

Fig. 8. **The 'hammer' example**: *from left to right:* $PD^g{}_{DISP}$ *query between A and B, an intermediate configuration* A_1, *and the solution* A_2.

an upper bound of $PD^g{}_{DISP}$ by gradually sliding the 'pawn' model on the 'CAD' model.

Figs. 7 and 8 show two more complex benchmarks that we have tested. In Fig. 7, the model 'torus-with-a-knot' has hyperbolic surfaces. This benchmark is difficult for the *containment optimization*-based algorithm [5], as that algorithm computes the convex decomposition of the complement of the model. On the other hand, our PD^g algorithm can easily handle this benchmark, and compute a tight upper bound on PD^g.

Table I summarizes the performance of our algorithm on different benchmarks. In our implementation, we set the maximum number of iterations as 30. For the most of the models we have tested, our algorithm can perform $PD^g{}_{DISP}$ query within 300*ms*, and $PD^g{}_\delta$ query with 450*ms*. Our current implementation is not optimized and the timings can be further improved.

C. Comparison and Analysis

Compared to prior method for PD^g computation in [5], our method can handle more complex non-convex models. This is because we reduce PD^g computation to proximity queries such as collision detection and contact determination. Since there are well known efficient algorithms for both these queries,

	1	2	3
A	L-Shape	Pawn	Hammer
tris #	20	304	1,692
B	Torus-with-a-Knot	CAD	Bumpy-Sphere
tris #	2,880	2,442	2,880
Avg $PD^g{}_{DISP}$(ms)	219	297	109
Avg $PD^g{}_\sigma$(ms)	156	445	138

TABLE I

Performance: *this table highlights the geometric complexity of different benchmarks we tested, as well as the performance of our algorithm.*

our method can handle complex non-convex models. Instead, the method in [5] reduces PD^g computation to *containment optimization*, which suffers from enumerating convex containers using convex decomposition, and can result in overly conservative query results for non-convex models.

Our algorithm computes an upper bound on PD^g, since the resulting configuration is guaranteed to be on the contact space. Moreover, in general, the algorithm converges to a local minimum due to the constrained optimization formulation. The termination condition for the iterative optimization is to check whether the gradient of the distance metric is proportional to that of the contact constraint after each iteration. However, some issues arise in checking this condition in practice. For example, in the case of DISP metric, one can only compute an approximation of the gradient, since no closed form is available for DISP metric. Furthermore, a convergence analysis is difficult, due to the discontinuity in contact space caused by multiple contacts.

VI. Conclusion

We present a practical algorithm to compute PD^g for non-convex polyhedral models. Using model-dependent distance metrics, we reduce the PD^g computation to a constrained optimization problem. Our algorithm performs optimization on the contact space, and the experimental results show that we can efficiently compute a tight upper bound of PD^g.

The main limitation of our approach is that our algorithm can not guarantee a global solution for PD^g computation. Its performance depends on the choice of an initial guess. For future work, it is worthwhile to analyze the convergence properties of the algorithm, as well as an error bound on the approximation. We would also like to apply our algorithm to motion planning, dynamic simulation, and haptic rendering.

Acknowledgements: We would like to thank Ming C. Lin and

Stephane Redon for providing helpful discussions. This project was supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134 and 0118743, ONR Contract N00014-01-1-0496, DARPA/RDECOM Contract N61339-04-C-0043 and Intel. Young J. Kim was supported in part by the grant 2004-205-D00168 of KRF, the STAR program of MOST and the ITRC program.

REFERENCES

[1] B. V. Mirtich, "Impulse-based dynamic simulation of rigid body systems," Ph.D. dissertation, University of California, Berkeley, 1996.

[2] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal of Numerical Methods in Engineering*, vol. 39, pp. 2673–2691, 1996.

[3] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," *Proc. of 3rd Workshop on Algorithmic Foundations of Robotics*, pp. 25–32, 1998.

[4] M. Saha, J. Latombe, Y. Chang, Lin, and F. Prinz, "Finding narrow passages with probabilistic roadmaps: the small step retraction method," *Intelligent Robots and Systems*, vol. 19, no. 3, pp. 301–319, Dec 2005.

[5] L. Zhang, Y. Kim, G. Varadhan, and D.Manocha, "Generalized penetration depth computation," in *ACM Solid and Physical Modeling Symposium (SPM06)*, 2006, pp. 173–184.

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

[7] D. Halperin, "Arrangements," in *Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O'Rourke, Eds. Boca Raton, FL: CRC Press LLC, 2004, ch. 24, pp. 529–562.

[8] Y. J. Kim, M. C. Lin, and D. Manocha, "Fast penetration depth computation using rasterization hardware and hierarchical refinement," *Proc. of Workshop on Algorithmic Foundations of Robotics*, 2002.

[9] B. Heidelberger, M. Teschner, R. Keiser, M. Mueller, and M. Gross, "Consistent penetration depth estimation for deformable collision response," in *Proceedings of Vision, Modeling, Visualization VMV'04*, November 2004, pp. 339–346.

[10] A. Sud, N. Govindaraju, R. Gayle, I. Kabul, and D. Manocha, "Fast proximity computation among deformable models using discrete voronoi diagrams," *Proc. of ACM SIGGRAPH*, pp. 1144–1153, 2006.

[11] Q. Lin and J. Burdick, "Objective and frame-invariant kinematic metric functions for rigid bodies," *The International Journal of Robotics Research*, vol. 19, no. 6, pp. 612–625, Jun 2000.

[12] J. Loncaric, "Normal forms of stiffness and compliance matrices," *IEEE Journal of Robotics and Automation*, vol. 3, no. 6, pp. 567–572, December 1987.

[13] F. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," *ASME J. Mechanical Design*, vol. 117, no. 1, pp. 48–54, March 1995.

[14] L. Zhang, Y. Kim, and D. Manocha, "C-DIST: Efficient distance computation for rigid and articulated models in configuration space," in *ACM Solid and Physical Modeling Symposium (SPM07)*, 2007, pp. 159–169.

[15] K. Kazerounian and J. Rastegar, "Object norms: A class of coordinate and metric independent norms for displacement," in *Flexible Mechanism, Dynamics and Analysis: ASME Design Technical Conference, 22nd Biennial Mechanisms Conference*, G. K. et al, Ed., vol. 47, 1992, pp. 271–275.

[16] M. Lin and D. Manocha, "Collision and proximity queries," in *Handbook of Discrete and Computational Geometry*, 2003.

[17] C. Ericson, *Real-Time Collision Detection*. Morgan Kaufmann, 2004.

[18] P. Agarwal, L. 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.

[19] G. van den Bergen, "Proximity queries and penetration depth computation on 3d game objects," *Game Developers Conference*, 2001.

[20] Y. Kim, M. Lin, and D. Manocha, "Deep: Dual-space expansion for estimating penetration depth between convex polytopes," in *Proc. IEEE International Conference on Robotics and Automation*, May 2002.

[21] S. Redon and M. Lin, "Practical local planning in the contact space," *ICRA*, 2005.

[22] C. Ong, "Penetration distances and their applications to path planning," Ph.D. dissertation, Michigan Univ., Ann Arbor., 1993.

[23] ——, "On the quantification of penetration between general objects," *International Journal of Robotics Research*, vol. 16, no. 3, pp. 400–409, 1997.

[24] V. Milenkovic and H. Schmidl, "Optimization based animation," in *ACM SIGGRAPH 2001*, 2001.

[25] M. Ortega, S. Redon, and S. Coquillart, "A six degree-of-freedom god-object method for haptic display of rigid bodies," in *IEEE Virtual Reality*, 2006.

[26] V. Milenkovic, "Rotational polygon containment and minimum enclosure using only robust 2d constructions," *Computational Geometry*, vol. 13, no. 1, pp. 3–19, 1999.

[27] G. Nawratil, H. Pottmann, and B. Ravani, "Generalized penetration depth computation based on kinematical geometry," Geometry Preprint Series, Vienna Univ. of Technology, Tech. Rep. 172, March 2007.

[28] J. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[29] J. Loncaric, "Geometrical analysis of compliant mechanisms in robotics," Ph.D. dissertation, Harvard University, 1985.

[30] S. M. LaValle, *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006.

[31] M. Hofer and H. Pottmann, "Energy-minimizing splines in manifolds," in *SIGGRAPH 2004 Conference Proceedings*, 2004, pp. 284–293.

[32] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *International Journal of Computational Geometry and Applications*, vol. 9, no. (4 & 5), pp. 495–512, 1999.

[33] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[34] J. Xiao and X. Ji, "On automatic generation of high-level contact state space," *International Journal of Robotics Research*, vol. 20, no. 7, pp. 584–606, July 2001.

[35] K. Egan, S. Berard, and J. Trinkle, "Modeling nonconvex constraints using linear complementarity," Department of Computer Science, Rensselaer Polytechnic Institute (RPI), Tech. Rep. 03-13, 2003.

[36] X. Ji and J. Xiao, "On random sampling in contact configuration space," *Proc. of Workshop on Algorithmic Foundation of Robotics*, 2000.

[37] D. Ruspini and O. Khatib, "Collision/contact models for the dynamic simulation of complex environments," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1997.

[38] X. Zhang, M. Lee, and Y. Kim, "Interactive continuous collision detection for non-convex polyhedra," in *Pacific Graphics 2006 (Visual Computer)*, 2006.

[39] S. Redon, "Fast continuous collision detection and handling for desktop virtual prototyping," *Virtual Reality*, vol. 8, no. 1, pp. 63–70, 2004.

[40] S. Ehmann and M. C. Lin, "Accurate and fast proximity queries between polyhedra using convex surface decomposition," *Computer Graphics Forum (Proc. of Eurographics'2001)*, vol. 20, no. 3, pp. 500–510, 2001.