Computer Aided Geometric Design 25 (2008) 489–502

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Efficient distance computation in configuration space

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

[a] Department of Computer Science, University of North Carolina at Chapel Hill, USA
[b] Department of Computer Science and Engineering, Ewha Womans University, South Korea

ARTICLE INFO

Article history:
Received 20 August 2007
Received in revised form 9 February 2008
Accepted 26 April 2008
Available online 3 June 2008

Keywords:
Distance metric
Configuration space

ABSTRACT

We address the problem of computing a measure of the distance between two configurations of a rigid or an articulated model. The underlying distance metric is defined as the maximum length of the displacement vectors over the vertices of the model between two configurations. Our algorithm is based on Chasles theorem from Screw theory, and we show that for a rigid model the maximum distance is realized by one of the vertices on the convex hull of the model. We use this formulation to compute the distance, and present two acceleration techniques: incremental walking on the dual space of the convex hull and culling vertices on the convex hull using a bounding volume hierarchy (BVH). Our algorithm can be easily extended to articulated models by maximizing the distance over its each link and we also present culling techniques to accelerate the computation. We highlight the performance of our algorithm on many complex models and demonstrate its applications to generalized penetration depth computation and motion planning.

1. Introduction

The notion of *configuration space* is used in many applications including motion planning (Lozano-Pérez, 1983; Latombe, 1991; Choset et al., 2005; LaValle, 2006), CAD/CAM (Joskowicz and Sacks, 1999), dynamic simulation and virtual environments (Ruspini and Khatib, 2000). For a rigid model in 3D, its configuration space $\mathcal{C}$ is defined as the set of all possible locations and orientations of the model and identified with SE(3), the group of the spatial rigid body transformations. In case of an articulated model with n free-floating bodies, the configuration space $\mathcal{C}$ defined as a Cartesian product of the configuration space $\mathcal{C}_i$ of each body in the articulated model; i.e., $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_n$. If the bodies in an articulated model form a kinematic chain, a tree or a closed loop, $\mathcal{C}$ can be formulated accordingly (LaValle, 2006)

A fundamental problem is computation of a measure of the distance between two arbitrary configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ for a model. Intuitively, the distance computed using some metric is used to quantify the extent of transformation of the model between two configurations $\mathbf{q}_0$ and $\mathbf{q}_1$.

In many applications, a key issue in distance computation is the choice of the underlying metric used to measure the distance. In sampling-based motion planning algorithms, a graph or roadmap is constructed by sampling the configuration space and connecting the nearby pairs of samples (Kavraki et al., 1996; Kuffner and LaValle, 2000; LaValle, 2006; Foskey et al., 2001). The distance metric is required to classify the pairs of samples that are close to each other. Eventually, these algorithms use local planning algorithms to compute collision-free paths between nearby samples and build a global roadmap. It has been shown that the choice of distance metric effects the connectivity of the roadmap and the performance of the overall planning algorithm (Amato et al., 2000; Kuffner, 2004; LaValle, 2006;

* Corresponding author.
E-mail addresses: zlj@cs.unc.edu (L. Zhang), kimy@ewha.ac.kr (Y.J. Kim), dm@cs.unc.edu (D. Manocha).
URL: http://gamma.cs.unc.edu/PDG/CDIST (L. Zhang).

0167-8396/$ – see front matter
doi:10.1016/j.cagd.2008.04.005

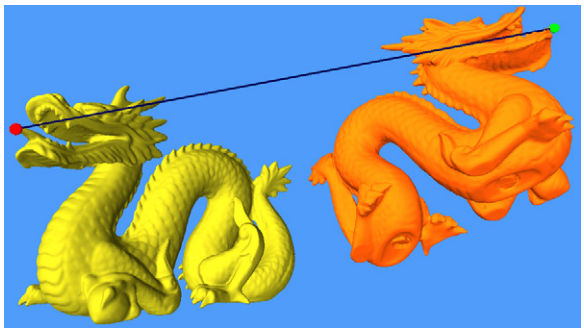

Fig. 1. C-DIST computation for a *Dragon* model: the rigid model of the dragon has 871,414 triangles. Our C-DIST algorithm performs the distance query based on DISP metric, which is defined as the maximum length of the displacement vector over every point on the model at two different configurations. Our algorithm takes about 15.2 μs, on average, to compute the vertex that has the maximum displacement length. We highlight the line segment connecting this vertex at the two configurations.

Plaku and Kavraki, 2006; Pisula et al., 2000; Geraerts, 2006). Furthermore, the distance metric is also required to define the notation *dispersion*, which is useful for evaluating the uniformness of the generated samples (Choset et al., 2005; LaValle, 2006).

The choice of distance metric also arises in measuring the extent of interpenetration between two overlapping models (Dobkin et al., 1993; Ong, 1993; Zhang et al., 2007b). The problem of computing the penetration depth frequently arises in dynamic simulation and haptic rendering. One way to characterize the extent of interpenetration between two intersecting models A and B is to search over all collision-free or contact configurations of A, and identify the closest one to the original configuration of A according to the distance metric.

Defining and calculating the distance for a rigid model undergoing translational motion is straightforward, because its configuration space is homeomorphic to the Euclidean space and we can use the Euclidean distance metric. For a model undergoing both translational and rotational motions, however, it is harder to even define a distance metric. However, the problem of defining an appropriate distance metric becomes harder for a model that undergoes both translational and rotational motion, The main issue involves coming up with a meaningful combination of translational and rotational components and formulating a metric that is *bi-invariant* with the choice of the inertial and body-fixed reference frames for the model, and independent of the underlying representation of the configuration space.

Different distance metrics have been proposed for rigid models, in particular in the area of robotics and kinematics (Latombe, 1991; Lin and Burdick, 2000; Park, 1995). At a broad level, they can be classified as: *model-independent* metrics that do not consider the shape of the model, and *model-dependent* ones that consider. Model-independent distance metrics, such as a weighted metric combining the translational and rotational components (Park, 1995; Tchon and Duleba, 1994), typically can be easily computed. However, users have to choose an appropriate weighting factor between the translational and rotational components. Furthermore, it has been shown that one cannot define a bi-invariant, model-independent distance metric (Loncaric, 1987; Park, 1995). In contrast, most of model-dependent metrics summarized in Section 2 do not need such weighting factors. These metrics are often bi-invariant and have appealing mathematical properties as well. However, for most of these metrics, no efficient algorithms are known to compute them.

In this paper, we investigate a model-dependent distance metric called DISP (see Fig. 2). This is defined as the maximum length of the displacement vector for each point $\mathbf{p}$ on a rigid or articulated model A at two configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ (Latombe, 1991; LaValle, 2006):

$$\mathrm{DISP}_A(\mathbf{q}_0, \mathbf{q}_1) = \max_{\mathbf{p}\in A} \left\| \mathbf{p}(\mathbf{q}_1) - \mathbf{p}(\mathbf{q}_0) \right\|_2. \tag{1}$$

This distance metric can meaningfully combine translational and rotational motions without relying on any weighting factor. Additionally, it is invariant with the choices of reference frames and independent of the representation of the underlying configuration space. To the best of our knowledge, no efficient algorithms are known to compute DISP for rigid and articulated models.

1.1. Main results

We present an efficient algorithm (C-DIST) to compute the DISP distance between two configurations of a rigid or articulated model. The main results include:

- We use *Chasles theorem* from Screw theory to show that the distance for a rigid model is always realized by one of the vertices of the convex hull. Based on this result, our distance computation algorithm reduces to comparing the length of the displacement vector of each vertex on the convex hull along the two configurations.
- We present two acceleration techniques to speed up our distance computation algorithm: incremental walking on the dual space of the convex hull and culling vertices on the convex hull using a bounding volume hierarchy (BVH).

- We extend our distance computation to articulated models by maximizing the DISP distance over its each link. We also present an efficient technique to accelerate the computation by using bounding boxes.
- We highlight its applications to generalized penetration depth computation and sampling-based motion planning.

We have implemented our algorithm. We highlight its performance for many complex rigid models with hundreds of thousands of triangles and articulated models. In practice, the distance computation takes tens of micro-seconds on a high-end PC.

1.2. Organization

The rest of the paper is organized as follows. In Section 2, we briefly survey related work on distance metric in configuration space. We introduce the DISP distance metrics in Section 3 and highlight its properties. In Section 4, we present our distance computation algorithm for rigid models and extend the algorithm to articulated models in Section 5. We describe its implementation in Section 6 and highlight the application of DISP distance metric in Section 7.

2. Previous work

In this section, we briefly survey previous work on distance metrics for rigid and articulated models in configuration space. Previous work can be classified into two categories: model-independent ones that do not consider the shape of the model, and model-dependent metrics that consider. Both kind of metrics are used for different applications. In *mechanism design*, for example, the operated models are not known a priori and therefore, model-independent metrics are used (Park, 1995). In motion planning and dynamics simulation, however, the shapes of the models are usually known in advance. Therefore, the shape of a model is used in defining distance metrics to provide useful mathematical and geometric properties, such as invariance.

2.1. Distance metric on SE(3)

For a 3D rigid model, its configuration space is defined as the set of all possible locations and orientations, and is identified with SE(3), the special Euclidean group in $\mathbb{R}^3$. For simplicity, we will refer to the model-independent distance metric on SE(3) as the distance metric on SE(3) throughout the rest of the paper.

There has been considerable work on distance metrics on SE(3). In theory, there is no natural choice of distance metric on SE(3). Loncaric (1985, 1987) shows that there is no bi-invariant Riemannian metric on SE(3). Park and Brockett (1994) prove that there is no differentiable bi-invariant distance metric on SE(3). Inspired by these results, Park (1995), Tchon and Duleba (1994) propose a commonly used distance metric that is em left-invariant with the choice of the inertial frame. In this case, however, users need to choose a weighting factor combining the translational and rotational components. Besides Riemannian metrics, pseudo-Riemannian metrics, such as the Klein form (Karger and Novák, 1985) are also used to define distance metrics on SE(3). Since there are no bi-invariant distance metrics on SE(3), some researchers have proposed approximate bi-invariant metrics (Larochelle and McCarthy, 1995; Etzel and McCarthy, 1996) or other left-invariant metrics (Larochelle et al., 2007).

2.2. Distance metrics in configuration space

In order to define a model-dependent distance metric, one can use the notion of a *displacement vector* for each point on the model between two configurations. The DISP distance metric is defined as the maximum length over all the displacement vectors (Latombe, 1991; LaValle, 2006; Lin and Burdick, 2000). A more general metric than DISP can be defined as the Hausdorff distance of a model between the two different configurations (Latombe, 1991). The *object norm*, proposed by Kazerounian and Rastegar (1992), is defined as an average squared length of all displacement vectors. Hofer and Pottmann (2004) use a similar definition by only considering a set of feature points of the model. Instead of the displacement vector, the trajectory traveled by any point on a moving model is used to define model-dependent distance metrics (Hsu et al., 1999; Zhang et al., 2007b); e.g., the generalized distance metric D_g in Zhang et al. (2007b). Choset et al. (2005), Kuffner (2004) suggest that the exact or approximate volume swept by a model during the transformation can be used to define a distance measure. Most of these distance metrics can be computationally expensive with the exceptions of Kazerounian and Rastegar (1992), Hofer and Pottmann (2004). In theory, all of these model-dependent distance metrics are applicable to both rigid and articulated models.

2.3. Screw theory and path interpolation

Screw motion theory has been widely used in mechanics, kinematics, and dynamics simulation (Ball, 1876; Murray et al., 1994; Kim and Rossignac, 2003; Buss, 2005). One of the major results in Screw theory is the Chasles theorem. We use this theorem in our algorithm.

An optimal interpolating curve between two input configurations, i.e., a minimum-length curve connecting them, is induced by the underlying distance metric. Such a curve is known as the *geodesic*, and for some distance metrics, it corresponds to the screw motion (Spivak, 1999; Park, 1995; Zefran et al., 1996). On the other hand, the problem of motion interpolation (Shoemake, 1985; Hofer and Pottmann, 2004; Zefran and Kumar, 1998) among multiple configurations results in border issues, since we also need to take into account the smoothness and other constraints of the overall interpolating motion.

2.4. Proximity computations

There is considerable literature on proximity computations between two or more objects in computer graphics, robotics and computational geometry (Gilbert et al., 1988; Lin and Canny, 1991; Larsen et al., 1999, 2000; Lin and Manocha, 2003; Johnson and Cohen, 2004; Sud et al., 2004). The set of proximity problems include distance computation between non-overlapping objects and penetration depth between intersecting objects. These computations often require an underlying distance metric to define proximity measure. However, most work in this area is based on the Euclidean distance metric, and few algorithms take into account both translational and rotational motions (Zhang et al., 2007b).

3. Distance metric in configuration space

In this paper, we use the distance metric, DISP (Latombe, 1991; LaValle, 2006), in configuration space. We start this section by introducing our notations used in the paper, and highlight many useful properties of DISP.

3.1. Notation

We use $\mathcal{C}$ to denote the configuration space of a rigid or an articulated model. We use bold face letters, such as a configuration $\mathbf{q}_0$, to distinguish a vector quantity from a scalar quantity. We use an upper case letter such as A to denote a rigid or articulated model. We use a lower case, bold face letter, e.g. $\mathbf{p}$, for either a point or a vertex on the model. Throughout the paper, we distinguish between a vertex that is a corner of a polyhedral model and, a point that can lie anywhere on the boundary of or inside the model. Let $A(\mathbf{q}_0)$ represents a model A at a configuration $\mathbf{q}_0$, and $\mathbf{p}(\mathbf{q}_0)$ denotes the position of a point $\mathbf{p}$ on A at the configuration $\mathbf{q}_0$.

3.2. Definition of DISP

Given a model A, the distance metric $\mathrm{DISP}_A(\mathbf{q}_0, \mathbf{q}_1)$ is defined using Eq. (1), and can be used to quantify the distance between these two arbitrary configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ in $\mathcal{C}$ (see Fig. 2). For simplicity, we often simplify the notation $\mathrm{DISP}_A(\mathbf{q}_0, \mathbf{q}_1)$ into $\mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_1)$, if the underlying model A is known in the context.

The DISP distance metric is applicable to both rigid and articulated models. Moreover, it is not required that the underlying model A be closed or watertight. Therefore, this metric can also be directly applied to objects that are represented even as a collection of triangles without any connectivity information (i.e., triangle soup models), or as a point set surface. Finally, for two models A and B, $\mathrm{DISP}_A(\mathbf{q}_0, \mathbf{q}_1) \geqslant \mathrm{DISP}_{A'}(\mathbf{q}_0, \mathbf{q}_1)$, if the model A' is a subset of the model A, i.e., $A' \subset A$.

3.3. Properties of metric space

The configuration space $\mathcal{C}$ equipped with DISP is a metric space, since it verifies the following properties:

- **Non-negativity:** $\mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_1) \geqslant 0$,
- **Reflexivity:** $\mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_1) = 0 \iff \mathbf{q}_0 = \mathbf{q}_1$,
- **Symmetry:** $\mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_1) = \mathrm{DISP}(\mathbf{q}_1, \mathbf{q}_0)$,
- **Triangle inequality:** $\mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_1) + \mathrm{DISP}(\mathbf{q}_1, \mathbf{q}_2) \geqslant \mathrm{DISP}(\mathbf{q}_0.\mathbf{q}_2)$.

The properties of non-negativity, reflexivity and symmetry follow from the definition of DISP. Next, we prove the triangle inequality property for this metric.

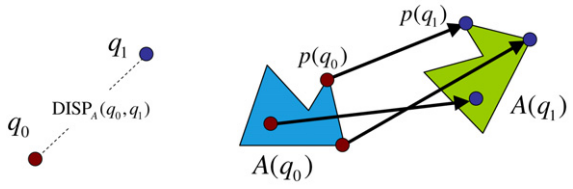

Fig. 2. The DISP distance of a model A between two configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ in configuration space is defined as the maximum length of the displacement vector between $\mathbf{p}(\mathbf{q}_0)$ and $\mathbf{p}(\mathbf{q}_1)$, where $\mathbf{p}$ is any point on A.

Proof. Suppose the point $\mathbf{p}$ on A has the maximum displacement given as $\mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_2)$. In other words, $\|\mathbf{p}(\mathbf{q}_0) - \mathbf{p}(\mathbf{q}_2)\| = \mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_2)$. In the Euclidean space, every point on A satisfies the triangle inequality, so does the point $\mathbf{p}$. Therefore, $\|\mathbf{p}(\mathbf{q}_0) - \mathbf{p}(\mathbf{q}_1)\| + \|\mathbf{p}(\mathbf{q}_1) - \mathbf{p}(\mathbf{q}_2)\| \geqslant \mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_2)$. Since $\mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_1) \geqslant \|\mathbf{p}(\mathbf{q}_0) - \mathbf{p}(\mathbf{q}_1)\|$ and $\mathrm{DISP}(\mathbf{q}_1, \mathbf{q}_2) \geqslant \|\mathbf{p}(\mathbf{q}_1) - \mathbf{p}(\mathbf{q}_2)\|$, $\mathrm{DISP}(\mathbf{q}_0, \mathbf{q}_1) + \mathrm{DISP}(\mathbf{q}_1, \mathbf{q}_2) \geqslant \mathrm{DISP}(\mathbf{q}_0.\mathbf{q}_2)$. □

As a result, the configuration space $\mathcal{C}$ equipped with DISP is a metric space, and algorithms that are based on the properties of a metric space are also applicable to $\mathcal{C}$. For example, one can use nearest neighbor search algorithms under DISP metric using the triangle inequality (Brin, 1995; Clarkson, 1999).

3.4. Invariance of DISP

We highlight a few invariance properties of the metric:

- **Invariance of reference frames:** DISP is independent of the choice of inertial reference frame and body-fixed reference frame (Lin and Burdick, 2000). Regardless of the choice of the frames, the distance defined by DISP for a model between two configurations does not change.
- **Independence of configuration space representation:** DISP is independent of the underlying representation of the configuration space. In case of a rigid model, there are many choices to represent the rotational degrees of freedom such as Euler angles, quaternions, or transformation matrices. The distance computed using DISP is independent of these representations, as well.

These invariance properties hold, since in the Euclidean space the length of the displacement vector of any point on A is invariant of the choice of reference frames and independent of the configuration space representation. In practice, these invariance properties are useful since one can choose arbitrary reference frames and representation of the configuration space to compute DISP.

4. C-DIST computation for rigid models

The DISP distance metric described in Section 3 has many elegant and useful mathematical properties. Given that no practical algorithms are known for computing DISP efficiently, its applications have been limited. In this section, we present a novel formulation of this metric and an efficient algorithm, C-DIST, to compute the distance for rigid models. We first show that the DISP distance of a rigid polyhedral model is equal to the maximum length of the displacement vectors over the vertices on its convex hull (CH). Following this formulation, a straightforward algorithm is to maximize the displacement vectors over all the vertices on the CH, which has a linear complexity in the size of the CH. Finally, we present two different techniques to accelerate the distance computation: incremental walking on the dual space of the CH, and culling vertices on the CH using a bounding volume hierarchy (BVH) structure. In practice, the culling technique can improve the performance by an order of magnitude.

4.1. Convexity in DISP computation

We first present a convex realization theorem for the DISP distance for a rigid model:

Theorem 1 *(Convex realization). Given a rigid polyhedral model A, the* DISP *distance of A between two arbitrary configurations is equal to the maximum length of the displacement vectors over all the vertices on the convex hull of A.*

Proof. We use the Chasles theorem from screw theory as shown in Fig. 3(a) and (b) (Ball, 1876; Murray et al., 1994). It states that a rigid body transformation from a configuration $\mathbf{q}_0$ to a configuration $\mathbf{q}_1$ can be realized by rotation about an axis followed by translation parallel to that axis. Such an axis is called a *screw axis*. As Fig. 3(c) shows, when a model first rotates around the screw axis ω by θ, and translates along ω by d, a point $\mathbf{p}$ on the model will be displaced to $\mathbf{p}_1$, then to $\mathbf{p}'$.

We compute the length of the displacement vector $\overrightarrow{\mathbf{pp}'}$. Let us represent the distance from the point $\mathbf{p}$ to the axis ω as r. Given that the vector $\overrightarrow{\mathbf{pp}_1}$ is orthogonal to $\overrightarrow{\mathbf{p}_1\mathbf{p}'}$, the squared length of the displacement vector $\overrightarrow{\mathbf{pp}'}$ is given as:

$$\|\overrightarrow{\mathbf{pp}'}\|^2 = \|\overrightarrow{\mathbf{pp}_1}\|^2 + \|\overrightarrow{\mathbf{p}_1\mathbf{p}'}\|^2 = 4r^2\sin^2(\theta/2) + d^2 = 2(1-\cos\theta)r^2 + d^2. \tag{2}$$

In Eq. (2), θ and d are independent of the model A and are solely governed by the input configurations $\mathbf{q}_0$ and $\mathbf{q}_1$. However, the distance r for every point on A to the screw axis ω varies. Since $(1-\cos(\theta)) \geqslant 0$ for any θ, a larger value of r implies a larger value of the length of the displacement vector. If we denote the maximum distance from every point on A to the screw axis ω as $\eta(A,\omega)$, DISP can be written as:

$$\mathrm{DISP}_A(\mathbf{q}_0, \mathbf{q}_1) = \sqrt{2(1-\cos\theta)\eta^2(A,\omega) + d^2}. \tag{3}$$

According to Eq. (3), proving Theorem 1 is equivalent to proving the following lemma:

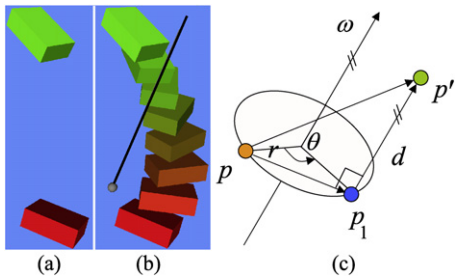

Fig. 3. Chasles theorem in screw theory: (a) shows a rigid body transformation of a rectangular bar. (b) The Chasles theorem states that any rigid transformation can be realized by rotation about an axis followed by translation parallel to that axis. Such axis is called a *screw axis*. (c) According to the Chasles theorem, when a model is transformed, the length of the displacement vector for any point $\mathbf{p}$ on the model can be calculated by first considering the rotation about the screw axis ω (from $\mathbf{p}$ to $\mathbf{p}_1$), then the translation along ω (from $\mathbf{p}_1$ to $\mathbf{p}'$).

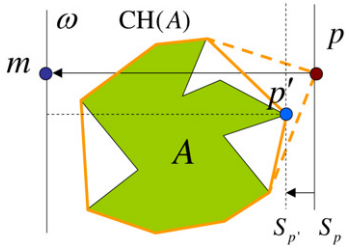

Fig. 4. $\eta(A, \omega)$, the maximum distance from points on a polyhedral model A to a line ω is equal to the maximum distance from the vertices on its convex hull to ω.

Lemma 1. *The maximum distance from points on a polyhedral model A to a line ω is equal to the maximum distance from the vertices on the convex hull of A to ω.*

We prove Lemma 1 by contradiction. Throughout the rest of the proof, we distinguish a vertex which comprises a corner of the polyhedral model from a point which can be designated anywhere on the model.

It can be easily shown that Lemma 1 holds when A is convex. If A is non-convex, we first find a **vertex p**, which is on the convex hull of A or $\mathrm{CH}(A)$, and is farthest to the line ω. Denote the projection of the vertex $\mathbf{p}$ on ω is $\mathbf{m}$. We construct a plane $S_{\mathbf{p}}$ which passes through $\mathbf{p}$ and is orthogonal to $\overrightarrow{\mathbf{pm}}$. Since $\mathbf{p}$ is the farthest point on $\mathrm{CH}(A)$ to ω and $A \subseteq \mathrm{CH}(A)$, A and ω must be located on the same half-space of S_p (see Fig. 4).

Assume that Lemma 1 does not hold for non-convex A. This means that $\eta(A, \omega) \neq \eta(\mathrm{CH}(A), \omega)$. Since $A \subseteq \mathrm{CH}(A)$, we have $\eta(A, \omega) < \eta(\mathrm{CH}(A), \omega)$. As a result, A does not intersect with $S_{\mathbf{p}}$ (Fig. 4). Next, we translate the plane S_p along the direction of $\overrightarrow{\mathbf{pm}}$ until it touches a point $\mathbf{p}'$ on A. Denote S'_p as the resulting plane which is passing through $\mathbf{p}'$ and parallel to S_p. Because A lies entirely on one side of the S'_p, we get a different convex hull for A. This contradicts the fact that the convex hull of an object is unique. As a result, Lemma 1 holds. □

Similarly to the proof for Theorem 1, we can also prove the following proposition for general smooth models.

Proposition 1. $\mathrm{DISP}_A(\mathbf{q}_0, \mathbf{q}_1)$ *distance of a rigid smooth model A between two configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ is equal to the maximum length of the displacement vectors over all the boundary points of* $\mathrm{CH}(A)$; $\mathrm{DISP}_A(\mathbf{q}_0, \mathbf{q}_1)$ *can be calculated using Eq.* (3)*, where r is the maximum distance from the boundary points on* $\mathrm{CH}(A)$ *to the screw axis.*

4.2. C-DIST computation algorithm

Two simple algorithms to compute $\mathrm{DISP}_A(\mathbf{q}_0, \mathbf{q}_1)$ for a polyhedral model A follow directly from Theorem 1 and Eq. (3).

Maximization of displacement vectors. According to Theorem 1, one can compute the convex hull $\mathrm{CH}(A)$ of A and find the maximum length of displacement vectors for all the vertices on $\mathrm{CH}(A)$. In many applications, we can compute the convex hull of a rigid model as a preprocessing step. At runtime, DISP can be efficiently computed by only considering the vertices on the convex hull. If the size of the convex hull is small, it is plausible to consider all the vertices on the convex hull, compare the length of all displacement vectors and compute their maximum.

Maximization of distance to screw axis. One can also use Eq. (3) to compute the DISP. In this equation, θ and d are determined by the motion between two configurations $\mathbf{q}_0$ to $\mathbf{q}_1$. $\eta(A, \omega)$, which depends on the underlying model A and screw axis can be computed by visiting all the vertices of $\mathrm{CH}(A)$ and computing the maximum distance to the screw axis ω.

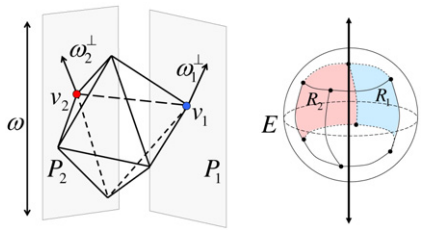

Fig. 5. Walking algorithm: Given a screw axis ω, $\eta(A,\omega)$ can be computed by visiting all the vertices on the convex hull whose support plane can have a normal $\omega^\perp$ orthogonal to ω. (Left) An initial vertex $v_k = v_1$ for the walking algorithm is located since v_1 can have a supporting plane P_1 whose normal is $\omega_1^\perp$ orthogonal to ω. The next search vertex $v_{k+1} := v_2$ is found since v_2 is adjacent to v_1 and has a supporting plane P_2 whose normal is $\omega_2^\perp$ orthogonal to ω. This search process is iterated until v_{k+1} becomes equal to v_1 again. (Right) The same process can be explained based on the Gauss map of the given convex hull. Enumerating all the vertices whose supporting plane can have a normal orthogonal to ω is equivalent to finding all the regions (including R_1, R_2 that are mapped from v_1, v_2, respectively) on the Gauss map that intersect with the equator E when ω is mapped to the pole of the Gauss map.

Each of the two methods described above has a linear complexity in the size of the convex hull, and is efficient for moderately complex models. Furthermore, these two methods are robust and simple to implement. Finally, these methods do not impose any topological requirements on the model. Even for a model represented as a triangle soup, i.e., without connectivity information or as a point cloud, DISP can be computed easily.

In practice, however, the number of vertices on the convex hull can be excessive. Therefore, we present techniques to accelerate our algorithm by reducing the number of accesses to the vertices on the convex hull. In the following Sections 4.3 and 4.4, we present two acceleration techniques: incremental walking on the dual space of CH, and culling vertices on CH by using a bounding volume hierarchy (BVH) structure.

4.3. Accelerating C-DIST computation by incremental walking

In order to reduce the number of accesses to the vertices on the convex hull, we present an optimization-based algorithm that performs feature walking on the dual space of the convex hull. We use the properties of Gauss map to compute the extremal vertices of convex hull, which are orthogonal to the screw axis.

Given Eq. (3), it follows that DISP is realized by one of the vertices on the convex hull $\mathrm{CH}(A)$, which has the maximum distance to the screw axis ω. We use this property to compute DISP in two steps:

1. Enumerate all the vertices v_i on $\mathrm{CH}(A)$ supported by a plane whose normal is orthogonal to ω.
2. Find a vertex in v_i that corresponds to $\eta(A,\omega)$.

The main computational task in the above algorithm lies in the first step. A relatively straightforward way to implement this step is (Fig. 5):

1. Choose any direction $\omega^\perp$ orthogonal to ω. Find a vertex v_1 whose supporting plane has a normal parallel to $\omega^\perp$ and set v_1 as the current search vertex $v_k := v_1$. Computing v_1 is known as support mapping of $\omega^\perp$ or extremal vertex query along $\omega^\perp$, and it can be computed in logarithmic time in the number of vertices of the convex hull (de Berg et al., 1997). In practice, the support mapping can be efficiently implemented using a lookup table.
2. Walk to the neighboring vertices v_{k+1} of v_k, if v_{k+1} has a supporting plane with a normal orthogonal to ω.
3. Repeat the above two steps, until v_{k+1} becomes equal to v_1.

Alternatively, we can compute all v_i's based on the Gauss map of $\mathrm{CH}(A)$. The mapping is defined from the feature space of an object to the surface of a unit sphere $\mathbb{S}^2$ as: a vertex is mapped to a region, a face to a point and an edge to a great arc (Spivak, 1999). The task of enumerating all v_i's boils down to finding the intersecting regions on the Gauss map with its equator when the north or south pole of the Gauss map corresponds to the direction of ω. The computational complexity of this algorithm is governed by two factors: finding an initial vertex v_1 and the walking step itself. Finding v_1 can have a logarithmic complexity in terms of the number of vertices of the convex hull and the walking step has a linear complexity in the number of vertices that are traversed.

In practice, computing the intersections between Gauss map regions and the equator can be performed by centrally projecting both the equator and the Gauss map to a plane and finding the intersections of convex polygons (projected Gauss regions) and a line (projected equator).

4.4. Accelerating C-DIST computation using a bounding volume hierarchy (BVH)

In practice, the vertices of the convex hull of the models are not distributed uniformly in 3D space. As a result, the walking scheme highlighted above can result in robustness problems, especially when accessing those vertices that are very close with each other. In this section, we present a different acceleration technique for C-DIST computation. Our method

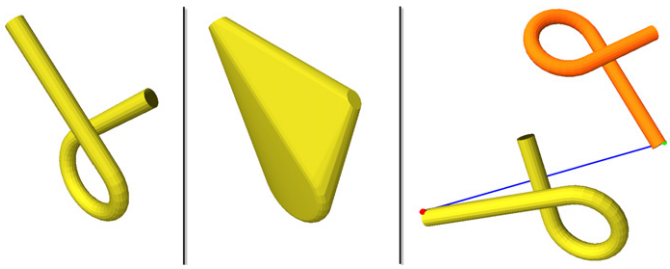

Fig. 6. C-DIST computation: The leftmost figure shows the Alpha model with 1008 triangles, and the middles its convex hull with 311 vertices. The rightmost figure shows a scenario when the model is placed at two arbitrary configurations. The vertex on this model, which realizes the longest displacement vector is found by our C-DIST algorithm and highlighted. The line segment connecting this vertex at the two configurations is also highlighted. Using SSV-tree, our C-DIST algorithm can perform distance query for this model within 5.6 μs.

uses a bounding volume hierarchy (BVH) tree structure to cluster the vertices on a convex hull. The BVH enables us to efficiently compute $\eta(A, \omega)$, the maximum distance between from every point on a model A to an axis ω. The acceleration is achieved because a node in the BVH as well as its descendant nodes can be culled if the distance η from this node to the axis is less than the global maximum distance.

In practice, we use the BVH of swept sphere volumes (SSV) (Larsen et al., 1999). SSV includes three different types of bounding volumes (BVs): point swept sphere (*PSS*), line swept sphere (*LSS*), and rectangle swept sphere (*RSS*). *PSS*, *LSS*, and *RSS* are created by sweeping a sphere along a point, a line and a rectangle in three-dimensional space, respectively.

SSV-tree construction for a point set. Let us denote S as a set of vertices on the convex hull of the given model A. As a preprocessing step, our algorithm recursively builds an SSV-tree for the point set S from top to bottom. We first compute its swept sphere volume, $SSV(S)$, as the root node of the SSV-tree. To decide whether to further subdivide a node N into two children nodes, we measure the density ρ defined as the number of vertices inside a node over its volume. If ρ is larger than some given threshold, we terminate the subdivision. Otherwise, we partition the point set Q inside N into two subsets Q_1 and Q_2 in a way to maximize the sum of densities for $SSV(Q_1)$ and $SSV(Q_2)$. For the purpose of maximization, we sweep a partitioning plane along the longest dimension of the node N and evaluate $\rho(Q_1) + \rho(Q_2)$ of two resulting point subsets, Q_1, Q_2. We choose a partitioning plane that maximizes this sum.

SSV-tree traversal and SSV-axis distance query. We use the SSV-tree structure to efficiently query the maximum distance from a point set S to the axis ω. By initializing the global maximum distance as $-\infty$ and starting from the root node, our algorithm traverses its associated SSV-tree in the depth-first order. During the traversal, we compute the maximum distance from the visited *SSV* node to the axis ω. Depending on the type of the underlying *SSV*, we need to compute the maximum distance between $1, 2$ or 4 corner spheres of the *SSV* and the axis. If this distance is not greater than the global maximum distance, we need not check the node as well as its descendant nodes any more, and can cull them. Otherwise, the depth-first order traversal continues. During the traversal, if a leaf *SSV* node is reached, we check whether the distance from any point contained in the *SSV* node to the axis is larger than the global maximum distance; if yes, we update the global maximum distance.

5. C-DIST computation for articulated models

The C-DIST computation algorithm for rigid models can be extended to articulated models, whose links can form serial or parallel chains, tree structure, or closed loops. In order to compute the DISP distance for an articulated model between two arbitrary configurations, we consider each of its links as a separate rigid body; the maximum DISP distance over all articulated links is the DISP distance for this articulated model.

This algorithm can be improved by conservatively estimating the DISP distance for each link using a bounding volume and comparing it with the DISP for other links that have been already calculated. Specifically, for a link L in an articulated model A, we precompute its bounding volume (e.g., an *oriented bounding box* of link L, $OBB(L)$). If DISP for all the links that have been already computed is greater than DISP for $OBB(L)$, we need not compute the exact DISP for link L and can cull it away.

In case of an articulated robot forming a serial chain, a link farther from the base of the robot typically undergoes a larger displacement as compared to the ones that are nearer to the base. Therefore, a simple heuristic to accelerate DISP computation for such an articulated robot is to first compute DISP for links that are farther from the base.

6. Implementation and performance

We have implemented our C-DIST computation algorithm and applied it to various rigid and articulated models. In this section, we highlight its performance on these complex benchmarks. If it is not explicitly specified, all the timings reported in the paper were taken on a 2.8 GHz Pentium IV PC with 2 GB of memory.

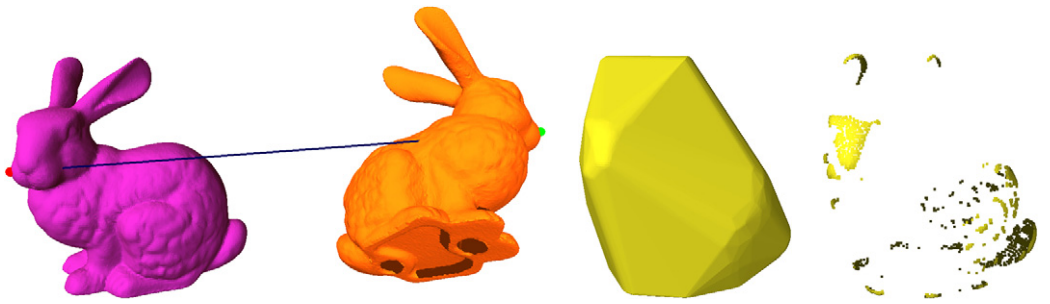

Fig. 7. C-DIST computation for *Bunny* model: This model has 69,451 triangles. (a) shows a scenario of the DISP distance query between two different configurations of the bunny model. Our C-DIST algorithm can perform the query within 8.0 μs on average. (b) shows the convex hull of this model. (c) shows the vertices on this convex hull, which are not uniformly distributed in 3D space. This can result in the robustness problems for the C-DIST algorithm based on incremental walking.

Table 1
Performance of C-DIST for rigid models: $\#V$, $\#V$ of CH denote the number of the vertices on each input model and the number of vertices on its convex hull, respectively. t_{pre} is the time for the preprocessing step, including the computation of convex hull (t_{pre_qhull}) and building the BVH structure (t_{pre_ssv}). t_{bf}, t_{ch}, t_{op} are the average DISP query time, based on three different methods. t_{bf} is the running time of the brute-force method that checks all the vertices of the input model. t_{ch} is the running time of our C-DIST method, which checks all the vertices on the convex hull. t_{op} is the running time of our accelerated C-DIST method that uses a SSV-tree

	Alpha	Cup	Bunny	Hand	Dragon
Triangle	1008	4226	69,451	86,361	871,414
#V	3024	3000	208,353	259,803	2,614,242
#V of CH	311	1019	1504	1836	2448
t_{pre} (s)	0.016	0.002	0.584	0.661	9.517
t_{pre_qhull} (s)	0.016	0.000	0.581	0.651	9.508
t_{pre_ssv} (s)	0.000	0.002	0.003	0.010	0.009
t_{bf} (μs)	184.8	231.1	13,633	16,730	169,062
t_{ch} (μs)	19.1	64.1	85.3	105.0	153.0
t_{op} (μs)	5.6	10.2	8.0	18.2	15.2
Speedup (t_{ch}/t_{op})	3.4	6.3	10.7	5.76	10.1

6.1. C-DIST implementation for rigid models

In our implementation, we precompute the convex hull for an input rigid model using *QHull*.[1] We further build a BVH structure—*swept sphere volumes* (SSV) tree for the vertices on the convex hull. For BVH-based DISP computation, we use the SSV-tree to compute the maximum distance from the model to the screw axis.

We test our C-DIST implementation on a set of rigid polyhedral models, including triangular mesh models, such as the *Cup* model, and triangle soup models without connectivity information, such as *Alpha*, *Bunny*, *Hand* and *Dragon*. The model complexity and the performance is summarized in Table 1.

Fig. 6 shows the C-DIST computation for *Alpha* model. In this test, we place this model at two arbitrary configurations and use our C-DIST algorithm to compute the DISP distance. We highlight the line segment that connects the vertex with the largest displacement at the two configurations. For this model, the brute force method of visiting all the vertices on the model takes 184.8 μs on average, for each DISP query. Our C-DIST algorithm that compares all the vertices on the convex hull can perform the query in 19.1 μs. Using SSV-tree, our C-DIST computation can perform the query in 5.6 μs.

Figs. 1, 7 and 10 highlight the application of C-DIST algorithm to complex models consisting of tens of thousands of triangles. According to Table 1, our C-DIST computation based on SSV-tree can perform each distance query within 20 μs for these models. We achieve up to 10 times speedup over an algorithm that computes the displacement for each vertex of the convex hull.

6.2. C-DIST implementation for articulated models

Our C-DIST implementation for articulated models is built on top of C-DIST computation for rigid models. We construct an OBB for each link of the model, and use them for culling.

Fig. 11 highlights a 9-DOF articulated model—*Puma* manipulator with 6 joints as well as 3 degree of freedoms of its base. Our C-DIST algorithm can perform the distance query in 27.91 μs on average. Fig. 12 shows a complex articulated model

[1] http://www.qhull.org/.

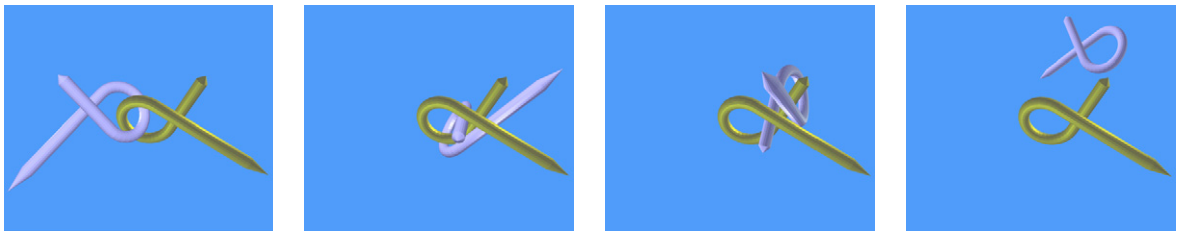

Fig. 8. Application to motion planning: Our C-DIST computation algorithm is integrated into a variant of RRT planner to solve this well-known challenging benchmark—the Alpha Puzzle. The sequence of the images demonstrate a collision-free path computed in order to separate the two interlocked alpha-shaped models. One benefit using DISP metric and C-DIST algorithm is that the users need not to choose a weighting factor between translational and rotational components.

Table 2
Performance of C-DIST for articulated models: Using *OBB* culling, our C-DIST can perform the DISP query for these two examples within 30 μs

	DOF	Triangle	#V of CH	t_{ch} (μs)	t_{obb} (μs)	Speedup
Puma	9	868	296	42.30	27.91	1.5
IRB-2400	6	3791	531	73.89	21.90	3.4

with 6 DOFs—*IRB2400 with an arcgun*. Our algorithm can perform DISP distance query in 21.90 μs. Table 2 summarizes the performance of C-DIST for these two articulated models.

6.3. Comparison

As compared to other distance metrics that are used for configuration space distance computation, such as any weighted metric combing the translational and rotational components, DISP has many elegant mathematical properties. For example, the users need not choose any weighting factor between the translational and rotational components. DISP metric shares many properties with another model-dependent metric—object norm, which is easier to compute than DISP metric. However, each of these metrics is suitable for different applications. Since object norm is defined as an average squared length of all displacement vectors of a model, it could characterize the variation of energy when a model is transformed from one configuration to another configuration. In contrast, the geometric property of DISP metric implied by the maximum operation in its definition makes DISP more useful for proximity queries and path planning. Since DISP has appealing properties and can be efficiently computed by our C-DIST algorithm, we investigate these applications in the next section.

7. Applications

In this section, we describe the applications of C-DIST algorithm to generalized penetration depth computation and sampling-based motion planning.

7.1. Generalized penetration depth computation

In dynamics simulation, haptic rendering and virtual environment, it is often necessary to quantify the amount of inter-penetration between two intersecting models. One such measures is *penetration depth*. A formulation for *generalized penetration depth* PD^g, which takes into account both translational and rotational motions to separate two inter-penetrated models, has been recently proposed (Ong, 1993; Zhang et al., 2007b). The formulation of PD^g includes two steps: defining a distance metric for a model in configuration space, which takes into account both translational and rotational motions, and minimizing this distance metric under non-penetration constraints. A metric based on trajectory length is used in Zhang et al. (2007b). However, it is difficult to compute this distance metric exactly, and only the computation for upper and lower bounds is known. On the contrast, the DISP distance metric can be efficiently computed by using our C-DIST algorithm. Along with its other mathematical properties, DISP metric is more suitable to define a generalized penetration depth PD^g_{DISP} for the moving model A and fixed model B, i.e.:

$$PD^g_{DISP}(A,B) = \min\left(\left\{DISP_A(\mathbf{q}_0,\mathbf{q}) \;\middle|\; \text{interior}\big(A(\mathbf{q})\big) \cap B = \emptyset\right\}\right), \tag{4}$$

where $\mathbf{q}_0$ is the initial configuration of A, and $\mathbf{q}$ is any configuration in configuration space.

PD^g_{DISP} is independent of the choice of inertial and body-fixed reference frames, as well as specific representation of the configuration space, due to the mathematical properties of the underlying DISP metric. Similarly to PD^g under D_g metric, one can show that for two convex models A and B, $PD^g_{DISP}(A,B)$ is equal to the translational penetration depth, $PD^t(A,B)$, that is defined as minimum translational distance to separate intersecting objects (Ong, 1993; Zhang et al., 2007b). Therefore, we can compute a lower bound on $PD^g_{DISP}(A,B)$ by decomposing A and B into convex pieces and taking the maximum PD^t over each pair of convex pieces from each model (Zhang et al., 2007b).

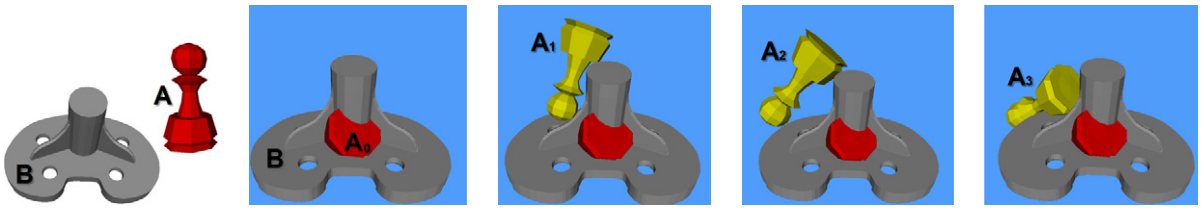

Fig. 9. PD^g_{DISP} query: (a) shows the models A—Pawn and B—CAD Part used in this test. (b) illustrates a typical PD^g_{DISP} query scenario where the model A_0 (A at configuration $\mathbf{q}_0$) overlaps with B. The goal is to compute their generalized penetration depth. (c, d, e) A_1 and A_2 are intermediate placements of A computed by the optimization step. 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 the distance to A_0.

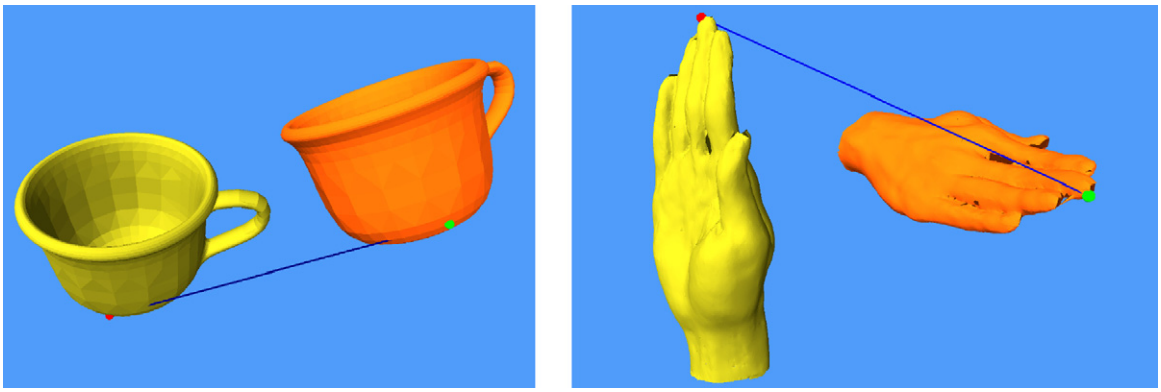

Fig. 10. C-DIST computation: Our algorithm can handle any polyhedral model represented as a triangular mesh, a triangle soup, or a point-set model. The *Cup* model on the left is represented as a triangular mesh, while the *Hand* model on the right is a scanned model and represented as a triangle soup with 86,361 triangles. Our algorithm can perform the distance query for these models within 10.2 μs and 18.2 μs, respectively.

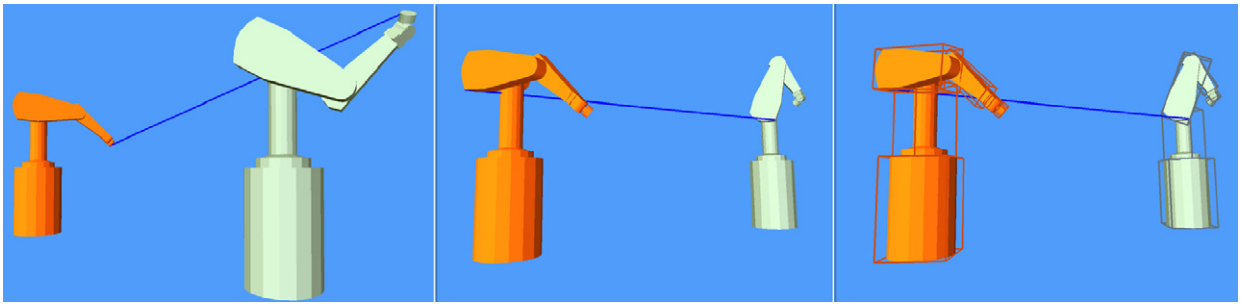

Fig. 11. C-DIST computation for an articulated model: *Puma*. Left: the *Puma* model with 9 DOF include 6 articulated joints and 3 DOF of its base. Middle: the result of the distance query by our C-DIST algorithm is highlighted. Right: the OBB associated with each link. For this model, our algorithm can perform the query within 27.91 μs, on average.

Table 3
Performance of PD^g_{DISP} query. Our optimization-based algorithm performs the PD^g_{DISP} query within 297 ms on average

Pawn	CAD Part	Avg PD^g_{DISP}
304 triangles	2442 triangles	297 ms

In Zhang et al. (2007a), we present a simple and practical approach to compute an upper bound on PD^g_{DISP} between two arbitrary non-convex models. We pose the problem as constrained optimization, and present efficient techniques to iteratively refine the solution. In Fig. 9, we highlight the results of our optimization-based PD^g_{DISP} approach for non-convex models. The performance of this approach, which is summarized in Table 3, benefits from our efficient C-DIST algorithm when evaluating the DISP distance.

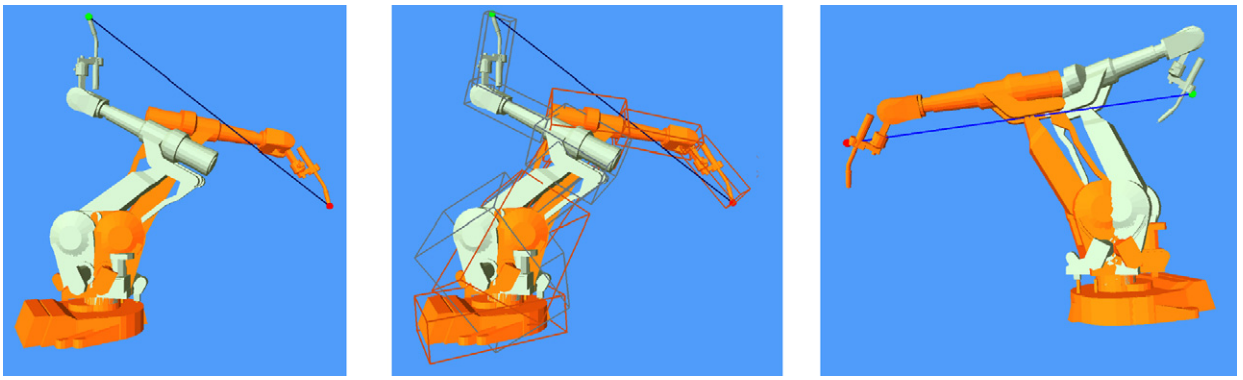

Fig. 12. C-DIST computation for an articulated model: *IRB2400 with an arcgun*. This model has 6 joints and 3791 triangles. Our algorithm can perform the query within 27.91 μs on average. In the middle figure, we highlight the OBB associated with each link which can be used for efficient culling.

7.2. Sampling-based motion planning

The C-DIST algorithm can be used for sampling-based motion planning approaches (Choset et al., 2005; LaValle, 2006). In these approaches, a roadmap is constructed by sampling the configuration space and connecting nearby pairs of samples. The C-DIST algorithm is used to evaluate the distance between samples for K nearest neighbors computation, which is one of integral parts of the algorithm. Fig. 8 shows a motion planning problem for a rigid robot in a 3D environment. Our C-DIST computation algorithm is integrated into a variant of RRT planner, which can solve this challenging planning problem with 4130.5 s (Zhang and Manocha, 2008). One benefit using DISP metric and C-DIST algorithm is that users need not choose any weighting factor between the translational and rotational components, while choosing meaningful factors for other distance metrics used in the literature usually is ad hoc in practice.

8. Conclusions and future work

We present a novel and efficient algorithm (C-DIST) to compute the DISP distance of a rigid or articulated model between two arbitrary configurations. We show that for a rigid model, the distance computation reduces to computing the maximum length of the displacement vectors for the vertices on the convex hull of the model. Our formulation is equivalent to computing the maximum distance between the model and the screw axis that is implied by the rigid transformation between the two given configurations. Furthermore, we present two techniques to accelerate the distance computation: incremental walking on the dual space of the convex hull and culling the vertices on the convex hull using the BVH of SSVs. We also extend our algorithm to articulated models. The experimental results show that our C-DIST computation can compute the DISP distance for complex rigid and articulated models in tens of micro-seconds on average. Finally, we highlight applications of our C-DIST algorithm to generalized penetration depth computation and motion planning.

Limitations and future work Our approach has a few limitations. The C-DIST optimization algorithm based on incremental walking is susceptible to degenerate configurations and needs special handling. Furthermore, our algorithm only computes the distance between the two configurations and not the actual path that realizes the DISP distance. This results in the following open problem:

Problem 1 *(Geodesic computation for DISP distance metric).* Given two configurations $\mathbf{q}_0$ and $\mathbf{q}_1$, find an interpolating curve $\mathbf{c}(t), t \in [0,1]$, such that $\mathbf{c}(t) = \mathbf{q}_0$ and $\mathbf{c}(t) = \mathbf{q}_1$ and minimizes $\gamma(\mathbf{q}_0, \mathbf{q}_1)$, where:

$$\gamma(\mathbf{q}_0, \mathbf{q}_1) = \lim_{n\to\infty} \sum_{i=0}^{i=n-1} DISP\big(c(i/n), c\big((i+1)/n\big)\big). \tag{5}$$

One can easily show a counter example to show that the path realized by the screw motion between $\mathbf{q}_0$ and $\mathbf{q}_1$ is not the geodesic induced by the DISP distance metric.

There are many other avenues for future work. We are interesting in quantitatively evaluating the improvement when applying our C-DIST algorithm to sampling-based motion planning. We are also interested in investigating other mathematical properties of DISP metric, such as the smoothness of the distance field induced by DISP metric. Finally, we would like to formulate and compute other useful metrics for measuring in configuration space, such as volume.

Acknowledgements

We would like to thank Frank Chongwoo Park, Steven M. LaValle, Jean-Claude Latombe and Bert Jütler for useful discussions. We would also like to thank the anonymous reviewers for their valuable comments. This research 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 Korea Research Foundation Grant funded by the Korean Government (KRF-2007-331-D00400) and the IT R&D program of MKE/IITA (2008-F-033-01, Development of Real-time Physics Simulation Engine for e-Entertainment).

References

Amato, N., Bayazit, O., Dale, L., Jones, C., Vallejo, D., 2000. Choosing good distance metrics and local planners for probabilistic roadmap methods. IEEE Trans. Robot. Automat. 16 (4), 442–447.

Ball, R., 1876. The Theory of Screws. Hodges and Foster, Dublin.

Brin, S., 1995. Near neighbor search in large metric spaces. In: International Conference on Very Large Data Bases, pp. 574–584.

Buss, S., 2005. Collision detection with relative screw motion. The Visual Computer 21, 41–58.

Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S., 2005. Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press.

Clarkson, K.L., 1999. Nearest neighbor queries in metric spaces. In: Discrete and Computational Geometry, pp. 63–93.

de Berg, M., van Kreveld, M., Overmars, M.H., Schwarzkopf, O., 1997. Computational Geometry: Algorithms and Applications. Springer-Verlag, Berlin.

Dobkin, D., Hershberger, J., Kirkpatrick, D., Suri, S., 1993. Computing the intersection-depth of polyhedra. Algorithmica 9, 518–533.

Etzel, K., McCarthy, J., 1996. A metric for spatial displacement using biquaternions on so(4). In: Proc. IEEE International Conference on Robotics and Automation, vol. 4, pp. 3185–3190.

Foskey, M., Garber, M., Lin, M., Manocha, D., 2001. A Voronoi-based hybrid planner. In: Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems.

Geraerts, R.J., 2006. Sample-based motion planning: Analysis and path quality. PhD thesis, Utrecht University.

Gilbert, E.G., Johnson, D.W., Keerthi, S.S., 1988. A fast procedure for computing the distance between objects in three-dimensional space. IEEE J. Robot. Automat. RA-4, 193–203.

Hofer, M., Pottmann, H., 2004. Energy-minimizing splines in manifolds. In: SIGGRAPH 2004 Conference Proceedings, pp. 284–293.

Hsu, D., Latombe, J.-C., Motwani, R., 1999. Path planning in expansive configuration spaces. Internat. J. Comput. Geom. Appl. 9 (4–5), 495–512.

Johnson, D.E., Cohen, E., 2004. Unified distance queries in a heterogeneous model environment. In: ASME DETC.

Joskowicz, L., Sacks, E., 1999. Computer-aided mechanical design using configuration spaces. Comput. Sci. Engrg. 1 (6), 14–21.

Karger, A., Novák, J., 1985. Space Kinematics and Lie Groups. Gordon and Breach Science Publishers.

Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Automat. 12 (4), 566–580.

Kazerounian, K., Rastegar, J., 1992. Object norms: A class of coordinate and metric independent norms for displacement. In: G.K. et al. (Eds.), Flexible Mechanism, Dynamics and Analysis: ASME Design Technical Conference, 22nd Biennial Mechanisms Conference, vol. 47, pp. 271–275.

Kim, B., Rossignac, J., 2003. Collision prediction for polyhedra under screw motion. In: Symposium on Solid Modeling and Applications.

Kuffner, J., LaValle, S., 2000. RRT-connect: An efficient approach to single-query path planning. In: Proc. IEEE International Conference on Robotics and Automation.

Kuffner, J., 2004. Effective sampling and distance metrics for 3d rigid body path planning. In: IEEE Int. Conf. on Robotics and Automation.

Larochelle, P., McCarthy, J., 1995. Planar motion synthesis using an approximate bi-invariant metric. ASME J. Mech. Des. 117 (4), 646–651.

Larochelle, P., Murray, A., Angeles, J., 2007. A distance metric for finite sets of rigid-body displacements via the polar decomposition. ASME J. Mech. Des. 129 (8), 883–886.

Larsen, E., Gottschalk, S., Lin, M., Manocha, D., 1999. Fast proximity queries with swept sphere volumes. Tech. Rep. TR99-018, Department of Computer Science, University of North Carolina.

Larsen, E., Gottschalk, S., Lin, M., Manocha, D., 2000. Distance queries with rectangular swept sphere volumes. In: Proc. of IEEE Int. Conference on Robotics and Automation, pp. 3719–3726.

Latombe, J., 1991. Robot Motion Planning. Kluwer Academic Publishers.

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

Lin, Q., Burdick, J., 2000. Objective and frame-invariant kinematic metric functions for rigid bodies. Internat. J. Robot. Res. 19 (6), 612–625.

Lin, M., Canny, J.F., 1991. Efficient algorithms for incremental distance computation. In: IEEE Conference on Robotics and Automation, pp. 1008–1014.

Lin, M., Manocha, D., 2003. Collision and proximity queries. In: Handbook of Discrete and Computational Geometry.

Loncaric, J., 1985. Geometrical analysis of compliant mechanisms in robotics. PhD thesis, Harvard University.

Loncaric, J., 1987. Normal forms of stiffness and compliance matrices. IEEE J. Robot. Automat. 3 (6), 567–572.

Lozano-Pérez, T., 1983. Spatial planning: A configuration space approach. IEEE Trans. Comput. C-32, 108–120.

Murray, R.M., Li, Z., Sastry, S.S., 1994. A Mathematical Introduction to Robotic Manipulation. CRC Press.

Ong, C., 1993. Penetration distances and their applications to path planning. PhD thesis, Michigan Univ., Ann Arbor.

Park, F., Brockett, R., 1994. Kinematic dexterity of robotic mechanisms. Int. J. Robotics Res. 13 (1), 1–15.

Park, F., 1995. Distance metrics on the rigid-body motions with applications to mechanism design. ASME J. Mech. Des. 117 (1), 48–54.

Pisula, C., Hoff, K., Lin, M., Manocha, D., 2000. Randomized path planning for a rigid body based on hardware accelerated Voronoi sampling. In: Proc. of 4th International Workshop on Algorithmic Foundations of Robotics.

Plaku, E., Kavraki, L.E., 2006. Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In: Workshop on Algorithmic Foundations of Robotics (WAFR).

Ruspini, D., Khatib, O., 2000. A framework for multi-contact multi-body dynamic simulation and haptic display. In: Proc. IEEE International Conference on Robotics and Automation.

Shoemake, K., 1985. Animating rotation with quaternion curves. In: Barsky, B.A. (Ed.), Computer Graphics (SIGGRAPH '85 Proceedings), vol. 19, pp. 245–254.

Spivak, M., 1999. A Comprehensive Introduction to Differential Geometry, third ed. Publish or Perish Press.

Sud, A., Otaduy, M.A., Manocha, D., 2004. DiFi: Fast 3D distance field computation using graphics hardware. Computer Graphics Forum (Proc. Eurographics) 23 (3), 557–566.

Tchon, K., Duleba, I., 1994. Definition of a kinematic metric for robot manipulators. J. Robotic Syst. 11 (3), 211–221.

Zefran, M., Kumar, V., 1998. Interpolation schemes for rigid body motions. Comp. Aided Des. 30 (3), 179–189.
Zefran, M., Kumar, V., Croke, C., 1996. Choice of Riemannian metrics for rigid body kinematics. In: ASME 24th Biennial Mechanisms Conference.
Zhang, L., Manocha, D., 2008. A retraction-based RRT planner. In: IEEE International Conference on Robotics and Automation (ICRA).
Zhang, L., Kim, Y., Manocha, D., 2007a. A fast and practical algorithm for generalized penetration depth computation. In: Proceedings of Robotics: Science and Systems.
Zhang, L., Kim, Y., Varadhan, G., Manocha, D., 2007b. Generalized penetration depth computation. Comp. Aided Des. 39 (8), 625–638.