C-DIST: Efficient Distance Computation for Rigid and Articulated Models in Configuration Space

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/CDIST

Abstract

The problem of distance computation arises in many applications including motion planning, CAD/CAM, dynamic simulation and virtual environments. Most prior work in this area has been restricted to separation or penetration distance computation between two objects. In this paper, we address the problem of computing a measure of distance between two configurations of a rigid or articulated model. The underlying distance metric is defined as the length of the longest displacement vector over the corresponding vertices of the model between two configurations. Our algorithm is based on Chasles theorem in Screw theory, and we show that the maximum distance can be realized only by a vertex of the convex hull of a rigid object. We use this formulation to compute the distance, and present two acceleration techniques to speed up the computation: 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 describe its application to proximity queries and motion planning.

Keywords: Distance metric, Configuration space

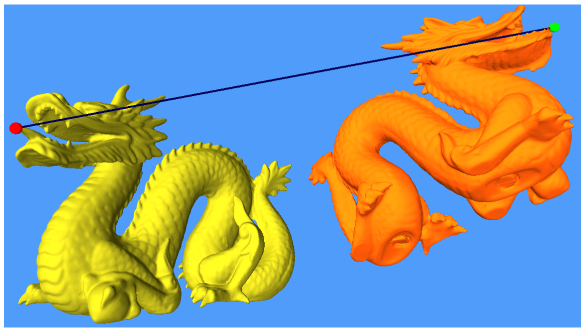

Figure 1: C-DIST *Computation for a* Dragon *model: This rigid model has* 871,414 *triangles. Our* C-DIST *algorithm can perform the* DISP *distance query, which is defined as the maximum length of the displacement vector over every point on the model at two different configurations (shown with the line). Our algorithm takes about* $15.2\mu s$*, on average, to perform this query.*

1 Introduction

The configuration space for a rigid or articulated model is defined as the space of all possible placements of the given model. The dimension of the configuration space corresponds to the number of degrees of freedom of the model. For example, the placement of a rigid model is determined by its orientation and position. Given a planar rigid model, its configuration space is $\mathbb{R}^2 \times \mathbb{S}^1$ and homeomorphic to the special Euclidean group of SE(2); for a spatial model, its configuration space corresponds to $\mathbb{R}^3 \times \mathbb{RP}^3$ and is homeomorphic to SE(3), where $\mathbb{RP}^n$ denotes n-dimensional real projective space. In case of an articulated model with n free-floating bodies in $\mathbb{R}^2$ or $\mathbb{R}^3$, its configuration space $\mathscr{C}$ is a Cartesian product of the configuration space $\mathscr{C}_i$ of each body in the articulated model; i.e., $\mathscr{C} = \mathscr{C}_1 \times \mathscr{C}_2 \times \cdots \times \mathscr{C}_n$. If the bodies in an articulated model form a kinematic chain or a tree, or have kinematic constraints, $\mathscr{C}$ can be formulated accordingly [LaValle 2006].

The notion of configuration space is used in many applications including motion planning [Choset et al. 2005; Latombe 1991; LaValle 2006], CAD/CAM, dynamic simulation and virtual environments [Mirtich 2000]. A fundamental problem in these applications is computation of a measure of distance between two different configurations of the model. In these applications, the measure of distance under an appropriate *distance metric* is used to quantify how far a model is displaced from one configuration to another.

A key issue is the choice of the metric or the distance measure, which affects the performance of the overall algorithm. 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]. A distance metric is used to check which nearby sample pairs need to be connected. Eventually, these algorithms check for a collision-free path between the nearby sample pairs and build a global roadmap. The choice of distance metric affects the connectivity of the roadmap and the performance of the overall planning algorithm [Amato et al. 2000; Kuffner 2004; LaValle 2006; Plaku and Kavraki 2006]. The distance metric in configuration space is also used to evaluate the sample generation technique used in the planning algorithm. For example, the statistical notion of *dispersion*, which can be used to evaluate the uniformness of samples, relies on a distance metric [Choset et al. 2005; LaValle 2006].

A key computation in dynamic simulation and haptic rendering is a measure to quantify the extent of interpenetration between two overlapping models [Dobkin et al. 1993; Ong 1993; Zhang et al. 2006a]. 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 model undergoing translational motion is relatively easy, because the well-defined Euclidean distance metric can be employed in this case. However, it is harder to even define a distance metric for a model undergoing both translational and rotational motion. The main challenge is to naturally combine the translational and rotational components and formulate a metric that is *bi-invariant* with the choice of the inertial and body-fixed reference frames for the model, and that is independent of the representation of the configuration space. For the spatial rigid body motion group SE(3), it has been shown that one cannot define a bi-invariant distance metric when the shape of the body is not considered [Loncaric 1987; Park 1995].

Different distance metrics have been proposed for rigid and articulated models, in particular in the area of robotics and kinematics [Latombe 1991; Lin and Burdick 2000; Park 1995]. In general, there are two different classes of distance metrics in configuration space: *model-independent* and *model-dependent* metrics according to whether the shape of the model is considered or not. Model-independent distance metrics, such as L_p norms, can be easily computed, but they are not bi-invariant. Some of these metrics are *left-invariant*, since it is invariant with the choice of the inertial frame [Park 1995; Tchon and Duleba 1994]. However, in these cases, users have to choose a weighting factor between the rotational and translational components. In contrast, model-dependent metrics are usually invariant with the choice of both reference frames, and have appealing mathematical properties as well. However, for most of these metrics, no efficient algorithms are known for distance computation.

In this paper, we investigate a model-dependent distance metric, DISP, of a model when it is placed at two different configurations. DISP is defined as the longest length of the displacement vector for each point on the model A at two configurations $\mathbf{q_0}$ and $\mathbf{q_1}$ [Latombe 1991; LaValle 2006] (Fig. 2):

$$\text{DISP}_A(\mathbf{q_0},\mathbf{q_1}) = \max_{\mathbf{p}\in A} ||\mathbf{p}(\mathbf{q_1}) - \mathbf{p}(\mathbf{q_0})||_2. \tag{1}$$

This distance metric can naturally combine translational and rotational motions without relying on any weighting factor. Additionally, it is invariant with the choices of reference frames, and is independent of the representation of the underlying configuration space. However, to the best of our knowledge, no efficient algorithms are known to compute DISP for complex 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. Our novel results include:

- We use *Chasles theorem* from Screw theory to show that the distance is realized by one of the vertices lying on the convex hull of the rigid model, and the maximum length of displacement vectors of such vertices is DISP. 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 application to proximity computation and motion planning. These include continuous collision detection and penetration depth computation.

We have implemented our algorithm. We highlight its performance for many complex rigid models with up to $871K$ triangles, and articulated models. In practice, distance computation takes tens of micro-seconds on complex models 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 metric 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 discuss some applications of DISP distance metric in Section 6 and describe its implementation 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. We differentiate a class of model-independent distance metrics from model-dependent ones based on whether or not the shape of a model is considered in defining the metrics. Both classes have their own applications. For example, in mechanical design, since manipulated models are not known a priori, model-independent metrics are used [Park 1995]. In motion planning and dynamics simulation, however, models under consideration are usually known in advance. Therefore, the shape of a model can be utilized in defining distance metrics to provide useful mathematical and geometric properties.

2.1 Distance Metric on SE(3)

The spatial rigid body displacement forms a group of rigid body motion SE(3). Throughout the rest of the paper, we will refer to a model-independent distance metric on SE(3) as a distance metric on SE(3).

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; Loncaric 1987] show 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 which is left-invariant with the choice of the inertial frame, although in this case, users need to choose a weighting factor between translational and rotational components. Besides Riemannian metrics, pseudo-Riemannian metrics, such as the Klein form [Karger and Novk 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 approximated, bi-invariant metrics [Larochelle and McCarthy 1995; Etzel and McCarthy 1996].

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 a model when the model is placed at two arbitrary 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 at 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 metric by only considering the displacement vectors for the feature points of a model. Instead of the displacement vector, the trajectory traveled by any point on a moving model was used to define model-dependent distance metrics [Hsu et al. 1999; Zhang et al. 2006a], e.g., a generalized distance metric D_g in [Zhang et al. 2006a]. [Choset et al. 2005; Xavier 1997] suggest using the exact or approximated volume swept by a model as a distance measure. With the exception of object norms, whose closed form is known, these distance metrics are computationally expensive. In theory, all of these model-dependent distance metrics are general for 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 that states: any rigid transformation can be realized by rotation around an axis with some angle combined with translation parallel to that axis.

An interpolation between two input configurations, i.e., a minimum-length curve connecting them, is usually implied by the distance metric with which the configuration space is associated. Such a curve is known as *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] between a set of configurations is more general since we also need to take into account smoothness of the overall interpolating curve.

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; Larsen et al. 2000; Lin and Manocha 2003; Johnson and Cohen 2004]. The set of proximity problems include distance computation between separating objects and penetration depth between intersecting objects. These computations often require an underlying distance metric to define a proximity measure. However, most work in this area is based on the Euclidean distance metric, although a few algorithms take into account the rotation motion along with the translation motion [Zhang et al. 2006a].

3 Distance Metric in Configuration Space

In this paper, we use the distance metric, DISP [Latombe 1991; LaValle 2006]. To the best of our knowledge, no efficient algorithms are known to compute DISP distance for general rigid and articulated models. In this section, we introduce our notation and highlight many useful properties of this metric.

3.1 Notation

We use $\mathscr{C}$ to denote the configuration space of a rigid or 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 a point or a vertex. Throughout the paper, we distinguish a vertex which is a corner on a polyhedral model from a point which can be designated anywhere on the model. $A(\mathbf{q})$ represents the placement of a model A at a configuration $\mathbf{q}$, and $\mathbf{p}(\mathbf{q})$ denotes the position of a point $\mathbf{p}$ on A at the configuration $\mathbf{q}$.

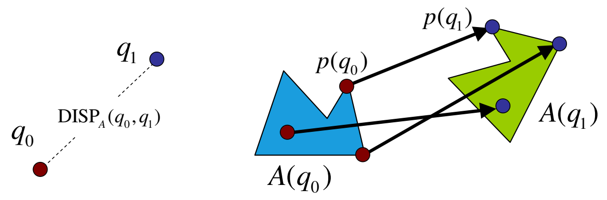

Figure 2: *The* DISP *distance at a model A between two configurations $\mathbf{q_0}$ and $\mathbf{q_1}$ in configuration space is defined as the longest displacement vector between* $\mathbf{p}(\mathbf{q_0})$ *and* $\mathbf{p}(\mathbf{q_1})$*, where* $\mathbf{p}$ *is any point on A.*

3.2 Definitions

Given a model A, the distance metric $\mathrm{DISP}_A(\mathbf{q_0},\mathbf{q_1})$ is defined by Eq. (1), and can be used to quantify the distance between these two arbitrary configurations $\mathbf{q_0}$ and $\mathbf{q_1}$ in the configuration $\mathscr{C}$ (see Fig. 2). When the underlying model A is known in the context, we simplify the notation $\mathrm{DISP}_A(\mathbf{q_0},\mathbf{q_1})$ into $\mathrm{DISP}(\mathbf{q_0},\mathbf{q_1})$.

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 with no connectivity information (i.e., triangle soup models), or as a point set surface. Finally, for two models A and B with the same initial and final configurations $\mathbf{q_0},\mathbf{q_1}$, $\mathrm{DISP}_A(\mathbf{q_0},\mathbf{q_1}) \geq \mathrm{DISP}_B(\mathbf{q_0},\mathbf{q_1})$, if the model B is a subset of the model A, i.e., $B \subset A$.

3.3 Properties of Metric Space

The configuration space $\mathscr{C}$ associated with DISP is a metric space, since it satisfies the following properties:

- **Non-negativity:** $\mathrm{DISP}(\mathbf{q_0},\mathbf{q_1}) \geq 0$,
- **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:** $\mathrm{DISP}(\mathbf{q_0},\mathbf{q_1}) + \mathrm{DISP}(\mathbf{q_1},\mathbf{q_2}) \geq \mathrm{DISP}(\mathbf{q_0}.\mathbf{q_2})$.

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

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, $||\mathbf{p}(\mathbf{q_0}) - \mathbf{p}(\mathbf{q_1})|| + ||\mathbf{p}(\mathbf{q_1}) - \mathbf{p}(\mathbf{q_2})|| \geq \mathrm{DISP}(\mathbf{q_0},\mathbf{q_2})$. Since $\mathrm{DISP}(\mathbf{q_0},\mathbf{q_1}) \geq ||\mathbf{p}(\mathbf{q_0}) - \mathbf{p}(\mathbf{q_1})||$ and $\mathrm{DISP}(\mathbf{q_1},\mathbf{q_2}) \geq ||\mathbf{p}(\mathbf{q_1}) - \mathbf{p}(\mathbf{q_2})||$, $\mathrm{DISP}(\mathbf{q_0},\mathbf{q_1}) + \mathrm{DISP}(\mathbf{q_1},\mathbf{q_2}) \geq \mathrm{DISP}(\mathbf{q_0}.\mathbf{q_2})$. $\square$

As a result, the configuration space $\mathscr{C}$ associated with DISP is a metric space, and algorithms that are based on properties of a metric space are also applicable to $\mathscr{C}$. For example, one can use nearest neighbor search algorithms, which are based on 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 or placements does not change.
- **Independence of configuration space representation:** DISP is independent of the 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.

These invariance properties hold, since in the Euclidean space as the length of the displacement vector of any point on A is invariant with 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 at two arbitrary configurations is equal to the maximum length of the displacement vectors over all the vertices on the convex hull of A.*

Proof The Chasles theorem in Screw theory [Ball 1876; Murray et al. 1994] 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 axis is called a *screw axis*. As Fig. 3 shows, when a model first rotates around the screw axis ω by θ, and translates along ω by d, any 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:

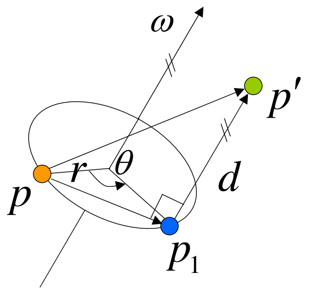

Figure 3: *Screw motion. For any point* $\mathbf{p}$ *on a model, its screw motion can be decomposed into rotation about a screw axis* ω *(from* $\mathbf{p}$ *to* $\mathbf{p}_1$*), followed by translation along* ω *(from* $\mathbf{p}_1$ *to* $\mathbf{p}'$*).*

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

In Eq. (2), θ and d are independent of the model A and are governed by the input rigid transformation between $\mathbf{q_0}$ and $\mathbf{q_1}$. The distance r from every point on A to the screw axis ω is different. Since $(1-cos(\theta)) \geq 0$ 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:

$$\text{DISP}_A(\mathbf{q_0},\mathbf{q_1}) = \sqrt{2(1-cos\theta)\eta^2(A,\omega)+d^2}. \quad (3)$$

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

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*.*

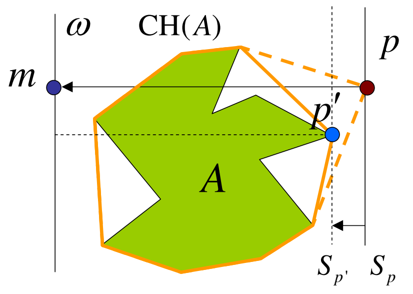

Figure 4: $\eta(A,\omega)$*, the maximum distance from points on a polyhedral model A to a line* ω *is equal to the maximum distance from the vertices on its convex hull to* ω*.*

We prove Lem.1 by using a contradiction argument. 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 Lem.1 holds when A is convex. If A is non-convex, we first find a **vertex** $\mathbf{p}$, which is on the convex hull of A or CH(A), and is farthest to the line ω. 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 CH(A) to ω and $A \subseteq$ CH(A), A and ω must be located on the same half-space of S_p (see Fig. 4)

Assume that Lem.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 hits 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, Lem. 1 holds. □

Similarly to the proof for Thm. 1, we can also prove the following proposition for general smooth models under rigid transformation.

Proposition 1 DISP$_A(\mathbf{q_0},\mathbf{q_1})$ *distance of a rigid smooth model A at 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* CH(A)*;* 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* CH(A) *to the screw axis.*

4.2 C-DIST Computation Algorithm

Two simple algorithms to compute DISP$_A(\mathbf{q_0},\mathbf{q_1})$ for a polyhedral model A follow directly from Thm. 1 and Eq. (3).

Maximization of Displacement Vectors. According to Thm. 1, one can compute the convex hull CH(A) of A and find the maximum length of displacement vectors for all the vertices on 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 underlying screw motion from $\mathbf{q_0}$ 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 CH(A) and computing the maximum distance to the screw axis ω.

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, since they rely on only inner and cross products on vectors. Finally, these methods do not impose any topological requirements on the model. Even for a model represented as a triangle soup, i.e., with no connectivity information or as 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 sub-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.

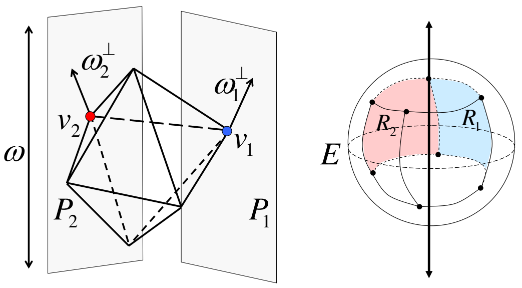

Figure 5: *Walking Algorithm. Given a screw axis* ω*,* $\eta(A,\omega)$ *can be computed by visiting all the vertices on the convex hull whose support plane can have a normal* $\omega^\perp$ *orthogonal to* ω*. (Left) an initial vertex* $v_k = v_1$ *for the walking algorithm is located since* v_1 *can 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.*

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 find 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 $CH(A)$, which has the largest value, $\eta(A,\omega)$, with respect to the screw axis ω. We use this property to compute DISP in two steps:

1. Enumerate all the vertices v_i on $CH(A)$ 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 $CH(A)$. 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 during the *walking* and their incident faces.

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 on the convex hull. In this section, we present a different acceleration technique for C-DIST computation. Our method uses a bounding volume hierarchy (BVH) tree structure to cluster the vertices on a convex hull. The BVH enables us to efficiently compute the distance $\eta(A,\omega)$ from a model A to an axis ω, 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 a 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 of N, which is 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 recursively. 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 be represented as serial or parallel chains, tree structure, or closed loops. In order to compute the DISP distance for an articulated model at 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 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 box (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 Applications

In this section, we describe a few applications of C-DIST algorithm. These include sampling-based motion planning, continuous collision detection and generalized penetration depth computation.

6.1 Sampling-Based Motion Planning

C-DIST can be used to improve the performance of sampling-based motion planning algorithms [Choset et al. 2005; LaValle 2006]. In these algorithms, a roadmap is constructed by sampling the configuration space and connecting nearby pairs of samples. The C-DIST algorithm is used to compute the K nearest neighbors for each sample.

Most prior algorithms compute the nearby samples in configuration space by using L_p, weighted L_p or other scale-dependent metrics [LaValle 2006]. As compared to these distance metrics, the DISP distance metric computed by C-DIST algorithm offers several benefits. First, the users need not choose any weighting factor between translational and rotational components. Second, when the robot moves in the Euclidean space, the points with the largest displacement are more likely to result in collisions with the obstacles. C-DIST algorithm computes the maximum displacement over all points on the robot transformed from one configuration to another configuration. Therefore, choosing K nearest neighbors according to DISP metric results in neighboring samples that have higher probabilities in terms of finding a collision-free path as compared to pairs selected by other metrics. As a result, one can construct a roadmap that can better capture the connectivity of the free space. Finally, using our C-DIST computation algorithm, we can efficiently and robustly compute DISP for complex models within a few micro-seconds. This is important as sampling-based planners typically generate a high number of samples and spend a high

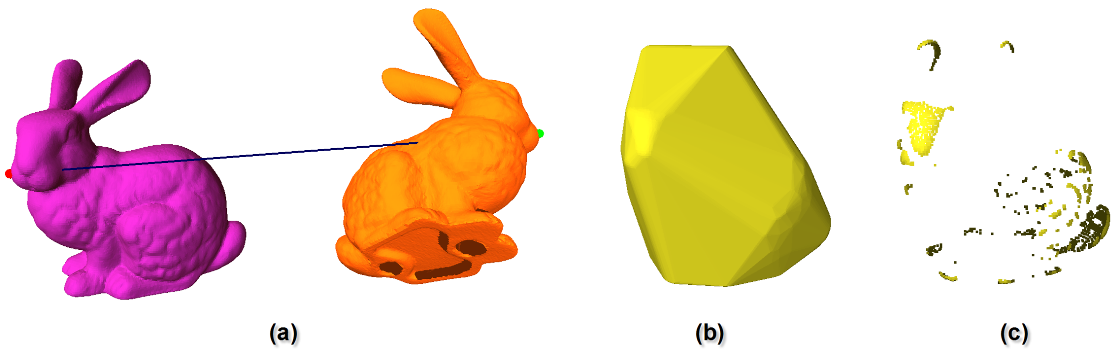

Figure 6: C-DIST *Computation for* Bunny *model. This model has* 69,451 *triangles. (a) shows a scenario of the* DISP *distance query for the model at two different configurations. Our* C-DIST *algorithm can perform the query within* 8.0μs *on average. (b) shows the convex hull of this model. (c) shows the vertices on this convex hull, which are not uniformly distributed in 3D space.*

fraction of the running time in distance computation.

6.2 Continuous Collision Detection

In applications such as dynamics simulation and motion planning, one often needs to check whether or not a model A collides with the environment B when A continuously moves along a given motion path $\mathbf{c}(t)$, $t \in [0,1]$. This problem is referred to as a *continuous collision detection* (CCD) problem and many approaches have been proposed to solve it efficiently [Redon et al. 2002; Schwarzer et al. 2005; Zhang et al. 2006b]. A class of CCD solutions is based on the following conservative test: if $A(\mathbf{c}(0))$ does not collide with B and the *separation distance* between them is greater than the maximum distance that any point on $A(\mathbf{c}(t))$ can travel during $t \in [0,1]$, it is safe to assume that no collision occurs during $t \in [0,1]$. In order to compute the traveled distance, one may integrate the trajectory traced by any point of $A(\mathbf{c}(t))$ from $t = 0$ to $t = 1$. Due to the high cost of computing the maximum exact trajectory length, most algorithms typically compute an upper bound, which is referred to as *motion bound* and use it for performing a conservative CCD test.

Based on the DISP displacement metric, we present a tighter CCD test condition by using the notion of *maximum length of displacement* instead of *maximum trajectory length* used in prior CCD methods [Schwarzer et al. 2005]. The maximum length of displacement corresponds to the longest length of displacement vectors of a moving object A, and can be formally defined as:

$$\lambda(\mathbf{c}) = \max\{\text{DISP}(\mathbf{c}(0), \mathbf{c}(t)) | t \in [0,1]\}. \tag{4}$$

Since λ characterizes the maximum distance that any point on A can be displaced, it is valid to replace maximum trajectory length by λ in prior CCD methods.

Our new CCD test condition is less conservative due to the fact that the length of the displacement vector for any point on A is never longer than the length of its trajectory. As Fig. 7 illustrates, especially when the model A involves a large amount of rotational motion, the difference between these two bounds can be large.

Displacement bound for screw motion. According to Eq. (3), one can compute an exact maximum displacement when the screw angle θ is within $[0,\pi]$, and compute a relatively tighter upper bound (π,∞) by the following equation:

$$\begin{cases} \lambda^2 = 2(1-cos\theta)r^2 + d^2 & 0 \le \theta \le \pi, \\ \lambda^2 \le 4r^2 + d^2 & \pi < \theta < \infty, \end{cases} \tag{5}$$

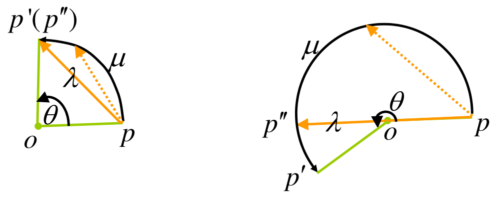

Figure 7: $\lambda \le \mu$. *When the point* $\mathbf{p}$ *is rotated around* $\mathbf{o}$ *by* θ *to* $\mathbf{p}'$*, the maximum length of its displacement vector* λ*, which is* $||\mathbf{pp}''||$*, is less than its maximum trajectory length* μ*, which is* $||\widehat{\mathbf{pp}'}||$ *for both cases. The larger the rotation angle* θ*, the bigger the difference between the two quantities.*

where d, $\theta \ge 0$ are the same as Eq. (3), and $r = \eta(A,\omega)$ is the maximum distance between any point on A to the screw axis and can be computed as a preprocessing step.

The displacement bound λ for screw motion is never larger than the motion bound. Moreover, according to Eq. (5), for the case that the rotational angle θ is larger than π, the displacement bound is much less than the motion bound. Therefore, for a CCD test with the same amount of separation distance, using a displacement bound can yield a tighter CCD test condition and make the overall algorithm less conservative. Considering that λ can be efficiently evaluated by Eq. (5), the new condition can be used to improve CCD algorithm for screw motion.

6.3 Generalized Penetration Depth Computation

In dynamics simulation, haptic rendering and virtual environment, it is often necessary to quantify the amount of interpenetration between two intersecting models. One of such measures is *penetration depth*. [Ong 1993; Zhang et al. 2006a] have proposed a formulation for *generalized penetration depth* PD^g, which takes into account both translational and rotational motions to separate two interpenetrating models. The definition 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. The D_g metric is used in [Zhang et al. 2006a]. However, that distance metric is difficult to compute, and one can only compute its upper and lower bounds. Since the DISP distance metric can be efficiently computed, it is more suitable to define a generalized penetration depth PD^g_{DISP} for two models A and B, i.e.:

$$\mathrm{PD}^g_{DISP}(A,B) = min(\{DISP_A(\mathbf{q}_0,\mathbf{q}) | \,\mathrm{interior}(A(\mathbf{q})) \cap B = \emptyset\}), \quad (6)$$

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

Similarly to PD^g, one can show that for two convex models A and B, $PD^g_{DISP}(A,B)$ is equal to $PD^t(A,B)$. 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. Furthermore, the efficient algorithm on DISP metric computation allows one to efficiently compute an upper bound for PD^g_{DISP} [Zhang et al. 2007].

6.4 Comparison

Compared with other distance metrics used for configuration space distance computation, such as L_p, weighted L_p or other scale-dependent metrics, DISP metric has many elegant mathematical properties. Since it can naturally combine translational and rotational components, the users need not choose any weighting factor between these two components. Using our efficient C-DIST computation algorithm, this metric would be more suitable than L_p and other weighted metrics for many applications, such as motion planning and proximity queries.

DISP metric shares many properties with a model-dependent metric - object norm, which is easier to compute than DISP metric. However, both metrics are 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 displaced 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, where the main goal is to avoid the obstacles.

7 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. All the timings reported in this section were taken on a 2.8GHz Pentium IV PC with 2 GB of memory.

7.1 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 have tested our C-DIST implementation on a set of rigid polyhedral models, including triangular meshes models, such as the *Cup* model, and triangle soup models, such as *Alpha Puzzle*, *Bunny*, *Hand* and *Dragon*. The performance of our C-DIST algorithm is summarized in Table 1.

Fig. 8 shows the *Alpha Puzzle* model as well as its convex hull. In this figure, we place this model at two different configurations and use our C-DIST algorithm to find the vertex with the largest displacement. We highlight the line segment that connects this vertex at the two configurations. For this model, the brute force method of visiting all the vertices on the model takes $184.8\mu s$ on average, for

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

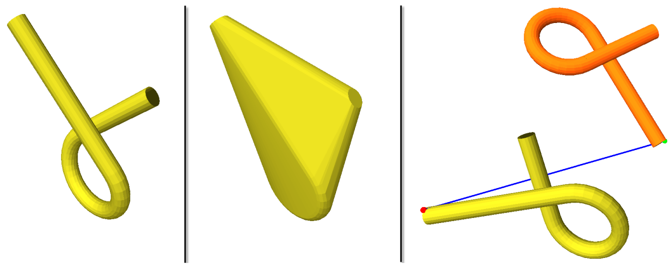

Figure 8: C-DIST *computation. The leftmost and middle figures show the alpha puzzle model with* 1,008 *triangles and its convex hull with* 311 *vertices. The rightmost figure shows a scenario when the model is placed at two different configurations. The vertex on this model, which has the maximum length of 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* $5.6\mu s$.

each DISP query. Our C-DIST algorithm compares all the vertices on the convex hull can perform the query in $19.1\mu s$. Using SSV-Tree, our C-DIST computation can perform the query in $5.6\mu s$.

Figs. 1 and 6 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 $20\mu s$ for these models. We achieve up to 10 times speedup over an algorithm that computes the displacement for each vertex of the convex hull.

7.2 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. 10 highlights an 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 $27.91\mu s$ on average. Fig. 11 shows a complex articulated model with 6 joints - *IRB2400 with an arcgun*. Our algorithm can perform DISP distance query in $21.90\mu s$. Tab. 2 summarizes the performance of C-DIST for these two articulated models.

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 at two arbitrary configurations. We show that for a rigid model, the distance computation reduces to computing the length of the longest displacement vector over 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 configuration. 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. Our algorithm can be easily extended 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 proximity computations and motion planning.

	Alpha Puzzle	Cup	Bunny	Hand	Dragon
#Tri	1,008	4,226	69,451	86,361	871,414
#V	3,024	3,000	208,353	259,803	2,614,242
#V of CH	311	1,019	1,504	1,836	2,448
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_ssv} (μs)	5.6	10.2	8.0	18.2	15.2
Speedup(t_{ch}/t_{op_ssv})	3.4	6.3	10.7	5.76	10.1

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 and building the BVH structure. t_{bf}, t_{ch}, t_{bf} 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 optimized* C-DIST *method that uses a SSV-Tree.*

	#DOF	#Tri	#CH(V)	t_{ch} (μs)	t_{obb} (μs)	Speedup
Puma	9	868	296	42.30	27.91	1.5
IRB-2400	6	3,791	531	73.89	21.90	3.4

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* $30\mu s$.

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:*

$$\gamma(\mathbf{q_0}, \mathbf{q_1}) = \lim_{n->\infty} \sum_{i=0}^{i=n-1} DISP(c(i/n), c((i+1)/n)). \tag{7}$$

One can find 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 qualitatively evaluating the improvement when applying our C-DIST algorithm to sampling-based motion planning and CCD computation. We are also interested in investigating other mathematical properties of DISP metric, such as the Riemannian metric induced by the DISP distance metric. Finally, we would like to formulate and compute other useful metrics in configuration space, such as area and volume.

9 Acknowledgement

We would like to thank Frank Chongwoo Park, Steven M. LaValle, Jean-Claude Latombe and Bert Jütler for useful discussions. 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 grant 2004-205-D00168 of KRF, the STAR program of MOST and the ITRC program.

References

AMATO, N., BAYAZIT, O., DALE, L., JONES, C., AND VALLEJO, D. 2000. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. and Autom. 16*, 4 (Aug), 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*, 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., AND 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*, 63–93.

DE BERG, M., VAN KREVELD, M., OVERMARS, M. H., AND SCHWARZKOPF, O. 1997. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin.

DOBKIN, D., HERSHBERGER, J., KIRKPATRICK, D., AND SURI, S. 1993. Computing the intersection-depth of polyhedra. *Algorithmica 9*, 518–533.

ETZEL, K., AND MCCARTHY, J. 1996. A metric for spatial displacement using biquaternions on so(4). In *Proc. IEEE International Conference on Robotics and Automation*, vol. 4, 3185–3190.

GILBERT, E. G., JOHNSON, D. W., AND KEERTHI, S. S. 1988. A fast procedure for computing the distance between objects in three-dimensional space. *IEEE J. Robotics and Automation vol RA-4*, 193–203.

HOFER, M., AND POTTMANN, H. 2004. Energy-minimizing splines in manifolds. In *SIGGRAPH 2004 Conference Proceedings*, 284–293.

HSU, D., LATOMBE, J.-C., AND MOTWANI, R. 1999. Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications 9*, (4 & 5), 495–512.

JOHNSON, D. E., AND COHEN, E. 2004. Unified distance queries in a heterogeneous model environment. In *ASME DETC*.

KARGER, A., AND NOVK, J. 1985. *Space kinematics and Lie groups*. Gordon and Breach Science Publishers.

KAVRAKI, L., SVESTKA, P., LATOMBE, J. C., AND OVERMARS, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580.

KAZEROUNIAN, K., AND RASTEGAR, J. 1992. 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, 271–275.

KIM, B., AND ROSSIGNAC, J. 2003. Collision prediction for polyhedra under screw motion. *Symposium on Solid Modeling and Applications*.

KUFFNER, J., AND 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'l Conf. on Robotics and Automation*.

LAROCHELLE, P., AND MCCARTHY, J. 1995. Planar motion synthesis using an approximate bi-invariant metric. *ASME Journal of Mechanical Design 117*, 4 (December), 646–651.

LARSEN, E., GOTTSCHALK, S., LIN, M., AND 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., AND MANOCHA, D. 2000. Distance queries with rectangular swept sphere volumes. *Proc. of IEEE Int. Conference on Robotics and Automation*, 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., AND BURDICK, J. 2000. Objective and frame-invariant kinematic metric functions for rigid bodies. *The International Journal of Robotics Research 19*, 6 (Jun), 612–625.

LIN, M., AND CANNY, J. F. 1991. Efficient algorithms for incremental distance computation. In *IEEE Conference on Robotics and Automation*, 1008–1014.

LIN, M., AND 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 Journal of Robotics and Automation 3*, 6 (December), 567–572.

MIRTICH, B. 2000. Timewarp rigid body simulation. *Proc. of ACM SIGGRAPH*.

MURRAY, R. M., LI, Z., AND 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., AND BROCKETT, R. 1994. Kinematic dexterity of robotic mechanisms. *Int. J. Robotics Research 13*, 1 (February), 1–15.

PARK, F. 1995. Distance metrics on the rigid-body motions with applications to mechanism design. *ASME J. Mechanical Design 117*, 1 (March), 48–54.

PLAKU, E., AND 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)*.

REDON, S., KHEDDAR, A., AND COQUILLART, S. 2002. Fast continuous collision detection between rigid bodies. *Proc. of Eurographics (Computer Graphics Forum)*.

SCHWARZER, F., SAHA, M., AND LATOMBE, J. 2005. Adaptive dynamic collision checking for single and multiple articulated robots in complex environments. *IEEE Tr. on Robotics 21*, 3 (June), 338–353.

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

SPIVAK, M. 1999. *A Comprehensive Introduction to Differential Geometry*, 3 ed. Publish or Perish Press.

TCHON, K., AND DULEBA, I. 1994. Definition of a kinematic metric for robot manipulators. *Journal of Robotic Systems 11*, 3, 211–221.

XAVIER, P. G. 1997. Fast swept-volume distance for robust collision detection. In *Proc. 1997 IEEE Int'l Conf. on Robotics and Automation*, 1162–1169.

ZEFRAN, M., AND KUMAR, V. 1998. Interpolation schemes for rigid body motions. *Computer-aided Design 30*, 3, 179–189.

ZEFRAN, M., KUMAR, V., AND CROKE, C. 1996. Choice of riemannian metrics for rigid body kinematics. In *ASME 24th Biennial Mechanisms Conference*.

ZHANG, L., KIM, Y., VARADHAN, G., AND D.MANOCHA. 2006. Generalized penetration depth computation. In *ACM Solid and Physical Modeling Symposium (SPM06)*, 173–184.

ZHANG, X., LEE, M., AND KIM, Y. 2006. Interactive continuous collision detection for non-convex polyhedra. In *Pacific Graphics 2006 (Visual Computer)*.

ZHANG, L., KIM, Y., AND MANOCHA, D. 2007. A fast and practical algorithm for generalized penetration depth computation. Tech. Rep. 07-001, Department of Computer Science, University of North Carolina at Chapel Hill.

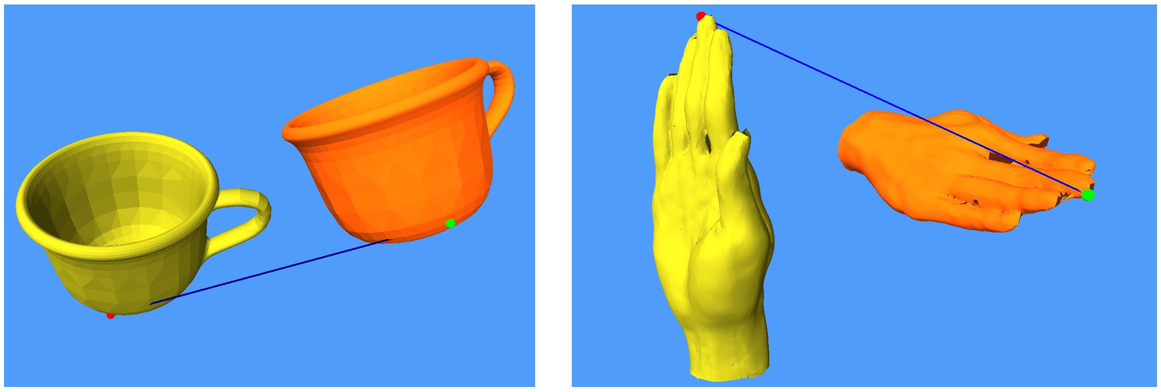

Figure 9: C-DIST *Computation. Our algorithm can handle any polyhedral model represented as a triangular mesh, a triangle soup, or a point-set model. The* Cup *model on the left is represented as a triangular mesh, while the* Hand *model on the right is a scanned model and represented as a triangle soup with* $86,361$ *triangles. Our algorithm can perform the distance query for these models within* $10.2\mu s$ *and* $18.2.\mu s$*, respectively.*

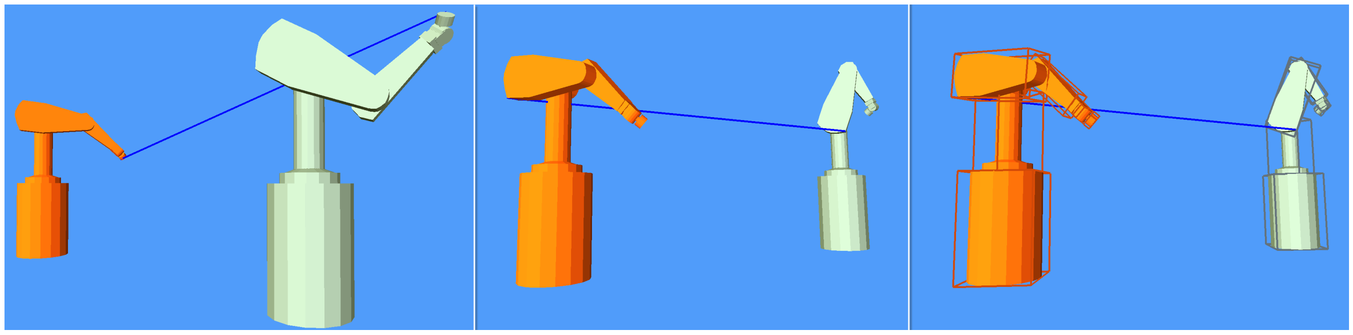

Figure 10: C-DIST *computation for an articulated model:* Puma*. Left: the* Puma *model with* 9 *DOF include* 6 *articulated joints and* 3 *DOF of its base. Middle: the result of the distance query by our* C-DIST *algorithm is highlighted. Right: the OBB associated with each link. For this model, our algorithm can perform the query within* $27.91\mu s$*, on average.*

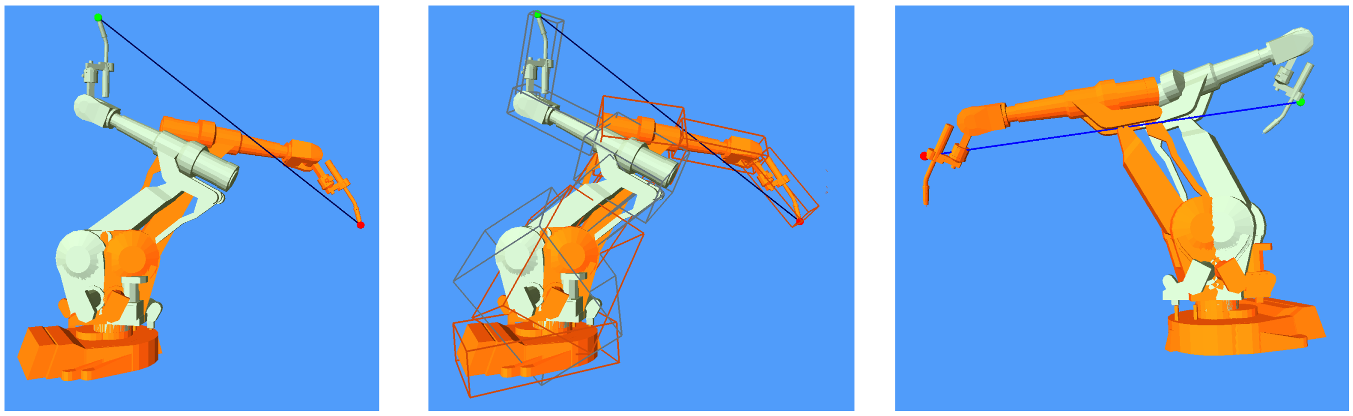

Figure 11: C-DIST *computation for an articulated model:* IRB2400 with an arcgun. *This model has 6 joints and 3,791 triangles. Our algorithm can perform the query within* $27.91\mu s$ *on average. In the middle, we highlight the OBB associated with each link which can be used for efficient culling.*