Computing the Inner Distances of Volumetric Models for Articulated Shape Description with a Visibility Graph

Yu-Shen Liu, Karthik Ramani, Min Liu

Abstract—A new visibility graph based algorithm is presented for computing the inner distances of a 3D shape represented by a volumetric model. The inner distance is defined as the length of the shortest path between landmark points within the shape. The inner distance is robust to articulation and can reflect well the deformation of a shape structure without an explicit decomposition. Our method is based on the visibility graph approach. To check the visibility between pairwise points, we propose a novel, fast, and robust visibility checking algorithm based on a clustering technique which operates directly on the volumetric model without any surface reconstruction procedure, where an octree is used for accelerating the computation. The inner distance can be used as a replacement for other distance measures to build a more accurate description for complex shapes, especially for those with articulated parts. The binary executable program for the Windows platform is available from https://engineering.purdue.edu/PRECISE/VMID.

Index Terms—Inner distance, visibility graph, articulated shape descriptor, volumetric models.

1 Introduction

Description and understanding of 3D shapes play a central role in computer vision, computer graphics, and pattern recognition. One of the principal challenges faced today is the development of shape descriptors with new capabilities. The simplest and most widely used shape descriptor may be based on the distance measure between sampling point pairs on shape surfaces, typically, which measures the Euclidean distance or geodesic distance. However, neither the Euclidean distance nor geodesic distance could not reflect shape articulation well. This paper presents a new visibility graph based algorithm for computing and applying the inner distances [1] in a 3D articulated shape represented by a volumetric model. One advantage of using the inner distance is that it is robust to articulation and it reflects the shape structure and deformation well without an explicit decomposition.

A shape descriptor is a concise representation of a shape and captures some of its essence [2]. After the descriptors are extracted from 3D geometric objects, they are compared in order to determine the shape similarity instead of comparing the full 3D models. Shape descriptors can be roughly classified by their invariance with respect to different shape transformations [2]–[4]. One class of shape descriptors, called *extrinsic* shape descriptor, captures some rigid-body-transformation invariance of global geometric properties (e.g. scale, translation and rotation). These techniques include shape distribution, moment invariance, spherical harmonic descriptor, 3D Zernike descriptor, etc. [5]–[8]. Most existing methods are only effective for comparing 3D rigid objects, but they can not handle the deformed shapes of flexible objects well. However, many 3D objects, such as living objects and molecules, are flexible and their spatial arrangement or pose may change.

Another class of shape descriptors, called *intrinsic* shape descriptor, is invariant with respect to isometric transformation or articulated deformation. These descriptors do not change with isometric embedding or articulation of the shapes. Several recent works have been developed for this purpose based on geometric or topological attributes of the shape, or both [2], [4], [9]–[17]. Some methods focus on topology or graph comparison [11], but the graph extraction process is often very sensitive to local shape changes and noise. Several techniques take into account the local features on the boundary surfaces of a shape [13]. In particular, Gal et al. [2] proposed the local diameter shape signature by computing the distance from a boundary surface to its medial axis. Other methods have been proposed for partial shape matching [18], [19]. However, many times, the local and partial methods do not provide a good description of the overall shape [2]. Some recent studies have also dealt with the nonrigid shape matching using diffusion distance, heat kernels, and spectral approaches [10], [16], [17], which consider topological changes of surface deformation.

Measuring the distance between sampling point pairs on surfaces of shapes is a fundamental problem. Here we compare three representative distances: Euclidean distance (ED), geodesic distance (GD), and inner distance

- *Y.S. Liu is at School of Software, Tsinghua University, Beijing 100084, China; Key Laboratory for Information System Security, Ministry of Education; Tsinghua National Laboratory for Information Science and Technology. E-mail: liuyushen@tsinghua.edu.cn. K. Ramani is at Purdue University, West Lafayette, IN, USA. E-mail: ramani@purdue.edu. M. Liu is at Institute of Manufacturing Engineering, Tsinghua University. E-mail: minliu@tsinghua.edu.cn. The core work was done during Y.S. Liu's post-doctoral studies at Purdue University.*

(ID). The ED descriptor [7] (or D2) is usually represented by a histogram of distance values, but it can not handle the deformed shapes well. An alternative way [4], [20], [21] is to replace ED by GD, which is defined as the length of the shortest path on the surface between sample points. Elad and Kimmel [4] presented a bending invariant descriptor for surface by combining GD and multidimensional scaling. Hamza and Krim [22] used GD for classification of 3D mesh models instead of ED in shape distributions. Ruggeri et al. [21] used a similar idea for matching point set surfaces. Bronstein et al. [20] applied GD for isometry-invariant partial surface matching. Although GD is insensitive to surface bending, it is weak in discriminating power for handling articulated objects [1], [2], [9], [23].

Ling and Jacobs [1], [23] recently proposed an algorithm for computing and applying the inner distances for building new 2D shape descriptors. ID is robust to articulated deformation and it is more effective at capturing shape structures than either ED or GD. Based on [23], Biswas et al. [24] proposed a framework for 2D shape indexing and retrieval. Liu and Zhang [25] embedded a surface mesh in 3D into a planar contour and then computed pairwise inner distances between sample points along the 2D contour using [23]. Bronstein et al. [9] analyzed 2D articulated shapes with inner distances. More recently, Rustamov et al. [14] introduced the interior distance between two points inside the mesh, which is obtained by propagating the boundary distance into the interior using barycentric coordinates.

Here we present a new method for computing the inner distances for a 3D shape represented by a volumetric model. One common way to compute pairwise inner distances between 2D contour points is to first construct a *visibility graph*, and then find the shortest path in the graph as the inner distance [1], [23], [25]. We follow a similar way for computing the inner distances for 3D shapes. Our work can be considered as an extension of computing the inner distances from 2D to 3D based on the visibility graph approach. One of the main difficulties that we have overcome was checking the visibility between sample points of the volumetric model for constructing a visibility graph. To resolve this problem, we propose a novel, fast, and robust visibility checking algorithm based on a clustering technique which operates directly on the volumetric model without any surface reconstruction procedure.

2 Preliminaries

2.1 Definition and Properties

We extend the definition of inner distance within a 2D silhouette [1] to a 3D solid. Let O be a 3D object as a connected and closed subset of $\mathbb{R}^3$.

Definition 1 (Inner distance): Given two points $\mathbf{x}, \mathbf{y} \in O$, the inner distance between $\mathbf{x}$ and $\mathbf{y}$, denoted as $d(\mathbf{x}, \mathbf{y}; O)$, is defined as the length of the shortest path connecting the two points within O.

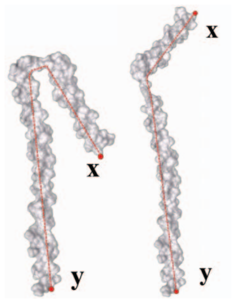


Fig. 1. The red dashed lines that connect two landmark points $\mathbf{x}$ and $\mathbf{y}$ denote the shortest paths within the molecular shapes. The right object is an articulated deformation to the left one, and the relative change of the inner distance between the corresponding pair of points ($\mathbf{x}$ and $\mathbf{y}$) is small during articulated deformation [26].

When O is convex, the inner distance reduces to the Euclidean distance. Apparently, the inner distance is invariant to rigid transformation. The general problem of computing the inner distance is quite complicated, and we make some simplifying assumptions. When there are multiple shortest paths, we arbitrarily choose one. In practice, we are interested in the shape of O, which is defined by its boundary surface denoted by ∂O. Therefore, we restrict landmark points sampled on the boundary surface ∂O. Fig. 1 illustrates the inner distance.

2.1.1 Articulation Insensitivity of Inner Distance

One appealing feature of the inner distance is articulation insensitivity. In this work, we consider O as a model of *articulated object*. Intuitively, O is an articulated object if (1) O can be decomposed into several rigid parts connected by flexible *junctions* and (2) the junctions between parts are very small compared to the parts [1]. Let Φ be a transformation that changes the pose of an object O. Φ is roughly considered as *articulated transformation* if the transformation of any part of O is rigid (rotation and translation only) and the transformation of the junctions can be nonrigid. This preserves the topology between the articulated parts. Intuitively, the inner distance reflects the shape structure and articulation well even without explicitly decomposing the shapes into parts. Ling and Jacobs [1] have proven that the inner distance is insensitive to articulated transformation of O by decomposing O into some rigid parts connected by junctions, where the change in the inner distance only depends on the size limitation of the junctions. Since we assume that the junctions of an articulated object O should have relatively small sizes compared to O, the relative change in the inner distance is very small too.

2.1.2 Comparison with Motion Planning

The inner distance path discussed in this paper shows some similarities to the shortest path in the *motion planning problem*, which typically finds the collision-free shortest path between two points in a closed space [27].

But unfortunately, computing the shortest path within a 3D polyhedron is an NP-hard problem. Numerous methods have been developed to deal with this problem such as the visibility graph, potential field, Voronoi diagram, and cell decomposition, most of which are only effective in 2D. In contrast with the classical motion planning problem with polyhedral obstacles, there are some new challenges in computing the inner distances for shape description. The 3D objects discussed between the motion planning problem and our application are distinct. In shape retrieval, many polygonal mesh models may contain holes, intersecting polygons, or be non-manifold, and do not enclose a volume unambiguously. In comparison to 3D motion planning, which solves an entirely collision-free shortest path between one pair of points, the presented inner distances demand relatively short computation time between all sample point pairs, and an approximate solution is acceptable. These observations lead to demand for other shape representations that enclose an unambiguous volume and are effective in computing the inner distances.

2.2 Volumetric Models

To avoid the effect of “polygon soups” in inner distance computation, we have chosen the volumetric models instead of polygonal meshes. Note that our method does not handle these situations of polygon soups since it relies on a voxelization procedure to repair the models and make them water-tight. In contrast, the volumetric representation encloses a volume properly and allows computation that is more robust to noise and perturbations [28]. The volumetric data is also common to many applications in medical scanners (e.g. CT and MRI), scientific simulations (e.g. CFD) [29], computational biology, and shape signatures [8], [30]–[33]. We consider the input data as a volumetric/voxelized representation of a 3D shape, which can be regarded as an approximation of the volume enclosed by a polygonal mesh. A volumetric model is represented as a uniform 3D lattice consisting of object points O and background points $\overline{O}$. We denote the boundary surface of O by ∂O. Fig. 2(a) shows all boundary points of ∂O colored in light gray. Our method is also suited to other shape representations through converting them into volumetric forms using some free software, such as *PolyMender* [34]. Volumetric data is generated by placing a 3D shape into a 3D cubic grid, where each lattice point is assigned either 1 or 0: 1 for object points O and 0 for background points $\overline{O}$.

2.3 The Visibility Graph

We have chosen the visibility graph for approximating the inner distance path, where the visibility graph is constructed by the links connecting the visible nodes. The shortest path between two given nodes goes through arcs of the visibility graph. Many works in 2D and 3D motion planning find the shortest path based on the visibility graph approach [27]. The key to constructing a visibility graph is to check the visibility between two nodes, which means that two nodes are connected in a graph if the corresponding points can see each other. For our purpose, we define the visibility between two sample points within a 3D object. In the 2D case, the visibility is easy to solve by intersecting the line segment connecting the source and target points with the 2D polygon contour [1], [23], [27]. However, it is a non-trivial task to check the visibility for a 3D volumetric model.

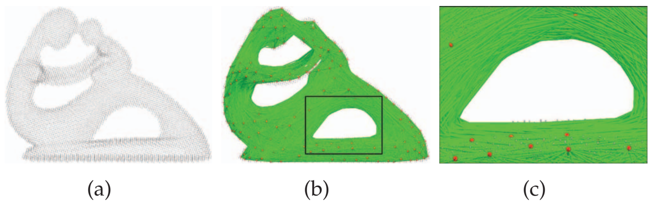


Fig. 2. Illustration of computing the inner distances. (a) All boundary points of the volumetric model of a fertility are colored in light gray. (b) The shape with 500 uniform sample points (red color) and their inner distances (green lines). (c) The magnified views. Note that the inner distances capture the shape, especially for some holes.

The visibility between two points has been widely researched in motion plan and computational geometry [27]. Several recent works have discussed the visibility in discrete geometry based on a volumetric representation [30]. However, most of these algorithms focus on the visibility outside a given object. In our case, the problem is quite different, where we want to decide if there is at least one path inside the object. Soille [35] defined the visibility between two points using the Bresenham digital line drawing algorithm. Coeurjolly et al. [36] presented a discrete definition of the classical visibility based on digital straight lines. Many existing techniques determining the visibility between two points are based on tracking the neighborhood from the starting point to the target one [30], [36], which becomes more complex than in 2D and the computational cost is high in 3D. The reader may consult Ref. [36] for a review of related work. In this paper, we present a fast algorithm for checking visibility based on a clustering idea.

3 Computing the Inner Distance

The outline of an algorithm for computing an approximation of volumetric model inner distances (**VMID**), is given below.

Algorithm 1 (VMID):

1. Sample k points $S = \{\mathbf{p}_1, ..., \mathbf{p}_k\}$ on the boundary surface ∂O of O using k-means clustering.
2. Define a visibility graph G over all sample points by connecting points $\mathbf{p}_i$ and $\mathbf{p}_j$ in S if $\mathbf{p}_i$ and $\mathbf{p}_j$ are visible, and an edge between $\mathbf{p}_i$ and $\mathbf{p}_j$ is added to the graph with its weight equal to the Euclidean distance $\|\mathbf{p}_i - \mathbf{p}_j\|$.

3. Compute the inner distance by applying a shortest path algorithm to the graph G.
4. (Optionally) Build the shape descriptor of O as the histogram of values of inner distances.

An example for illustrating the inner distance computation is shown in Fig. 2.

3.1 Sample Points

The input volumetric model consists of a point array. If all points of the boundary surface ∂O are utilized for the final inner distance computation, this will increase the storage and computing costs. Therefore, we first sample points from ∂O. One issue of concern is the sample density. The more samples we take, the more accurately and precisely we can reconstruct the shape. However, a large number of sample points increases the storage and computation costs, so there is an accuracy/time tradeoff in the choice of the number k of sample points. In our experiments, we have found that using $k \in [300, 1000]$ yields shape distributions with low enough variance and high enough resolution to be useful for our initial experiments. A second issue is the sampling method. We implement two sampling methods: random sampling and k-means clustering based sampling. Random sampling can not yield a good approximation. We use Lloyd's algorithm of k-means clustering for obtaining the uniform clusters of boundary points [37]. The point in each cluster which is nearest to its centroid is chosen as the representative sample point. Other non-uniform point clustering techniques such as curvature-adaptive clustering [38] can also be applied to our work by restricting different stopping criteria, but this leads to an increase in computation time.

3.2 Visibility Check

A crucial part for defining a visibility graph G is to check the visibility between all pairwise points. To address this problem, we present a novel, fast, and robust visibility check algorithm based on a clustering technique that does not require any surface reconstruction. In some sense, our clustering idea can be considered as an extension of the ray tracing routine [39], which directly computes the intersection of a ray and a point cloud by intersecting a cylinder around the ray with the point cloud. Then, the intersection is computed as a weighted average of the points that are inside the cylinder. However, [39] only finds one of the intersection points. We extend their method based on a clustering technique. Unlike the previous intersection approaches [39], [40], our algorithm does not reconstruct or approximate a surface from a point cloud while finding all intersection points. A similar clustering idea has appeared in our previous work [41] for computing the area of a point-sampled surface. In [41], a major drawback is that the intersection problem is strongly dependent on the normal vectors of points on the boundary surface. In contrast, the visibility check algorithm presented in this section is normal-free. The procedure is described as follows (see Fig. 3).

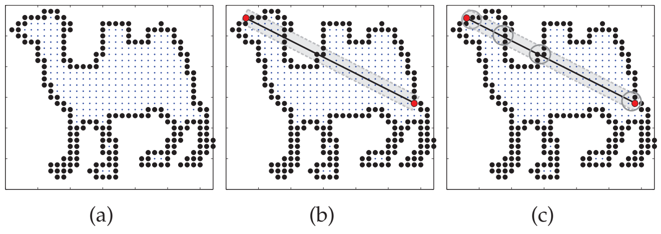


Fig. 3. Illustration of the visibility check for a 2D model. (a) The input point array (blue points) sampled from a camel image with extracting the border (black points). (b) Collect the inclusion points that are inside a cylinder (gray) around a line segment connecting two sample points (red). (c) Cluster the inclusion points, where each cluster is highlighted in a circle.

*Algorithm 2 (**VisibilityCheck**):*

1. Collect the boundary points of ∂O inside a cylinder around a line segment $\boldsymbol{\ell}$ connecting two sample points $\mathbf{p}_i$ and $\mathbf{p}_j$.
2. Cluster the collected points by projecting them onto the line segment $\boldsymbol{\ell}$.
3. Classify the clusters for checking the visibility.

3.2.1 Collecting Inclusion Points

We consider a cylinder around $\boldsymbol{\ell}$ with the radius r (see Fig. 3(b)). To determine r, we need to obtain the density of the point set ∂O, where the density is the maximum size of a gap in ∂O. Suppose that d is the edge length of a voxel and it usually is set as a unit value (i.e. $d \equiv 1$), then the longest distance in the neighborhood around a voxel is $\sqrt{3}d$. Therefore, the density radius is chosen as $\sqrt{3}d$ in this paper. Typically, we choose $r = \sqrt{3}d$ as the radius of the cylinder for obtaining sufficient intersection points and less time. We call these points of ∂O inside the cylinder, the *inclusion points*.

Octree construction. To speed up searching for the inclusion points, we use octrees described in [42]. Assuming that an octree is constructed for ∂O, we classify the cells of the octree as two types: boundary cells containing points of ∂O, and empty cells containing no point of ∂O. Let $\mathcal{C}_{\mathrm{b}} = \{C_i\}$ be a set of boundary cells, where C_i is the i^{th} boundary cell. If $\boldsymbol{\ell}$ does not intersect with C_i, we continue looking for another boundary cell until the intersection yields. All points of ∂O within the cylinder that intersects the cells C_i are collected as inclusion points of $\boldsymbol{\ell}$ with ∂O, as shown in Fig. 4.

3.2.2 Clustering by Projection

After collecting the inclusion points, we then cluster them (see Fig. 3(c)) for counting the number of intersection points. Clustering methods are widely used in computer graphics to reduce the complexity of 3D objects.

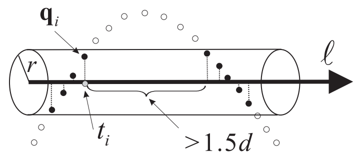


Fig. 4. Cluster the inclusion points by projection, where r is the radius of a cylinder around $\boldsymbol{\ell}$ connecting two sample points, and the black points denote the inclusion points inside the cylinder. We map each inclusion point $\mathbf{q}_i$ into 1D parameter coordinate $t_i \in \mathbb{R}$ by projecting $\mathbf{q}_i$ onto the line segment $\boldsymbol{\ell}$.

For instance, Pauly et al. [38] used region-growing and hierarchical clustering methods to simplify point clouds. Unlike [38], our clustering method maps 3D points into 1D parameter coordinates by projecting the inclusion points into the line segment $\boldsymbol{\ell}$. Suppose that $\{\mathbf{q}_i\} \subseteq \partial O$ is a set of inclusion points of $\boldsymbol{\ell}$ and ∂O. We first project each point $\mathbf{q}_i$ onto $\boldsymbol{\ell}$, and get one corresponding parameter $t_i \in \mathbb{R}$. Hence we obtain a set $\{t_i\}$ of parameters. Secondly, the set $\{t_i\}$ is sorted in increasing order. Here we suppose below that $\{t_i\}$ has already been sorted. Finally, we build the clusters by $\{t_i\}$ as described below. Starting from the minimal parameter of $\{t_i\}$, a cluster Q_0, which is a set of some inclusion points in $\{\mathbf{q}_i\}$, is built by comparing the distance of adjacent parameters. This cluster is terminated when the distance of two adjacent parameters is larger than a maximum bound (we typically choose $1.5d$ as the bound). Then, starting from the terminated parameter, the next cluster is built repetitively. Clustering is terminated until the maximal parameter is reached. The center of each cluster can be regarded as one intersection point. After classifying the clusters, we choose the number of resultant clusters as the number of intersection points. An upper bound of the running time to the clustering between two sample points $\mathbf{p}_i$ and $\mathbf{p}_j$ is $O(m\log(m))$ because of sorting 1D parameter coordinates of inclusion points, where m is the number of total boundary points ∂O. Fig. 4 shows the procedure for building clusters by projection.

3.2.3 Classifying Clusters for Checking Visibility

According to the number of clusters, we classify the intersection into the following three cases: (1) containing only one intersection point (visibility); (2) containing two intersection points (visibility/invisibility); (3) containing more than two intersection points (invisibility). For case 1, where the number of intersection points is only one, the line segment $\boldsymbol{\ell}$ connecting $\mathbf{p}_i$ and $\mathbf{p}_j$ is coincident with the boundary surface ∂O, and we consider this case as visible. For case 3, where the number of intersection points is more than two, $\mathbf{p}_i$ and $\mathbf{p}_j$ are classified as invisible. For case 2, $\mathbf{p}_i$ and $\mathbf{p}_j$ are either visible or invisible. Either the line segment $\boldsymbol{\ell}$ falls entirely inside the given shape O or $\boldsymbol{\ell}$ (except the two endpoints) falls outside O. To distinguish between inside or outside, we compute the midpoint of $\mathbf{p}_i$ and $\mathbf{p}_j$ and check the density value of the closet neighbor lattice point of the midpoint. If the density value is nonzero, $\mathbf{p}_i$ and $\mathbf{p}_j$ are visible; they are invisible, otherwise. After all pairs of sample points are checked for visibility, we can define the graph G.

3.3 Computing the Shortest Path

We estimate the inner distances between all sampling point pairs by computing their shortest paths. Algorithms for finding the shortest paths in a graph are well known. For example, Hilaga et al. [11] measured the geodesic distance on a triangulated surface by the Dijkstra graph search algorithm. Elad and Kimmel [4] used the fast marching on triangulated domains. Here, we choose Dijkstra's algorithm to compute the inner distances between all sampling points in the visibility graph G. Dijkstra's algorithm is a graph search algorithm that solves the single source shortest path problem for a graph. The time complexity is $O(k^3)$ for k sample points. In order to implement Dijkstra's algorithm more efficiently, Fibonacci heap is used as a priority queue. We use the code package of Dijkstra's algorithm [43].

Now we convert a set of inner distances to a shape descriptor in a similar manner to shape distribution [2], [7]. Given k sample points, the number of inner distances of the shape is at most $k^2/2$. Specifically, we evaluate $k^2/2$ inner distance values from the shape distribution and construct a histogram by counting how many values fall into each of the fixed sized bins (we typically choose 128 bins). Empirically, we have found that using $k = 500$ samples and 128 bins yields shape descriptors with low enough variance and high enough resolution to be useful for our experiments.

Computational complexity. We assume that the input volumetric object O consists of n object points, where ∂O contains m boundary points ($m \ll n$). An upper bound of the running time to check the visibility between one pair of sample points is $O(m\log(m))$, as discussed before. As a result, the complexity of visibility check between all pairs of sample points is $O(k^2 m\log(m))$ for k sample points on ∂O ($k \leq m$). After the graph is ready, the all-pair shortest path algorithm, known as Dijkstra's algorithm, requires $O(k^3)$. The time complexity of this algorithm, using Fibonacci heaps in the implementation of Dijkstra's algorithm, is $O(k^2\log(k) + k|E|)$, where $|E|$ is the number of edges in the visibility graph. For our inner distance computation, a great deal of the running time is spent in the visibility check stage.

Alternatively, one can use the fast marching method (FMM) for computing the inner distances on a volumetric model O, which takes $O(kn\log n)$ between all pairs of k sample points.

4 Results and Discussion

We have implemented our algorithm in C++ and experimented with a large number of volumetric models. All

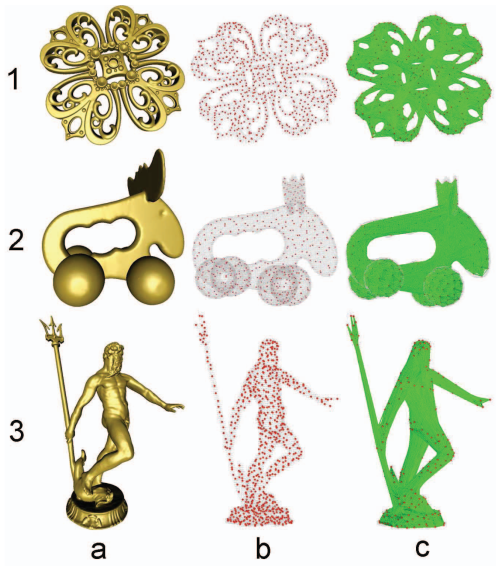


Fig. 5. Visualizing the inner distances of three volumetric models with 500 sample points each. Rows 1–3: filigree, elk, and neptune. Column **a** shows the original surfaces of three models and column **b** shows the corresponding sample points (red), while column **c** shows the paths of inner distances (green lines).

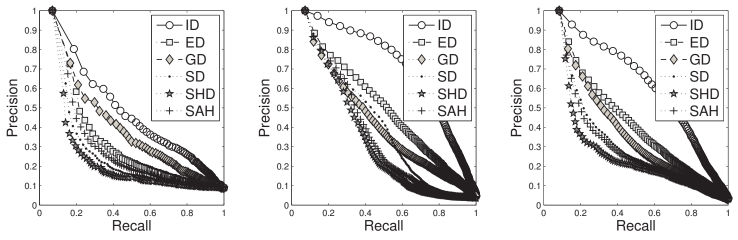
TABLE 1
Computational time.

Model	Fig.	n	m	k	T1[a](s)	T2[b](s)	T3[c](s)
Fertility	2	83,555	26,546	500	0.29	14.3	339
Filigree	5	24,974	19,519	500	0.02	13.3	90
Elk	5	160,801	47,696	500	0.25	28.3	702
Neptune	5	20,154	11,251	500	0.03	8.1	71

a. T1 is the time for constructing an octree at depth 5.
b. T2 is the time for computing inner distances with our method.
c. T3 is the time for computing inner distances using FMM.

the experiments were run on a PC with Pentium Dual-Core 2.60 GHz CPU and 2G RAM, excluding the time for loading models. An axis-aligned octree at depth 5 is constructed in the preprocessing step and this procedure usually takes less than 0.5 seconds. We use 500 sample points for approximating the inner distances of all models.

To show the ability of inner distances approximating 3D shapes, we demonstrate our method on several artificial and scanned models in Fig. 5. Each model is originally represented in polygonal formats and converted into volumetric forms using *PolyMender* [34]. Observe how the inner distances capture the holes for filigree and elk. Table 1 gives the time in seconds for some volumetric models referred to in this paper, where "n" is the number of object points of the models, "m" is the number of boundary points, and "k" is the number of sample points. For our test, our method is around 20 times faster than FMM for some complex models, where FMM was implemented using a Fibonacci heap.



(a) ISDB (b) MolMovDB (c) CDB

Fig. 6. Precision-recall curves computed with several known descriptors for three databases.

To demonstrate the utility of deformation invariant descriptors, we have developed a shape search system and tested this system for several databases containing abundant articulated deformation of living objects and molecules. In particular, many molecules of interest are flexible and undergo significant shape deformation as a part of their function, but most existing methods of molecular shape comparison treat them as rigid bodies [8], [44], which may lead to incorrect measures of the shape similarity of flexible molecules. To address this issue, we have applied the presented inner distance descriptors for retrieving the Database of Macromolecular Movements (MolMovDB) [26]. MolMovDB presents a diverse set of molecules that display large conformational changes in proteins and other macromolecules (see http://www.molmovdb.org/). The user selects a 3D model from the database and the application computes the similarity measure for all models in the database. To compare the effectiveness of the proposed descriptor, we executed a series of shape matching experiments with three different databases of 3D articulated models:

- **ISDB** [2] is a database of different articulated models of animals and humans containing about 104 models.
- **MolMovDB** [26] is a database of different molecules with large conformational changes classified into 214 groups with a total of 2,695 PDB files.
- **CDB** is a union of the two previous databases.

We have pre-calculated all models in three databases. In general, computing the inner distances for most query models does not take more than 20 seconds. In all examples, the query time on a large database took under a second. The ID descriptor is compared with several known descriptors [6]: ED, GD, Shape Distribution (SD), Spherical Harmonic Descriptor (SHD), and Solid Angle Histogram (SAH) in terms of the performance in retrieving similar shapes. We use standard evaluation procedures from information retrieval, namely *precision-recall* curves, for evaluating the various shape distance descriptors [45]. Precision-recall curves describe the relationship between precision and recall for an information retrieval method. A perfect retrieval retrieves all relevant

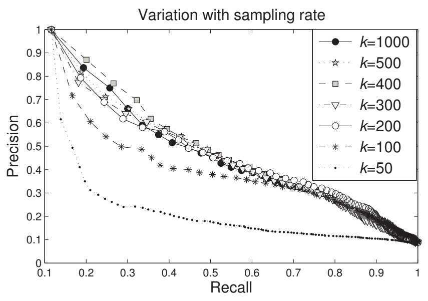


Fig. 7. Variation with increasing sampling rates k.

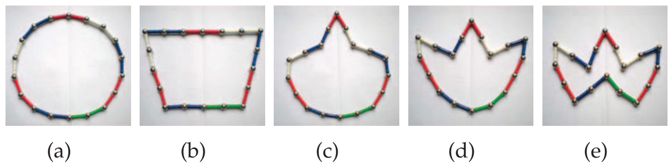


Fig. 8. Using GD, the above five shapes will have the identical pairwise boundary geodesic distances. For articulated shapes, we assume they are composed of different part structures and thus should have dissimilar shape descriptors. While using ID, these shapes are distinguishable since the inner distance changes when the part structures evolve.

models consistently at each recall level, producing a horizontal line at a precision = 1.0. Fig. 6 shows the average precision-recall curves for the ISDB, MolMovDB, and CDB. The detailed evaluation is given in **Supplementary Material**. The results show that the ID method performs better than other descriptors for flexible models in the three databases. Surprisingly, the GD method performed worse than the ED method in Figs. 6(b) and 6(c). Although GD is insensitive to surface stretching or tearing, it remains invariant to all inelastic deformations, i.e., deformations that do not "stretch" or "tear" the object, as long as the deformation preserves geodesic (curve lengths) on the object boundary. From our experiments, we also found that some molecules with one domain are often judged as similar to others with two or three domains when using GD descriptors. In the molecule database, GD can not give good search results as well as compared to ED. Fig. 7 demonstrates the variation in precision-recall curves of ID with increasing sampling rates $k \in \{50, 100, 200, 300, 400, 500, 1000\}$ for ISDB. The high sampling rate (e.g. $k \geq 300$) performs better than the low one (e.g. $k = 50$), while performance variation is small when $k > 500$.

Comparison with GD. Natural articulated shapes, although they belong to the nonrigid deformable shape category, have a deformation freedom which is actually limited since each rigid part can only rotate around a flexible junction. Therefore, articulation can actually be considered as a decomposed rigid transformation. If we assume the ideal case such that all junctions are degenerate to single points, an articulation is a topology-preserving map, which Euclidean-isometrically maps parts and junctions. From this perspective, both GD and ID metrics are insensitive to articulated deformation. Our argument for choosing ID instead of GD is that, like ID, GD is invariant to articulations, but GD is also invariant to broader classes of deformation (bends) which preserve geodesic metric. The shape can be bent drastically and ends up with significant structural changes in terms of subparts or junctions. Therefore, GD is weak in discriminating power for handling articulated objects. Fig. 8 illustrates our point better, in which all five shapes are isometries in geodesic metric space; therefore, using GD, they are indistinguishable. On the other hand, ID is more appropriate since it reflects the essential invariant property for articulation. ID is in essence equal to ED inside each part while it remains unchanged for structure preserved sub part rotation. It is meanwhile very sensitive to concavity/convexity changes of shape. This makes ID superior to GD as an articulation invariant shape representation.

Comparison with diffusion distance. Coifman et al. [46] introduced diffusion maps and diffusion distances as a method for data parametrization and dimensionality reduction. The *diffusion distance* between two points in the point cloud involves the average of all the paths connecting these two points (average probability of travelling between the points). Some recent studies [10], [14], [47] have applied the diffusion distance to shape description. The diffusion distance is also a bending invariant function. In contrast to the traditional distance measure (such as ED or GD), which is defined as the length of a single path connecting two landmark points, the diffusion distance is the average of multiple paths between two points. This paper only considers the distance measured along a single path. However, it is of interest to explore the visibility graph with the diffusion distances for computing the average length of paths connecting two landmark points in the sense of inner distances, which can help resolve the topological sensitivity problem for shape articulation; this is, however, beyond the scope of this paper. We leave this application to a separate publication.

5 Conclusion

We have presented a new method for computing and applying the inner distances for 3D volumetric shapes with a visibility graph. The inner distance for shape description has some common properties shared with many well known extrinsic shape descriptors. For instance, it is invariant under rigid transformation and insensitive to noise and small local geometric perturbations. Furthermore, it has some additional features compared to the existing intrinsic shape descriptors. One main advantage of the inner distance is that it is insensitive to articulated deformation and can approximate the complex shapes.

SUPPLEMENTARY MATERIAL

An online supplement to this article provides the detailed evaluation of experimental results.

ACKNOWLEDGMENT

The authors appreciate all anonymous reviewers whose comments significantly improved this paper. The research was supported by the National Science Foundation of China (61003095), ANR-NSFC (60911130368), Chinese 973 Program (2010CB328001), and National Institutes of Health (GM-075004).

REFERENCES

[1] H. Ling and D. Jacobs, "Shape classification using the inner-distance," *IEEE Trans. on PAMI*, vol. 29, no. 2, pp. 286–299, 2007.
[2] R. Gal, A. Shamir, and D. Cohen-Or, "Pose-oblivious shape signature," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 261–271, 2007.
[3] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Topology-invariant similarity of nonrigid shapes," *International Journal of Computer Vision*, vol. 81, no. 3, pp. 281–301, 2009.
[4] A. Elad and R. Kimmel, "On bending invariant signatures for surfaces," *IEEE Trans. on PAMI*, vol. 25, pp. 1285–1295, 2003.
[5] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs, "A search engine for 3D models," *ACM Transactions on Graphics*, vol. 22, no. 1, pp. 83–105, 2003.
[6] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani, "Three dimensional shape searching: State-of-the-art review and future trends," *Computer-Aided Design*, vol. 37, pp. 509–530, 2005.
[7] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 807–832, 2002.
[8] L. Sael, B. Li, D. La, Y. Fang, K. Ramani, R. Rustamov, and D. Kihara, "Fast protein tertiary structure retrieval based on global surface shape similarity," *Proteins*, vol. 72, no. 4, pp. 1259–1273, 2008.
[9] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel, "Analysis of two-dimensional non-rigid shapes," *International Journal of Computer Vision*, vol. 78, no. 1, pp. 67–88, 2008.
[10] A. M. Bronstein, M. M. Bronstein, R. Kimmel, M. Mahmoudi, and G. Sapiro, "A Gromov-Hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching," *International Journal of Computer Vision*, 2009, in press.
[11] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. Kunii, "Topology matching for fully automatic similarity estimation of 3D shapes," in *Proceedings of ACM SIGGRAPH*, 2001, pp. 203–212.
[12] V. Jain and H. Zhang, "A spectral approach to shape-based retrieval of articulated 3D models," *Computer-Aided Design*, vol. 39, no. 5, pp. 398–407, 2007.
[13] R. Rustamov, "Laplace-beltrami eigenfunctions for deformation invariant shape representation," in *Proceedings of the fifth Eurographics symposium on Geometry processing*, 2007.
[14] R. Rustamov, Y. Lipman, and T. Funkhouser, "Interior distance using barycentric coordinates," *Computer Graphics Forum (SGP'09)*, vol. 28, no. 5, pp. 1279–1288, 2009.
[15] K. Siddiqi, J. Zhang, D. Macrini, A. Shokoufandeh, S. Bouix, and S. Dickinson, "Retrieving articulated 3-D models using medial surfaces," *Machine Vision and Applications*, vol. 19, no. 4, pp. 261–275, 2008.
[16] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *Computer Graphics Forum (SGP'09)*, vol. 28, no. 5, pp. 1383–1392, 2009.
[17] M. Ovsjanikov, A. M. Bronstein, M. M. Bronstein, and L. J. Guibas, "Shapegoogle: A computer vision approach for invariant shape retrieval," in *Proc. NORDIA*, 2009.
[18] S. Biasotti, S. Marini, M. Spagnuolo, and B. Falcidieno, "Subpart correspondence by structural descriptors of 3D shapes," *Computer-Aided Design*, vol. 38, no. 9, pp. 1002–1019, 2006.
[19] T. Funkhouser and P. Shilane, "Partial matching of 3D shapes with priority-driven search," in *SGP'06*, 2006.
[20] A. M. Bronstein, M. Bronstein, and R. Kimmel, "Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching," *Proc. National Academy of Sciences (PNAS)*, vol. 103, no. 5, pp. 1168–1172, 2006.
[21] M. R. Ruggeri and D. Saupe, "Isometry-invariant matching of point set surfaces," in *Eurographics 3DOR*, 2008.
[22] A. B. Hamza and H. Krim, "Geodesic object representation and recognition," *Discrete Geometry for Computer Imagery, LNCS*, vol. 2886, pp. 378–387, 2003.
[23] H. Ling and D. Jacobs, "Using the inner-distance for classification of articulated shapes," in *CVPR'05*, 2005, pp. 286–299.
[24] S. Biswas, G. Aggarwal, and R. Chellappa, "Efficient indexing for articulation invariant shape matching and retrieval," in *CVPR'07*, 2007.
[25] R. Liu and H. Zhang, "Mesh segmentation via spectral embedding and contour analysis," *Computer Graphics Forum*, vol. 26, no. 3, pp. 385–394, 2007.
[26] Y.-S. Liu, Y. Fang, and K. Ramani, "IDSS: deformation invariant signatures for molecular shape comparison," *BMC Bioinformatics*, vol. 10, no. 157, 2009.
[27] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer, 2008.
[28] J. Tangelder and R. Veltkamp, "A survey of content based 3D shape retrieval methods," in *SMI'04*, 2004, pp. 145–156.
[29] A. Kaufman, D. Cohen, and R. Yagel, "Volume graphics," *IEEE Computer*, vol. 26, no. 7, pp. 51–64, 1993.
[30] B. Li, S. Turuvekere, M. Agrawal, D. La, K. Ramani, and D. Kihara, "Characterization of local geometry of protein surfaces with the visibility criterion," *Proteins*, vol. 71, no. 2, pp. 670–683, 2008.
[31] D. Raviv, M. Bronstein, A. M. Bronstein, and R. Kimmel, "Volumetric heat kernel signatures," in *Proc. 3DOR*, 2010.
[32] R. Rustamov, "Robust volumetric shape descriptor," in *Eurographics 3DOR*, 2010.
[33] T. Ju, M. Baker, and W. Chiu, "Computing a family of skeletons of volumetric models for shape description," *Computer-Aided Design*, vol. 39, no. 5, pp. 352–360, 2007.
[34] T. Ju, "Robust repair of polygonal models," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 888–895, 2004.
[35] P. Soille, *Morphological Image Analysis*. Springer, 1999.
[36] D. Coeurjolly, S. Miguet, and L. Tougne, "2D and 3D visibility in discrete geometry: An application to discrete geodesic paths," *Pattern Recognition Letters*, vol. 25, no. 5, pp. 561–570, 2004.
[37] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. on PAMI*, vol. 24, pp. 881–892, 2002.
[38] M. Pauly, M. Gross, and L. Kobbelt, "Efficient simplification of point-sampled surfaces," in *Proceedings of IEEE Visualization'02*, 2002, pp. 163–170.
[39] G. Schaufler and H. Jensen, "Ray tracing point sampled geometry," in *Proceedings of the 11th Eurographics Workshop on Rendering*, 2000, pp. 319–328.
[40] A. Adamson and M. Alexa, "Ray tracing point set surfaces," in *SMI'03*, 2003, pp. 272–279.
[41] Y.-S. Liu, J.-H. Yong, H. Zhang, D.-M. Yan, and J.-G. Sun, "A quasi-Monte Carlo method for computing areas of point-sampled surfaces," *Computer-Aided Design*, vol. 38, no. 1, pp. 55–68, 2006.
[42] B. Adams and P. Dutré, "Interactive boolean operations on surfel-bounded solids," in *Proceedings of SIGGRAPH'03*, 2003, pp. 651–656.
[43] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[44] P. Daras, D. Zarpalas, A. Axenopoulos, D. Tzovaras, and M. Strintzis, "Three-dimensional shape-structure comparison method for protein classification," *IEEE/ACM Trans Comput Biol Bioinform*, vol. 3, no. 3, pp. 193–207, 2006.
[45] S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani, "Developing an engineering shape benchmark for CAD models," *Computer-Aided Design*, vol. 38, no. 9, pp. 939–953, 2006.
[46] R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, pp. 5–30, 2006.
[47] M. Mahmoudi and G. Sapiro, "Three-dimensional point cloud recognition via distributions of geometric distances," *Graphical Models*, vol. 71, no. 1, pp. 22–31, 2009.