Computer-Aided Design 41 (2009) 293–305

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Robust principal axes determination for point-based shapes using least median of squares

Yu-Shen Liu [a,*], Karthik Ramani [a,b]

[a] School of Mechanical Engineering, Purdue University, West Lafayette, IN, 47907, USA

[b] School of Electrical Computer Engineering (by courtesy), Purdue University, West Lafayette, IN, 47907, USA

ARTICLE INFO

Article history:
Received 6 March 2008
Accepted 15 October 2008

Keywords:
Principal axes
Point-based shapes
Least median of squares (LMS)
Forward search
Principal component analysis (PCA)

ABSTRACT

A robust technique for determining the principal axes of a 3D shape represented by a point set, possibly with noise, is presented. We use techniques from robust statistics to guide the classical principal component analysis (PCA) computation. Our algorithm is based on a robust statistics method: least median of squares (LMS), for outlier detection. Using this method, an outlier-free major region of the shape is extracted, which ignores the effect on other minor regions regarded as the outliers of the shape.

In order to effectively approximate the LMS optimization, the forward search technique is utilized. We start from a small outlier-free subset robustly chosen as the major region, where an octree is used for accelerating computation. Then the region is iteratively increased by adding samples at a time. Finally, by treating the points on minor regions as outliers, we are able to define the principal axes of the shape as one of the major region. One of the advantages of our algorithm is that it automatically disregards outliers and distinguishes the shape as the major and minor regions during the principal axes determination without any extra segmentation procedure. The presented algorithm is simple and effective and gives good results for point-based shapes. The application on shape alignment is considered for demonstration purpose.

1. Introduction

One of the important tasks in computer graphics and computer vision is the determination of *location* and *orientation* of an object within a specified frame of reference [1,2]. Typically, this information is also called the *pose* of the object. There are several representations of an object's pose, including the principal axes, affine transformation, moment invariants, medial axis transform, and others [1]. The simplest and most widely accepted one for 3D shapes is based on the principal axes of the object [1], which completely consists of its orientation and position with respect to an orthogonal frame or coordinate system. One main problem using traditional global techniques is that the derived principal axes might be quite different for some similar shapes [3,4].

In computer graphics and geometric modeling, 3D shapes are commonly represented by explicit surfaces, implicit surfaces, or polygonal meshes. With 3D scanners becoming the standard source for geometric data acquisition, point sets have received an increasing attention as an alternative shape representation [5–11]. The point-based representation allows more flexibility when a globally consistent surface topology is not necessarily required [8], and can cover wide shapes, such as non-manifold. A geometric shape referred to in this paper is represented by a set of points sampled from its underlying surfaces. Our method presented in this paper is also suitable for other shape representations through converting them into point sets.

In this paper, we focus on the problem of the principal axes determination for 3D shapes represented by point sets, possibly with noise, by combining robust statistics methods.

1.1. Previous work

The principal axes or pose determination of 3D shapes is one of the most important techniques in robotics, computer graphics, and computer vision. It is used in many areas, such as shape alignment [12], object recognition [13], generating 2D drawing views from 3D models [14]. There have been some effective approaches in extracting the pose of both polyhedral and smooth objects by associating some additional information, such as the surface normals and areas, except the point position. These approaches include extended Gaussian image (EGI) [13], complex extended Gaussian image (CEGI) [2], and some related improvement techniques [15,14]. A review of many available methods for both polyhedral and smooth objects is beyond the scope of this paper. The reader may consult Refs. [2,15,14] for detailed expositions. In this section, the work most related to ours will be reviewed.

* Corresponding author. Tel.: +1 765 494 0309; fax: +1 765 494 0539.
E-mail addresses: liuyushen00@gmail.com, liu28@purdue.edu (Y.-S. Liu), ramani@purdue.edu (K. Ramani).

0010-4485/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2008.10.012

Research on determining the principal axes of 3D shapes can be categorized as the global and partial approaches. The most common technique for determining the principal axes of 3D shapes is principal component analysis (PCA) that is a global approach. The main advantages of PCA are simple, fast, and applicable to most of 3D models, also including non-manifold and point sets. Each object is analyzed by PCA in three principal axes (or eigenvectors), and according to their eigenvalues these vectors are mapped to three axes. PCA is generally considered to work well for a variety of classes. The PCA approach seeks a projection that best represents the data in a least squares sense [16]. One problem for the least squares method is lack of robustness. Indeed, one single outlier can have an arbitrarily large effect [17]. The principal axes derived by PCA might be quite different for some similar shapes due to some small local difference between shapes [3]. Our main goal in this paper is to ignore the effect on the minor regions of 3D shapes during the principal axes computation.

More recently, Passalis et al. [12] introduced a method for improving PCA by combining the symmetry planes of a 3D model. Their method relies on the assumption that most of real life objects are symmetrical with respect to a plane. First, a symmetry plane of the model is determined using a global optimization technique. Then the perpendicular vector to the symmetry plane is used as the first axis, and the remaining two axes are determined by projecting all vertices of the model onto the plane of symmetry and performing a 2D PCA. This strategy actually reduces 3D into 2D by projecting 3D points into the symmetry plane, but for the 2D case there is still the same problem as 3D using PCA. Certainly, for classes of objects with more symmetry planes, their method can completely eliminate PCA by seeking the primary and secondary symmetry planes. However, most 3D life objects contain deformation, so multiple symmetry planes are difficult to be found. In addition, the determination of symmetry is also a non-trivial task [18,19].

In general, the partial approaches first segment the model into some patches, and then the patches on major regions are used for the final pose computation. Recently, Gal and Cohen-Or [4] presented a partial shape matching method for shape alignment. Their method first segments the models into some salient geometric features, and then searches for matching between pairs of salient features from each model. This partial method can produce better results than those using PCA applied to the whole model. However, segmentation is a non-trivial task itself. It often fails to segment surfaces sampled from non-manifold or large and complex point sets acquired by 3D scanning devices, and consequently results in an unsuccessful pose determination. Furthermore, both segmentation and partial matching require the expenditure of large amounts of time and space if the number of points and patches is gigantic.

Another class of approaches focuses on topological or graph comparison for determining the pose. The topology-based approaches rely on the fact that 3D model topology is an important shape characteristic. For instance, Hilaga et al. [20] proposed the use of the skeleton of an object for retrieving. A drawback of topological approaches is that relatively small anomalies on the object's surface can have a significant impact on the topological properties of the object [12]. In addition, the topological approaches also have some problems similar to partial shape matching methods for non-manifold and complex objects.

1.2. Robust statistics methods

Determining the principal axes of point sets using PCA is similar to the linear regression using least squares [16]. The linear regression belonging to a statistical method is considered to be *robust* if it has a large *breakdown* point. A breakdown point might be loosely defined as the smallest percentage of outliers that can cause the estimator to take an arbitrarily large aberrant values [17, 7]. For instance, the breakdown point of the median of a set of values is 50% [17], whereas least squares has a breakdown point of 0%.

Robust statistics methods have been applied to various computer vision applications [21]. For example, Torre and Black [22] proposed a robust PCA approach for automatic learning of linear models from data that may be contaminated by outliers. However, there is little attention on 3D computer graphics. Jones et al. [23] and Fleishman et al. [24] have applied the bilateral filter to mesh denoising, which can be considered as a robust statistics technique. Pauly et al. [25] introduced a method for analyzing the uncertainty and variability of a point set. Their method can be regarded as a *backward* method that can not detect *masked outliers* [7]. Masked outliers are outliers that can not be identified from the statistics of a model that is dealt with to the entire sample set. The strategy of backward methods for fitting a model first fits a model to the entire sample set and then tries to delete bad samples. The reader may consult Ref. [7] for other several works related to robust statistics methods in computer graphics.

Fleishman et al. [7] recently presented a new robust fitting method from a set of points in order to overcome the drawback of backward methods. The main tool that they use is the *forward* search algorithm which has a significant advantage in detecting outliers over commonly used backward methods. The main strategy of their algorithm is as follows. A subset of the data is first fitted using the second degree polynomials based on the forward search algorithm, and then the rest of the data is identified as outliers. To fit multiple surfaces, the above procedure is repeated for the remaining point sets. Our work presented in this paper is in the same spirit and applies the forward search to the principal axes determination.

1.3. Contributions

The work most related to ours is Fleishman et al.'s work [7]. From statistical view point, the method in [7] treats the points across the discontinuities as outliers in order to define sharp features. Unlike their method for defining a surface from a point cloud, our goal is to determine the principal axes of 3D shapes. Our work is based on a powerful statistic technique, called least median of squares (LMS) [17,26], to improve the PCA limitations. Using this method, an outlier-free major region of a 3D shape is extracted, which ignores the effect on other minor regions regarded as the outliers of the shape. At the last phase of our algorithm, the principal axes of the major region are regarded as the final ones of the shape. In order to effectively approximate the LMS optimization, the forward search technique [27] is utilized. The basic idea in forward search is to start from a small subset of robustly chosen samples of the data that excludes outliers. To accelerate the extracting of the initial subset, we exploit the octree for approximating the shape and sampling points. Then the principal axes are computed iteratively by adding one sample into the subset at a time while monitoring certain statistical estimates. We can use the method to deal with noise, outliers, and minor regions on shapes. The presented algorithm is simple and effective and can give good results for point-based 3D shapes. The application on shape alignment is considered for demonstration purpose. The main contributions of our work can be summarized as follows:

– Propose a new robust technique for principal axes determination by guiding the classical PCA computation based on least median of squares and the forward search technique. The proposed algorithm automatically disregards outliers and distinguishes the shape as the major and minor regions during the

principal axes determination. Our algorithm can be applied to large and complex point sets acquired by 3D scanning devices or sampled from multi-surfaces (or non-manifold).
- The octree-based approximation and point sampling methods accelerate the extraction of the initial subset in forward search.
- Apply our algorithm to some shape matching techniques, such as shape alignment.
- Investigate the effect on noise and sampling density and compare our method with previous works.

2. Robust estimation

In this paper, we consider the following input conditions. For this case in 3-dimensional space, let $\mathcal{C}_N = \{\mathbf{p}_i | i = 1, \ldots, N\}$ be a set of unorganized data points, where $\mathbf{p}_i = (x_i, y_i, z_i)^{\mathrm{T}}$ is a 3D vector. The set $\mathcal{C}_N$ of data points is assumed to be a sampling of underlying surfaces of a 3D shape with or without boundary. We suppose that the unorganized data points, often referred to a *point set*, *point clouds*, or *scattered data points* in the literature, may have non-uniform distribution with considerable noise.

The most commonly used technique for determining the principal axes of 3D shapes is the PCA technique. In this section, we will investigate the reasons and limitations on the application. Then a robust statistics method, i.e. least median of squares (LMS), will be introduced to overcome the PCA limitations. Finally, the forward search technique is utilized in order to effectively approximate the LMS optimization.

2.1. Review of PCA

In statistics, PCA is a technique for simplifying a data set by reducing multi-dimensional data sets to lower dimensions for analysis. The basic idea of PCA is to seek a projection that best represents the data in a least squares sense [16]. We first review this procedure by considering a 3D shape represented by a point set $\mathcal{C}_N$. The specified reference frame for the 3D shape consists of an origin and three principal axes. PCA assumes the origin is at the sample mean point $\mathbf{o}$. PCA wants to find a vector $\mathbf{e}$ through the origin $\mathbf{o}$ such that the sum of the squared distances between various $\mathbf{p}_i \in \mathcal{C}_N$ and the corresponding projection point $\mathbf{p}_i^*$ onto $\mathbf{e}$ is as small as possible. The squared-error criterion function is defined by

$$J(\mathbf{p}_1^*, \ldots, \mathbf{p}_N^*, \mathbf{e}) = \sum_{i=1}^{N} \|\mathbf{p}_i^* - \mathbf{p}_i\|^2. \tag{1}$$

Let $\mathbf{e}$ be a unit vector. Then the equation of the line through the origin $\mathbf{o}$ in the direction of $\mathbf{e}$ can be written as $\mathbf{x} = \mathbf{o} + \alpha\mathbf{e}$, where the scalar α corresponds to the distance of any $\mathbf{x}$ from the mean $\mathbf{o}$. If each projection point $\mathbf{p}_i^*$ of $\mathbf{p}_i \in \mathcal{C}_N$ onto the line is represented by

$$\mathbf{p}_i^* = \mathbf{p}_i^*(\alpha_i) = \mathbf{o} + \alpha_i\mathbf{e}, \quad \alpha_i \in \mathbb{R}. \tag{2}$$

We can find an optimal set of coefficients α_i by substituting Eq. (2) into Eq. (1) and minimizing the squared-error criterion function

$$J(\alpha_1, \ldots, \alpha_N, \mathbf{e}) = \sum_{i=1}^{N} \|(\mathbf{o} + \alpha_i\mathbf{e}) - \mathbf{p}_i\|^2. \tag{3}$$

By adding the constrain $\|\mathbf{e}\| = 1$ and setting the derivative to zero for finding the best direction $\mathbf{e}$, the solution to this problem involves the so-called *covariance matrix* $\mathbf{A}$ [16] defined by

$$\mathbf{A} = \sum_{i=1}^{N} (\mathbf{p}_i - \mathbf{o})(\mathbf{p}_i - \mathbf{o})^{\mathrm{T}}. \tag{4}$$

The eigenvector corresponding to the largest eigenvalue of the covariance matrix $\mathbf{A}$ is the first principal axis $\mathbf{e}$.

In fact, PCA is similar to the linear regression in a least-squares sense. However, a single sample with a large error, an *outlier*, can change the principal axes arbitrarily. This results in that the derived principal axes might be quite different for some similar shapes [3,4].

2.2. Least median of squares

To overcome the lack of robustness using least squares in Eq. (1), some robust methods might be used for improving PCA, such as making use of some weight functions for bounding the influence of outliers. However, most robust methods are *least sum of squares* by replacing the square by something else, and they can not raise a high breakdown point [17].

In our case, we assume that a 3D shape represented by a point set $\mathcal{C}_N$ consists of two parts: a *major* region and the remaining *minor* regions, and there is no overlap between them. The major region is expected to contain at least 50% points of the entire point set, so the remaining minor regions have up to 50% points. In our work, not only the noise but also the minor regions are considered as outliers for determining the principal axes of the point-based shape. Our motivation is to improve the least sum of squares in PCA with a high breakdown point (up to 50%). This above assumption, i.e. the major and minor regions making up of the 3D shape, is similar to the partial shape alignment [4], in which the major region is defined by the set of some salient features and the remaining can be considered as minor regions.

The *least median of squares* (LMS) is a robust regression method that estimates the parameters of the model by minimizing the median of the absolute *residuals*. In other words, LMS replaces the sum of least squares by a median. LMS satisfies a 50% breakdown point [17]. The resulting estimator using LMS can resist the effect of nearly 50% of contamination in the input data, which is applicable to our case. In our case, we define the absolute residual as the distance between the test point $\mathbf{p}_i$ and the projection point $\mathbf{p}_i^*$ onto the first principal axis through the origin: for the ith point $r_i = \|\mathbf{p}_i^* - \mathbf{p}_i\| = \|(\mathbf{o} + \alpha_i\mathbf{e}) - \mathbf{p}_i\|$. In this paper, we search a best direction $\mathbf{e}$ that minimizes the median of the residuals as follows:

$$\min_{\mathbf{e}} \ \underset{i}{\text{median}} \ \|(\mathbf{o} + \alpha_i\mathbf{e}) - \mathbf{p}_i\|, \tag{5}$$

where $\mathbf{e}$ is the first principal axis that will be computed. Rousseeuw [17] has also pointed out there always exists a solution for LMS.

Eq. (5) can be solved using the following random sampling algorithm (i.e. RANSAC) [7,28]. First, k points are selected at random, and the first principal axis is computed using the standard PCA algorithm to the points. Next the median of the residuals of the remaining $N - k$ points is computed. The process is repeated T times to generate T candidate axes. The axis with the minimal median is selected as the final principal axis $\mathbf{e}$. For the remaining two principal axes, we might use the similar strategy acquired by projecting all points onto the plane perpendicular to $\mathbf{e}$ and through $\mathbf{o}$ and performing a 2D resolution for Eq. (5). A small value of k does not use all of the available points to PCA computation, while a larger value of k requires more iterations. If k is too large, the algorithm becomes sensitive to outliers including noise and minor regions.

2.3. The forward search algorithm

The forward search algorithm [27] is a robust method that avoids the need to fix k. Fleishman et al. [7] applied this technique to reconstruct surfaces from point clouds. The forward algorithm first searches a small outlier-free subset and then iteratively refines

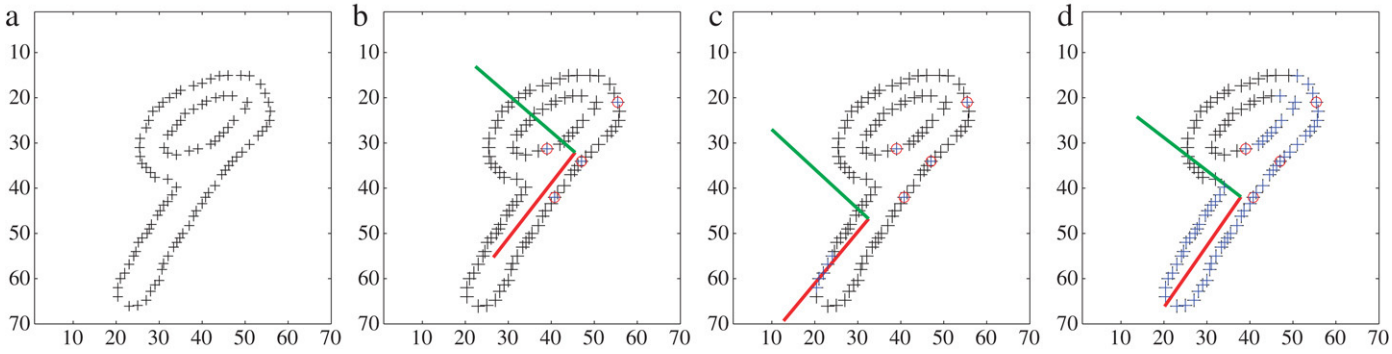

Fig. 1. The illustration of determining the principal axes with the forward search technique. (a) The input data points sampled from the contour of a handwritten digit "9". (b) First, determine robustly the principal axes to a small subset (4 red points) using PCA, where the red and green lines are the first and second principal axes, respectively. (c) Next, add points with smallest residual (blue points) into the subset and recompute PCA to the updated subset, where the result after five iterations is shown. (d) The final principal axes of the forward search is shown. In (d), the remaining points (black points) are regarded as outliers or minor regions, and the final principal axes are defined using the major regions (blue and red points). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the subset by adding one sample at a time. This is in contrast to the backward algorithms, which first deal with the entire data points and then delete bad samples. Fleishman et al. [7] show that some outliers with a large breakdown point usually fail on fitting based on the backward algorithms, whereas the forward algorithm will give satisfactory results. The initial model is computed for Eq. (5) using the LMS method with a small k value, typically k close to p for a model with p parameters [27,7] ($p = 3$ in the 3D case). In our implementation, we choose $k = 4$.

During the forward search, a number of parameters can be monitored to detect the influential points. Typically, the forward search will add the good-samples first and only when these are exhausted, outliers will be added. Atkinson et al. [27] suggested several statistics, including the residual-plot, Cook's distance and others, to be monitored. For their purposes, these are plotted on a graph and inspected visually. The maximal residual $r_{\max}$ is monitored by Fleishman et al. [7]. In our technique, we also monitor the maximal residual similar to Fleishman et al.'s strategy [7]. However, we compute $r_{\max}$ based on the initial subset. It will be discussed in Section 3.3.

Using the forward search technique for solving Eq. (5), we present the main procedure of determining the principal axes of a point set $\mathcal{C}_N$ as follows:

1. Choose a small outlier-free subset Q using LMS.
2. The principal axes and the origin are computed using PCA to Q.
3. The point with the lowest residual in the remaining points is added into Q.
4. Repeat steps 2 and 3 until the error is larger than a predefined threshold $r_{\max}$ and identify the points in $\mathcal{C}_N - Q$ as outliers.

Fig. 1 shows an example of this process in two dimensions (see caption). The process is simple, but there are some limitations for effective computation for large and complex point sets acquired by 3D scanning devices. We will take advantage of the octree and point sampling to significantly accelerate the process and improve its stability. The overall algorithm will be introduced in the next section.

3. Robust principal axes determination

3.1. Initial robust estimator

In the first step of the forward search algorithm, the initial subset is computed using the LMS algorithm with a small k value (see Section 2.3). If the number N of points in $\mathcal{C}_N$ is small, the choice of the initial subset can be performed by exhaustive enumeration of all $\binom{N}{k}$; otherwise, the LMS uses the RANSAC algorithm (see Section 2.2) that requires a large iteration number T to achieve a high probability of finding a good estimator. The LMS, as a statistical method, assumes that the samples (points) are independent. If g is the probability of selecting a single good sample at random from the original point set $\mathcal{C}_N$, then the probability P of successfully finding k good samples after T iterations can be computed by $P = 1 - (1 - g^k)^T$ [7]. Furthermore, for every iteration, LMS requires a sort of the residuals of the remaining $N - k$ points to find their median, so this will also require the expenditure of large amounts of time and space for sorting for a large T if N is gigantic. In general, there is a large number of points for a 3D shape acquired by 3D scanning devices. For instance, Fig. 2 shows a 3D shape with 23,400 points, and the running time for finding an initial subset ($k = 4$) is about 401.8 s with $T = 5000$ iterations, where the process of computing and sorting the remaining residuals is repeated 5000 times.

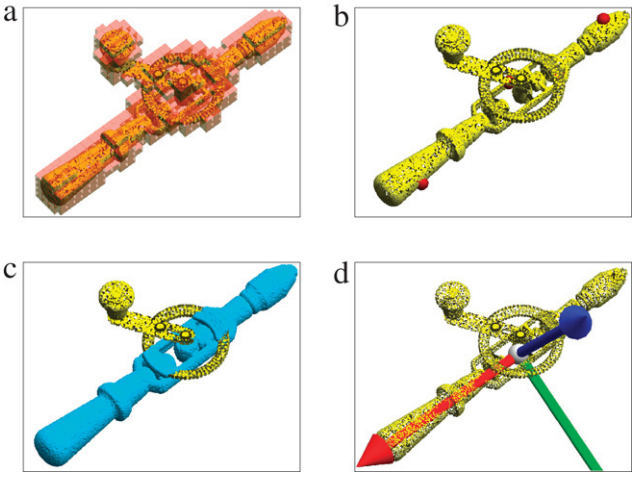

Fig. 2. The illustration of robust principal axes determination for a point set representing the drill hand model with 23,400 points. (a) First, an octree of depth $d = 5$ is constructed with. (b) Next, the initial subset with 4 points (red) is chosen after 5000 iterations. (c) The final major region (blue points) is shown using the forward search technique. (d) The final principal axes are determined using the points on the major region, In this paper, the red, green, and blue axes correspond to the first, second, and third principal axes, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.1.1. Octree-based approximation

In this phase, we use octrees described by Adams et al. [29] to accelerate the initial subset searching, where the points of $\mathcal{C}_N$

are considered as surfels with zero radius. In some point-based processing, such as Boolean operations [29], surface reconstruction [30], and area computation of point-based models [9], the octree can be utilized. Thus, it is not an additional price for the point set. In the preprocessing step, an axis-aligned octree of depth d [29] can be constructed. Adams et al. [29] suggested that $d = 4$ or 5 can lead to both a small approximation error for shapes and little computation time. We typically choose $d = 5$ in our implementation.

Suppose that we have constructed an octree for $\mathcal{C}_N$, and have classified the cells of the octree as two types: boundary cells containing points of $\mathcal{C}_N$, and empty cells containing no point of $\mathcal{C}_N$ [29]. For each boundary cell of the octree, we compute a center of points in the boundary cell as a *feature* point that characterizes all points inside the boundary cell. Let $\mathcal{F}$ be the set of feature points for all boundary cells of the octree. For every iteration, k points are first selected at random from the original point set $\mathcal{C}_N$, and three principal axis are determined using PCA. Let $\mathbf{e}$ be the first principal axis obtained using PCA. Next the residuals are computed by projecting feature points in $\mathcal{F}$ onto $\mathbf{e}$. Unlike the original LMS method, we compute the median of the residuals of feature points in $\mathcal{F}$ instead of ones of $N - k$ points in $\mathcal{C}_N$. The octree-based approximation yields both the good approximation results and little computation time. Some other tree data structure can also be used for approximating the point set, such as B-Trees [11].

3.1.2. Octree-based point sampling

In the first step of the forward search algorithm, k points are selected at random for every iteration for computing three principal axes. Point sampling is an intermediate step for a variety of computer graphics applications, and specialized sampling strategies have been developed to satisfy the requirements of each problem (such as Refs. [31–33]). A simplest sampling strategy is to choose k points randomly from N points in $\mathcal{C}_N$ using pseudo-random number generators, but it might occur the chance of point clustering. For example, k points appear on the same boundary cell of the octree of $\mathcal{C}_N$, which results in an invalid sample. Plenty of invalid samples will result in a low convergence rate.

Instead, our method for generating unbiased random points with respect to the number of points in each boundary cell proceeds as follows. First, for each boundary cell, we count the number of points inside the boundary cell, which is also called the *density* of the boundary cell. Then, we store the density of each boundary cell in an array along with the cumulative density of boundary cells visited so far. Next, k boundary cells are selected with probability proportional to their densities. This procedure is finished by generating k random numbers between 0 and the total cumulative density and performing a binary search on the array of cumulative densities. For each selected boundary cell, one sample point is chosen randomly from the points inside the boundary cell. Intuitively, the above sampling method gives k uniform random points with respect to the density of the boundary cells. Our idea is similar to the area-based sampling strategy presented by Osada et al. [33]. They uniformly sample points from a triangle mesh with probability proportional to the area of triangles.

We find that the sampling based on octree can offer a faster convergence rate than one using the direct sampling from points of the original model. Of course, other sampling methods that sample according to curvature or other surface properties would be possible as well. However, these methods usually require the large expense of computation time, and this is not applicable to fast computation in our applications.

Fig. 2(a) illustrates an octree of depth $d = 5$ for a point-based shape, where the octree has 499 boundary cells that are displayed. In Fig. 2(b), a good initial subset with $k = 4$ points (red) is obtained with $T = 5000$ iterations using the above octree-based approximation and point sampling methods, and its running time is about 1.822808 s.

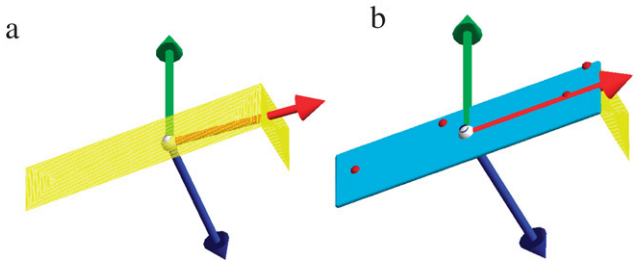

Fig. 3. Determining principal axes for point sets sampled from surface patches. The input is a wedge data with 14,687 points sampled from two planes. (a) PCA. (b) Our method. Here four red points are the initial subset and blue points are major region. Note that the points sampled from the small plane do not effect our method unlike PCA. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.2. Residual computation

In Section 2.2, we defined the residual as the distance between the test point $\mathbf{p}_i$ and the corresponding projection point $\mathbf{p}_i^*$. Eq. (5) minimizing the median of projection distance onto the first principal axis is an extension of PCA. In the special 2D case, LMS corresponds to finding the narrowest "strip" covering half of the points [17]. In our case, the optimization in Eq. (5) also implies that the major region of 3D shapes is nearly a *cylinder-like* shape surrounding the first principal axis. The assumption can work well for most mechanical parts (such as pipeline and bearing) and real life objects (such as human, animal, and tree). However, there are still some limitations for point sets sampled from surface patches. The cylinder-like shape is not expected to be similar for most surface patches whose major regions are near planes. To overcome the disadvantage for surface patches using LMS, we might redefine the residual as the distance between the test point $\mathbf{p}_i$ and the projection $\mathbf{p}_i^*$ onto the reference plane defined by the first and second principal axes. Based on the new residual definition, the LMS optimization can be implemented by projecting all points onto the reference plane, which is determined by two principal axes using PCA, at every iteration. Fig. 3 shows an examples for determining the principal axes for a point set sampled from surface patches using the new residual definition. Of course, some other residuals can also be defined for being suitable for some special style of shapes. For some shapes without obvious major regions, our algorithm is also incapable as the standard PCA.

3.3. Forward search on point sets

After a robust estimator is computed for a small number of points using the above algorithm, the results are three reference principal axes and an initial outlier-free subset Q. The threshold of maximal tolerated residual $r_{\max}$ is computed using Q as follows. The k points in Q are projected onto the first principal axis (or a reference plane in Section 3.2). Suppose that r_t is the maximal residual of k points. We define $r_{\max}$ to be proportional to r_t

$$r_{\max} = \lambda r_t, \tag{6}$$

where λ is the value of residual band. We typically use $\lambda = 1.25$. A small value of λ does not use all of the available samples to determine the major region, while a larger value for λ requires more iterations. The second step of the forward search is to iteratively add one point with lowest residual to the set Q at each iteration by sorting the residuals of the remaining points. The iteration is terminated until the lowest residual is larger than $r_{\max}$.

Since we expect a less iteration and little computation time for the forward search, instead of only adding one point at each iteration, we add more points every time. The number m of adding points can be set by users. We typically choose $m = 60$ or more for the dense point sets. Fig. 2 illustrates the procedure of robust principal axes determination for a 3D shape.

Algorithm 1 : **RobustPrincipalAxes**($\mathcal{C}_N$, **o**, $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, Q)

Input:
 $\mathcal{C}_N \in \mathbb{R}^{3\times N}$: the given point set with N points
Output:
 o: the origin of the reference frame
 $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$: the first, second, and third principal axes
 $Q \subseteq \mathcal{C}_N$: the working subset, i.e. the major region
Local variables:
 k: the number of random samples
 $r_{\max}$: the maximal tolerated residual
 I: the current iteration
 $c_{\text{rem}} \subseteq \mathcal{C}_N$: the set of the remaining points
 m: the number of points added into Q at every iteration
 MAX_ITERS: the maximal number of iterations
begin
1: $Q \Leftarrow \varnothing$;
2: **LMS**($\mathcal{C}_N$, k, Q);
3: Compute $r_{\max}$ using Q via Eq. (6);
4: $I \Leftarrow 0$;
5: **while** ($I++ <$ MAX_ITERS) **do**
6: Compute the origin **o** and three principal axes: $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ for Q using PCA;
7: $c_{\text{rem}} \Leftarrow \mathcal{C}_N - Q$;
8: Compute the residual as the distance between each point in c_{rem} and its projection point onto the line through **o** in the direction of $\mathbf{e}_1$;
9: Get m points with lowest residuals for c_{rem};
10: **if** (the maximal residual of the m points $> r_{\max}$) **then**
11: The points whose residuals are smaller than $r_{\max}$ are added into Q;
12: **return**
13: **end if**
14: Add the m points into Q;
15: **end while**
16: Project $\mathcal{C}_N$ onto the plane determined by $\mathbf{e}_2$, $\mathbf{e}_3$ and **o**, then perform a 2D robust PCA for updating the principal axis $\mathbf{e}_2$ and $\mathbf{e}_3$;
end

3.4. The algorithm implementation

The outline of an algorithm for robust principal axes determination, called **RobustPrincipalAxes**, is given in Algorithm 1. The algorithm takes as input a point set $\mathcal{C}_N$ and computes the origin **o** and three principal axes $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ of the reference frame. This is achieved through an iterative procedure with the aid of a variable Q which is a working subset of $\mathcal{C}_N$. Initially, Q is computed using the LMS algorithm through selecting k points at random with T iterations, as illustrated in Algorithm 2. Here, the LMS procedure is accelerated for approximating and sampling using an octree of depth d for $\mathcal{C}_N$. The octree can also be constructed in the preprocessing step.

According to Algorithm 2, the proposed algorithm is passed into a point set ($c_n \Leftarrow \mathcal{C}_N$), and a loop with T iterations begins. At each iteration, a subset c_{temp} with k points is randomly selected using the octree-based sampling, and the first principal axis $\mathbf{e}_1$ and the origin **o** is computed for c_{temp} using the classical PCA. Then, the residuals of c_n are calculated as the distance between the feature point of each cell and its projection point onto the line through **o** and $\mathbf{e}_1$. Next, the median r_{half} of the residuals is obtained. If r_{half} is less than the minimal residual $r_{\min}$, $r_{\min}$ and Q are updated as r_{half} and c_{temp}, respectively.

During the iterative procedure in Algorithm 1, the cardinality of Q is gradually increased by adding m points with lowest residuals every time. In this way, one is able to increase Q regarded as the major region in the forward search. If the residuals of the remaining points ($\mathcal{C}_N - Q$) are more than a threshold $r_{\max}$, the procedure is terminated. Finally, the points in the major region Q are used to compute the first principal axis $\mathbf{e}_1$, and the remaining points are identified as outliers or minor regions.

Algorithm 2 : **LMS**(c_n, k, Q)

Input:
 $c_n \in \mathbb{R}^{3\times n}$: the working point set with n points
 k: the number of random samples
Output:
 $Q \subseteq c_n$: the initial subset
Local variables:
 d: the depth of the octree for c_n
 T: the number of iterations
 $c_{\text{temp}} \subseteq c_n$: the initial subset
 o: the origin of the reference frame
 $\mathbf{e}_1$: the first principal axis of the reference frame
 r_{half}: the median of residuals
 $r_{\min}$: the minimal residual
begin
1: Construct the octree of depth d for c_n; /* *the octree can also be constructed in the preprocessing step* */
2: $r_{\min} \Leftarrow \infty$;
3: **for** ($i = 0; i < T; i++$) **do**
4: Select randomly a subset c_{temp} with k points using the octree-based sampling;
5: Compute the first principal axis $\mathbf{e}_1$ and the origin **o** for c_{temp} using PCA;
6: **for** (boundary cells of the octree of c_n) **do**
7: Compute the residual as the distance between the feature point of each cell and its projection point onto the line through **o** in the direction of $\mathbf{e}_1$;
8: **end for**
9: Compute the median r_{half} by sorting the residuals;
10: **if** ($r_{\text{half}} < r_{\min}$) **then**
11: $r_{\min} \Leftarrow r_{\text{half}}$;
12: $Q \Leftarrow c_{\text{temp}}$;
13: **end if**
14: **end for**
end

Algorithm 1 is mainly used to compute the first principal axis $\mathbf{e}_1$. The remaining two principal axes might be computed using the strategy similar to Ref. [12] by projecting all points of $\mathcal{C}_N$ onto the plane perpendicular to $\mathbf{e}_1$ and through the origin **o**. Then a 2D case for Algorithm 1 is performed for computing $\mathbf{e}_2$ and $\mathbf{e}_3$.

4. Results and applications

We have implemented the technique presented in the previous section and tested it on a large number of different point-based 3D shapes. The algorithm described above is implemented in C++. In this paper, the execution time is given in seconds on a Pentium IV 1.70 GHz processor with 512M RAM excluding the time of loading point sets.

Before computing the principal axes, a preprocessing step includes an octree construction that can be performed in a short time. Table 1 gives the time in seconds for some point-based shapes referred to in this paper, where "N" is the number of points of the models, "Major%" is the percentage of the major region that belongs to the original point set, "m" is the number of points added at each iteration (see Section 3.3), "T1" and "T2" are the time of constructing an octree and computing the principal axes. The computation time increases with the number of points of the models used. The time also depends on the size of major region

Fig. 4. Testing the major regions (blue points) of similar models using our method. (a) One dinosaur with different poses. (b) One horse with different poses. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Results of the **RobustPrincipalAxes** algorithm for some point-based shapes.

Model	Fig.	N	Major%	m	T1[a](s)	T2[b](s)
Drill	2	23,400	74.4	60	0.07	9.32
Wedge	3(b)	14,687	80.3	60	0.06	3.89
Gun	6(a)	15,412	66.5	60	0.06	3.42
Cat	9	17,633	71.4	60	0.07	7.91
Dancer	5(a)	75,206	63.8	300	0.42	16.4
Neptune	5(b)	112,220	51.6	300	0.28	21.3

[a] T1 is the time of constructing an octree at depth 5.
[b] T2 is the time of computing the principal axes.

and the number of iterations. In all examples, we use $T = 5000$ iterations in Algorithm 2.

4.1. Testing major regions for articulated and scanned models

One important issue of principal axes computation using our method is its robustness of computing the major regions for different poses of the same object. We test the 3D models with multiple poses in Gal et al.'s database [34], which contains some similar articulated models with multiple poses. All models in this database are represented by triangled meshes, so we first sample the triangled meshes to generate the point sets as the tested models. Then our method is applied to the point-based models for obtaining the major regions. Most of real life objects in this database, such as animal models, contain the similar major region with a cylinder-like shape. Fig. 4 shows some results of major regions for the dinosaur and horse models, respectively. Here, our method can recognize that the body of animal is the major region. The major regions computed by our method are insensitive to pose changes of the same object.

In addition, some real point sets acquired by 3D scanning devices are also tested for computing the major regions and the final principal axes (see Fig. 5). These selected models are obtained from AIM@SHAPE Shape Repository. Fig. 5(a) shows the result of a dancer model. In this example, the computed major region, which contains automatically the portions: one leg, the head and hands, and part of body, captures the main pose of the dancer. In Fig. 5(b), another complex example is given for the Neptune model. Although the lance and pedestal hold part of the model, the principal axes on the major region are still determined on the Neptune's body. In Fig. 5(c), the major region is captured as the body of the pig, not its legs. Our results show that our method works good on both scanned or articulated objects.

4.2. Shape alignment

Many shape matching methods require an alignment step in order to position all objects in a standard orientation [4,12]. These searching methods rely on the ability to align models by some global similarity transformation (rotation +uniform scale) that normalizes the models and establishes some correspondence between them. Most global alignment methods first determine a rigid-body transformation and a uniform scale, which align two models together as closely as possible, before measuring the distance between them. This is typically achieved by PCA applied to the whole model. The PCA method does not discriminate between the major regions, and can easily cause similar local features to be misaligned. Our alignment algorithm based on robust principal axes determination first searches the major region of each model. Then for each model, the origin and three principal axes are computed for the corresponding major region as a specified frame of reference. Finally, the transformation is applied to two reference frames for finishing the final alignment.

In Fig. 6, a 3D example is shown for aligning two gun models, where the gun in Fig. 6(b) is part of one in Fig. 6(a) only through removing the cartridge clip. Fig. 6(c) shows that a global PCA alignment fails to align them correctly, whereas our method correctly aligns as shown in Fig. 6(d). Comparison between two major regions for Fig. 6(a) and (b) is given in Fig. 7. Another example is shown in Fig. 8, where two point sets are sampled from the drills with different angle-rotation hands. Our alignment (see Fig. 8(d)) is better than a global PCA alignment (see Fig. 8(c)).

In addition, an animal example is shown in Fig. 9, where the same models appear in two different poses. In spite of the obvious difference between them, our algorithm aligns the two better than a global PCA alignment, where the body of the cat is automatically identified as the major region not its legs. Our alignment is based on a majority scheme that finds the transformation which satisfies the major regions not the whole shapes.

5. Discussion

In this section, we will discuss some parameters used in our algorithm, influences on noise and irregular samples, comparison with previous works, and limitations of our algorithm.

5.1. Parameters

In Algorithm 1 and 2, the parameters of the algorithms are: $r_{\max}$, T, m, and MAX_ITERS. In our implementation, the maximal