Computer-Aided Design 44 (2012) 537–553

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

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

Towards locally and globally shape-aware reverse 3D modeling

Manish Goyal [a], Sundar Murugappan [a], Cecil Piya [a], William Benjamin [a], Yi Fang [a], Min Liu [c], Karthik Ramani [a,b,*]

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

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

[c] Institute of Manufacturing Engineering, Tsinghua University, Beijing, 100084, China

ARTICLE INFO

Article history:
Received 15 December 2010
Accepted 17 December 2011

Keywords:
Reverse 3D modeling
Digital shape reconstruction
CAD model parameterization
Volumetric segmentation

ABSTRACT

The process of re-creating CAD models from actual physical parts, formally known as digital shape reconstruction (DSR) is an integral part of product development, especially in re-design. While, the majority of current methods used in DSR are surface-based, our overarching goal is to obtain direct parameterization of 3D meshes, by avoiding the actual segmentation of the mesh into different surfaces. As a first step towards reverse modeling physical parts, we extract (1) locally prominent cross-sections (PCS) from triangular meshes, and (2) organize and cluster them into sweep components, which form the basic building blocks of the re-created CAD model. In this paper, we introduce two new algorithms derived from Locally Linear Embedding (LLE) (Roweis and Sauk, 2000 [3]) and Affinity Propagation (AP) (Frey and Dueck, 2007 [4]) for organizing and clustering PCS. The LLE algorithm analyzes the cross-sections (PCS) using their geometric properties to build a global manifold in an embedded space. The AP algorithm, then clusters the local cross sections by propagating affinities among them in the embedded space to form different sweep components. We demonstrate the robustness and efficiency of the algorithms through many examples including actual laser-scanned (point cloud) mechanical parts.

1. Introduction

Digital shape reconstruction (DSR) in Computer Aided Design (CAD) is a process which involves extraction of high level parametric information from low level mesh or point cloud data. With the help of various CAD modelers available today, this high level parametric information is converted into surface parameterized models that can be modified or analyzed for further improvement and development. The initial complex and important step in DSR is segmentation of the mesh model [1].

Two major approaches that have been developed in segmenting a mesh model include surface-based and volume-based techniques (commonly known as feature based). Fig. 1 shows the difference between the two approaches. Both these approaches involve segmenting a mesh model into either surfaces or volumes. Digital model reconstruction through current surface-based segmentation methods does not lend itself to intuitive and flexible manipulation in contrast to what a parameterized CAD solid model does (see Fig. 1). Parameterized CAD models are associated with high level shape definition parameters such as radius, angle, width, and geometric constraints, while surface-based representations have low-level shape parameters such as knots, weights, and control points which are counter-intuitive to manipulation, especially for designers [2]. For example, from a functional design point of view, a digitally reconstructed model of an aerospace engine blade cannot embody original hydrodynamic properties without the parametric representation. Moreover, reconstructing a solid by stitching surfaces usually results in an inaccurate and inconsistent CAD model, and is also time consuming and laborious. The evolution of CAD modelers from surface to volume based design has led the volume based approach to gain more importance since volumetric or feature segmentation represents the design intent more closely and accurately.

Our overarching goal is to obtain direct parameterization of 3D meshes, by avoiding the actual segmentation of the mesh into different surfaces. Fig. 3 shows the difference between the traditional reverse engineering pipeline and our approach. As a first step towards reverse modeling physical parts, we extract (1) locally prominent cross-sections (PCS) from triangular meshes, and (2) organize and cluster them into sweep components. These sweep components form the basic building blocks in recreating a CAD model of the original object with user interaction.

We refer to the extracted cross-sections that are closed as 'Full Prominent Cross-Sections' (FPCS) and those that are open as 'Partial Prominent Cross-Sections' (PPCS). Feature intersection is addressed by the introduction of PPCS created in the regions of sweep intersections (red colored PCS in center model of Fig. 2). An individual set consists of a large number of uniformly distributed PCS, each of which approximates a local sweep in the small region

* Corresponding author at: School of Mechanical Engineering, Purdue University, West Lafayette, IN, 47907, USA. Tel.: +1 765 494 5725; fax: +1 765 494 0539.
E-mail address: ramani@purdue.edu (K. Ramani).

0010-4485/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2011.12.004

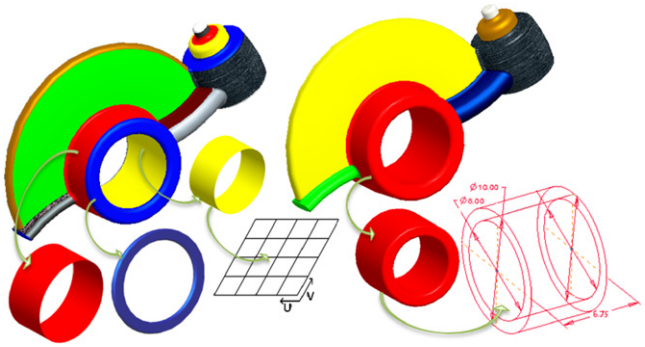


Fig. 1. Difference between surface–volume segmentation and subsequent parameterizations.

around that PCS. Fig. 5 shows a set of local cross-sections which collectively represents a single volumetric sweep segment. It is not feasible or necessary to cover each and every mesh facet with PCS as the amount of time and data will increase drastically for dense mesh models. Therefore, for the purpose of extraction we assume that for a small region around each PCS, the sweep cross-section is constant and represented by a single PCS.

In this paper, we introduce two new applications of the algorithms—Locally Linear Embedding (LLE) [3] and Affinity Propagation (AP) [4] for organizing and clustering PCS. The LLE algorithm analyzes the cross section (PCS) using their geometric properties to build a global manifold in an embedded space. The AP algorithm then clusters the local cross sections by propagating affinities among them in the embedded space to form different sweep components. The method may produce multiple but feasible sweep components corresponding to a particular portion of the original part. In such cases, user interaction is required to resolve the ambiguous interpretations. We also show the construction of a CAD model from the extracted sweep components using *CATIA*^{TM}.

We clearly distinguish that our intent is not to reconstruct the original object as designed in a complete sense. We characterize our capabilities as being able to handle those shapes with swept volumes where the "non-interacting" parts carry enough evidence together with the partial cross-sections. For example, when shell operations take out a large portion of the sweep, our method will not work.

1.1. Background

Our semi-automatic approach transforms the physical part into a set of generalized sweep components. A generalized sweep involves two components namely, the 2-dimensional profile(s) or cross-section(s) being swept and the 3-dimensional trajectory (trajectories) along which they are swept orthogonally. The different modeling operations typically used in creating CAD models in tools like *Pro/ENGINEER*^{TM} are nothing but special cases of a generalized

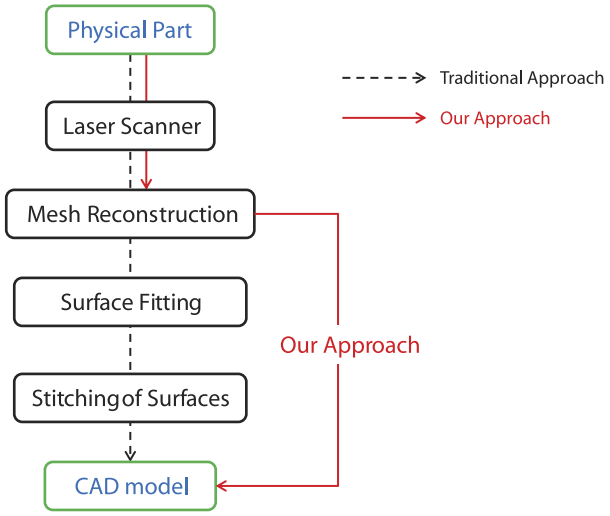


Fig. 3. Traditional reverse engineering pipeline versus our approach. We obtain the direct parameterization of CAD model by completely skipping the reconstruction of surfaces.

sweep. For example, an 'extrusion' is a sweep operation involving a 'constant' sketch swept along a 'linear' trajectory. The other cases are listed in Table 1. For any 2-dimensional cross-section swept along a trajectory (Fig. 4), the swept volume can be described as [5]:

$$X(u, s) = T(s)\Gamma(u) + \Psi(s) \tag{1}$$

where $u = [u1, u2]^T$, $\Gamma(u)$ represents a section being swept (a surface parameterized in two variables $(u1, u2)$), $\Psi(s)$ is the swept path parameterized by the arc length s, $T(s)$ the transformation matrix and $X(u1, u2, s)$ characterizes the set of all points inside and on the boundary of the swept volume. The swept surface $\Gamma(u)$ is a 2-dimensional section, hence its boundary can be represented by a single parameter t. Thus Eq. (1) can be represented as:

$$X(t, s) = T(s)\Gamma(t) + \Psi(s) \tag{2}$$

$X(t, s)$ characterizes the set of all the points on the boundary of sweep. Given a set of local cross-sections representing a single sweep, we can compute the transformation $T(s)$ between any two consecutive sections. Each individual cross-section can be parameterized to compute $\Gamma(t)$. And finally, the trajectory equation $\Psi(s)$ can be determined by approximating a curve which is perpendicular to each cross-section and passes through their centroid.

In the following section, we introduce two new algorithms LLE and AP, used to build a global manifold and subsequently cluster them to obtain sets of PCS. To the best of our knowledge these ideas are new to the field of DSR or CAD model segmentation. These two algorithms can be adapted to segment any data set having a representation of local distances.

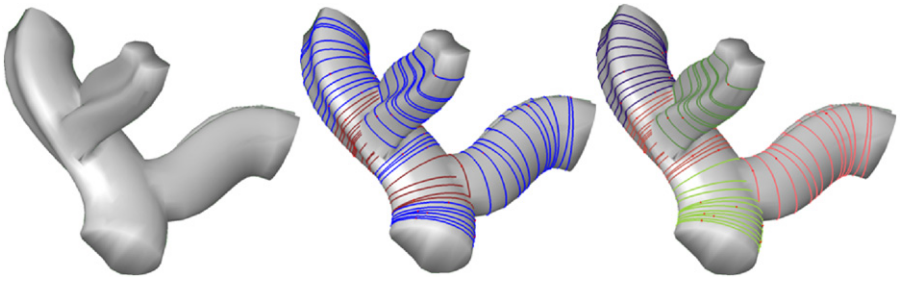
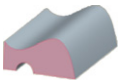
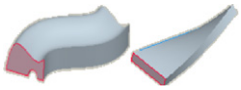
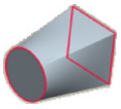
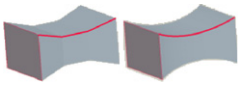
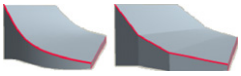


Fig. 2. Volumetric understanding: (From Left to Right) mesh model, PCS generation (full cross-sections are shaded in blue and partial cross-sections in brown), and sets of PCS representing different sweep segments obtained after clustering. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
The commonly used modeling operations in a CAD tool (*Pro/E*) that can be represented as generalized sweep.

	CAD operation	Sketch/cross-section	Trajectory	Example
1	Extrusion	Constant geometry and topology	Linear	
2	Revolve	Constant geometry and topology	Circular arc	
3	Helical sweep	Constant geometry and topology	Spiral curve	
4	Sweep	Constant (or) variable geometry, but same topology	Any curve	
5	Blend/loft	Variable geometry, but same topology	Linear	
6	Parallel blend	Constant geometry and topology	Linear	
7	Rotational blend	Constant (or) variable geometry, but same topology	Arc	
8	Swept blend	Constant (or) variable geometry, but same topology	Any curve	

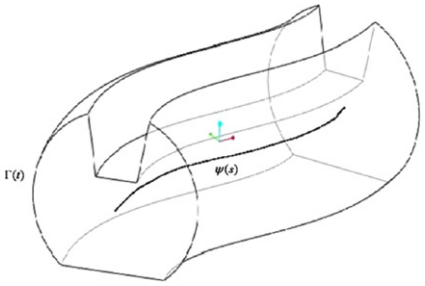


Fig. 4. Representation of basic sweep.

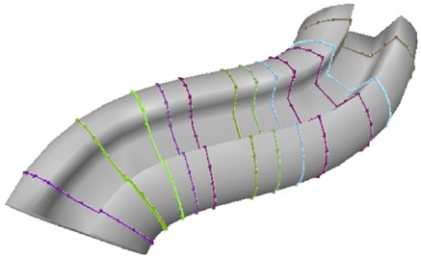


Fig. 5. Set of PCS representing a sweep section.

1.1.1. Locally Linear Embedding (LLE)

Roweis et al. (LLE) described an unsupervised machine learning algorithm in [3] that computes low-dimensional and neighborhood preserving embedding of high-dimensional data called Locally Linear Embedding (LLE). The data, which is assumed to lie on a nonlinear manifold, is mapped to a single global coordinate system of lower dimensionality. The mapping is derived from the symmetries of locally linear reconstructions, causing the actual computation of the embedding to reduce to a sparse eigenvalue problem. The algorithm attempts to compute a low-dimensional embedding with the property that nearby points in the high-dimensional space remain nearby and co-located to one another in the low-dimensional space. This algorithm only requires distances between nearby points to create the low-dimensional embedding. In contrast, various other dimensionality reduction algorithms require distance measure between distant points as well. For example, algorithm based on multidimensional scaling (MDS) [6] embed data by preserving straight line distances between all pairs of points. Even in recent algorithm such as ISOMAP [7] these distance are measured along the shortest path along the input manifold.

The LLE algorithm can be summarized as follows:

1. Discover the K-nearest neighbors of a point. Using the pairwise distances between points compute the set of nearest neighbors $N_i \in V$ of the point X_i
2. Compute the reconstruction weights W_{ij}; in this step a each of the nearest neighbors is assigned a weight W_{ij} which minimizes the error

$$E(W) = \sum_i \left| X_i - \sum_j W_{ij} X \right|^2. \tag{3}$$

   The weights W_{ij} represent the amount of contribution of each of the point while reconstructing the point X_i. The constraints on this minimization are $\sum_j W_{ij} = 1$ and $W_{ij} = 0 \forall j \in V/N_i$
3. Embedding in reduced dimensional space: In this step the high dimensional coordinates X_i are embedded in a low dimensional space $d \ll D$ by taking the $d + 1$ smallest eigenvectors and discarding the bottom most.

$$M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj}. \tag{4}$$

Here $\delta_{ij} = 1$ when $i = j$ or 0 otherwise. This eigen value optimization results in the d dimensional coordinates which optimize the function $C(W) = \sum_i |Y_i - \sum W_{ij} Y_j|^2$ using the weights computed in step 2.

1.1.2. Affinity Propagation (AP)

Affinity Propagation by Frey and Dueck [4] is an assumption-free clustering algorithm that simultaneously considers all data points as potential exemplars, exchanges real valued message between data points until a high-quality set of exemplars and corresponding clusters gradually emerges.

Affinity Propagation takes as input a set of pair-wise similarities between data points and finds clusters on the basis of maximizing the total similarity between data points and their exemplars. Affinity propagation sends two types of messages between data points: (i) Responsibilities are sent from data points to candidate exemplars and reflect the evidence of how well-suited the message-receiving point is to serve as an exemplar for the sending point. (ii) Availabilities are sent from candidates exemplars to data points and reflect the evidence of how appropriate it would be for the message-sending points to be the exemplar for the message-receiving points.

The algorithm can be summarized as follows:

1. Input: pair wise similarities $s(i, j)$ and set initial availabilities $a(i, k) = 0$.
2. Repeat: Responsibilities and availability updates until convergence

$$r(i, k) = s(i, k) - \max_{k' \text{ s.t } k' \neq k} \{a(i, k') + s(i, k')\} \tag{5}$$

$$a(i, k) = \min \left\{ 0, r(k, k) + \sum_{i' \text{ s.t. } i' \notin i, k} \max\{0, r(i', k)\} \right\}. \tag{6}$$

3. Output: Clusters.

1.2. Contributions

In this paper, we have

- Improved our previous algorithm to extract PCS [8], by decreasing the number of spurious PCS and subsequently eliminating them by performing a simple neighborhood analysis using a similarity measure computed between PCS.
- Developed an algorithm to construct the global manifold from local distances between computed cross-sections using Local Linear Embedding (LLE).
- Developed an assumption-free clustering method based on affinity propagation [4] to group the cross-sections into sweep segments. These segments then form the basic building blocks of the reconstructed CAD model.

2. Related work

2.1. Segmentation

A critical step in DSR is segmentation and there are a large number of segmentation methods that exist today. They can be broadly divided into two categories: (i) 2d or surface-based and (ii) 3D or volume-based. Surface segmentation is achieved by segmenting the part into different surfaces. Similarly volumetric segmentation requires segmenting a part into different volumes or features. The differences between the two are highlighted in Fig. 1. Vrady et al. [1] provide a comprehensive survey in this area, which outlines the main flow from data capture, pre-processing, segmentation to a final CAD model creation. The main theme that recurs in all the prior work is related to estimating the local geometries (surface normals and curvature), surface segmentation (division of the points based on some 'surface' characteristics), classification (such as cylindrical) and reconstruction (finding the best fit) [9].

2.1.1. Surface segmentation

The area of mesh analysis has been explored in detail and a large number of methods have been proposed to segment the mesh model into different surfaces. Recently, there have been new methods that attempt to reconstruct surfaces. They use this surface information to extract shape feature by stitching surfaces together. Vrady et al. [10] combines the result from 'Morse theory' with special geometric modeling algorithms. They create a CAD like structure reflecting the original design intent by separating the primary region from highly curved transition region. Ye et al. [2] introduce an automatic surface based modeling strategy for organic shapes. Yang et al. [11] uses a moving least-square surface as underlying surface representation and also uses its properties to enable curvature adaptive intersection with CAD geometry. A benchmark study of 3D mesh segmentation is done by Chen et al. [12], which concluded that not one particular algorithm is better than the other. Also, the algorithms based on nonlocal shape feature resembles more closely to human perception of mesh segmentation.

Lavou et al. [13] presents a curvature based technique that decomposes a mesh object into homogeneous surface segments. Attene et al. [14] fit surfaces to possible primitives such as planes, spheres and cylinders by minimizing the approximation error. There are different attributes (symmetry, convexity, geodesic distance, etc.) based on which surface segmentation can be performed. Among them, planarity, elements angles and curvature are more appropriate and are widely used to analyze a surface of a mesh model as surveyed by Shamir [15].

Stamati and Fudos [16] describe a method that decomposes a point cloud into regions belonging to the constituent features of the corresponding CAD model. The method determines the concavity intensity, i.e., the shortest distance between a point and the 3D convex hull of the point cloud for each point in the point cloud and further uses this concavity intensity information to segment the point cloud. This method fails to fully extract volumetric information about the model. It only segments the surface components into constituent parts. Weiss et al. [17] describe a method that performs the surface reconstruction process by fitting surface patches over specific segments of the point cloud. Each patch represents a unique segment of the model's surface.

2.1.2. Volumetric segmentation

Volumetric segmentation is based on human perception, which is, how humans will decompose a part into different components. For example, a cat model can be decomposed into claws, legs, head and tail. Similarly a CAD part shown in Fig. 1 can be decomposed into seven different segments as marked in different colors. The aim of volumetric segmentation is to find meaningful volumetric components that correspond to 3D semantic parts [15]. Approaches to obtain volumetric segmentation proposed by Mortara et al. [18,19] obtain curves by taking intersection of balls centered on a vertex and mesh. They are limited to tube like structures and are applied for skeleton detection.

A study on 3D CAD model parameterizations using both surface-based and part-based techniques is done by Agathos et al. [20]. The region growing methods by Zhang et al. [21] segment part based on curvature information and a user defined threshold for high negative curvature. Zuckerberger et al. [22] uses a dual graph of polyhedral surface to collect convex area but faces the problems of over segmentation.

Katz et al. [23] describe a method to hierarchically segment a 3D mesh into constituent core and secondary components. First, a pose invariant representation of the mesh using multi-dimensional scaling is generated followed by detection of feature points through user defined criteria to establish secondary components

within the model. The core component is extracted through 'spherical mirroring' and secondary segments using breadth first search. This method is analogous to our work in terms of segmenting a cad model into constituent sweep components. However, it only generates surface based segmentations and fails to extract volumetric information of the model. Benk et al. [24] describe a method that entails constraint fitting in both 2D and 3D. It describes a set of constraints most frequently encountered in reverse engineering applications, and proposes a methodology to recognize these constraints within point sets. In our work, the extracted cross-sections are represented as a collection of points. The next logical step is to parameterize the cross-sections and the sweep trajectory to construct features in a CAD modeler. We will employ a similar method to constraint fit our PCS.

Li et al. [25] describe a set of algorithms tailored towards detecting the presence of specific geometric patterns and characteristics within point sets/meshes. It is very likely that such patterns exist as a result of intentional decisions made during the design of the corresponding model, and such design intent need to be coherently understood and reintegrated during the process of digital shape reconstruction. The PCS provides a strong prospect to detect specific geometric patterns and consistencies, and can be facilitated by evaluating shape and positional characteristics of the PCS.

Shapira et al. [26] describe a method that attempts to attain fundamental volumetric understanding of the geometry before segmenting the surface. The shape diameter function relates the exterior surface of a model to its inner volumetric characteristics, since it behaves as a parameter that represents distances between the model surface and the medial axis of the model. It is analogous to our work in terms how it utilizes volumetric information to extract constituent components of a geometric model.

More recent work by Ke et al. [27] suggests a sectional strategy for modeling point clouds without triangulation. Slicing, curve feature recognition, and constrained fitting were introduced. The focus of the work was post processing of the profiles for recognition of the feature points in a curve for splines and constraints. Then they constructed the surfaces from the cross-sections by skinning. They did not deal with complexities posed by feature intersections or sweeps along a trajectory. Rather, they resorted to looking at the normals of the entire object on the Gauss sphere at one time to see the translational patterns. This limits their approach to global structure and ignores the local geometries of a mesh object. In contrast, our method analyze the normals of local section on Gauss map to obtain prominent cross-sections (PCS), and then combine PCS to form global structure.

Yilmaz et al. [28] have attempted to reconstruct regions of a CAD model from a point cloud using the notion of cross sectional contours to describe the local sweep segments. However, because the process of cross section specification and orientation in this method is performed manually, it lacks an automated nature. On the contrary, our method is tailored to utilize the k-means and gauss-map algorithms to automatically detect appropriate locations and orientations for each cross sectional contour. Furthermore, the reconstruction performed in [28] only takes into account the cross sections residing immediately adjacent to the reconstructed region and therefore fails to consider any geometric pattern that may be prevalent within the neighborhood that extends beyond the adjacent points of that region. In our method, the PCS are distributed across the entire sweep segment and not just at points adjacent to the reconstructed region. This enables our method to retain a greater amount of geometric information during the reconstruction of the sweep segments.

Our method is different in aim, scope, and approach from the past work. We call our method, 'shape-aware volumetric understanding' as we completely skip reconstructing surfaces (typically done in reverse engineering) and directly extract higher level, volumetric information from mesh models. Each set of PCS represents a volumetric sweep component and is unique in terms of cross-section and/or trajectory.

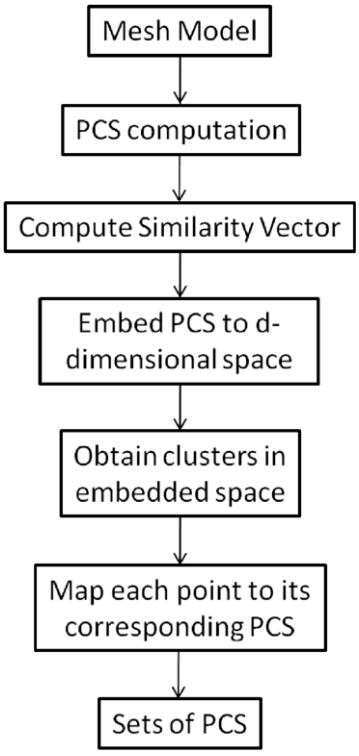


Fig. 6. System overview: takes a mesh model as input. Computes PCS and subsequently cluster them to form different sets of PCS.

3. System overview

Initially, Our method extracts a large number of PCS spanning across the mesh model (Section 4.1). We then compute a similarity vector V_{ij} (Section 4.2) which represents the transformation and shape similarity between PCS P_i and P_j. $\|V_{ij}\|$ gives us a local distance measure between any two PCS and we use this information to filter out the spurious PCS by doing a simple neighborhood analysis (Section 4.3). Next, V_{ij} is used to map the remaining PCS to a point in d-dimensional space (Section 5) by an embedding algorithm based on Locally Linear Embedding (LLE) [3]. This mapping is such that the Euclidean distance between any two points in the embedded space represents their relative global distance measure in the original space. Thus, the point distribution obtained is a true global distribution and corresponds to transformational and shape similarity between two PCS. Finally, the Euclidean distance in the transformed space is used for clustering these points into N different clusters (Section 6). Here the value of N is unknown as there can be any number of sweep segments in a given mesh models. The relaxation of not having prior knowledge of N is a distinguishing characteristic of our algorithm. To obtain such sets of PCS we adapted an assumption-free global clustering algorithm called 'Affinity Propagation' (AP) [4]. It is assumption-free because the number of clusters is not needed as input. It is global because it considers the possibility of clustering every two points, even when they are located far away from each other. After clustering, each point is mapped back to its corresponding PCS to obtain N sets of PCS, where each set (group of PCS) represents a single continuous sweep segment. When the points in each of the clusters is replaced by their corresponding PCS we get sets of PCS each of which corresponds to a unique sweep segment in the input mesh model. Fig. 6 shows the complete flow of our method.

4. Robust computation of PCS

The prominent cross-section (PCS) at a point is a 2-dimensional section that represents the cross-section of global sweep passing

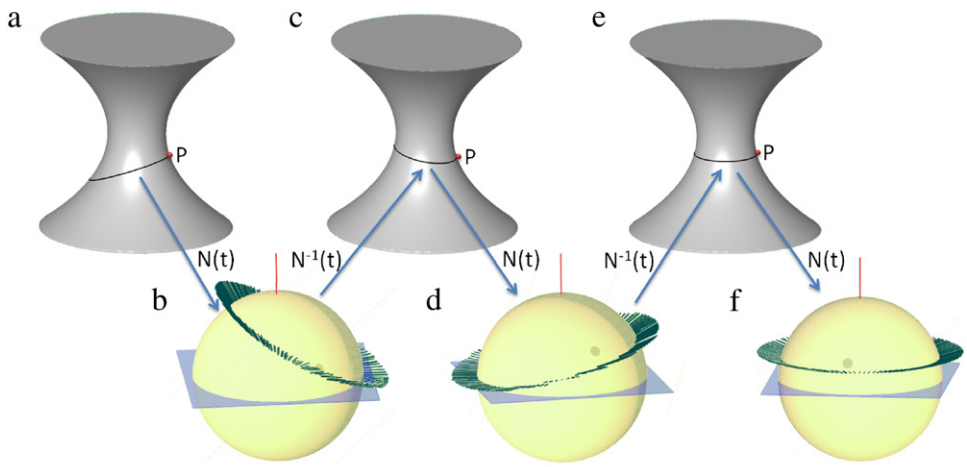


Fig. 7. Different steps in PCS computation at a given seed point P (red dot): (a) initial cross-section obtained from $K_{max} \times N$ at seed point, (b) normals obtained from initial intersection, mapped on the Gauss map, (c) intermediated cross-section obtained from least-square fit of normals on Gauss map, (d) normals obtained from intermediate intersection, mapped on the Gauss map, (e) final PCS obtained after convergence, (f) plot shows normals of final PCS on Gauss map. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

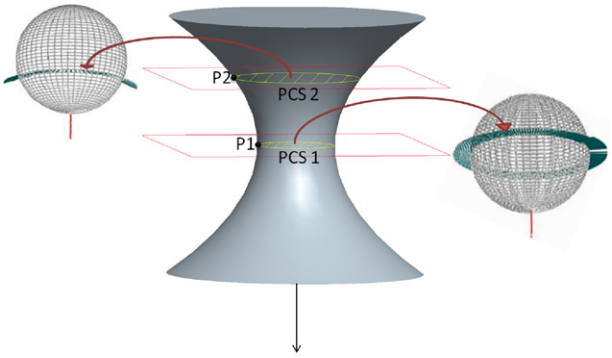


Fig. 8. PCS computed at two different seed points P1, P2 and their corresponding sectional Gauss map. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

through that point. For example, Fig. 8 shows two seed points P1 and P2 and their corresponding swept cross-sections PCS1 and PCS2 (yellow shaded area), both of which are a part of same sweep segment. We obtain a large number of such uniformly spaced seed points on a mesh model and then compute their corresponding PCS using the algorithm described in the next section.

In our previous work [8], we defined the concept of sectional Gauss map and used it to find PCS at any given point on a mesh object. A sectional Gauss map is a map that transforms a unit normal on a surface to a point on a unit sphere. Fig. 8 also shows the sectional Gauss map corresponding to the PCS1 and PCS2. Fig. 7 shows few intermediate steps during the computation of a PCS. Fig. 7(a) shows a point P and cross-section on initial plane, which is computed using the curvature and normal information at P ($K_{max} \times N$). $N(t)$ is a mapping that maps the unit normal of all the mesh–plane intersection points to that on the Gauss sphere. Fig. 7(b) shows plot of unit normals on Gauss map. Mapping $N^{-1}(t)$ orients the initial plane on the point P along the newly fit plane obtained from the normals on Gauss map. This process is repeated several times until the direction of initial plane (red line on Gauss map) converges to the direction of the fit plane obtained on Gauss map. For more details on the algorithm, computation of PPCS and its robustness please refer to [8].

The algorithm that computes PCS presented by Sellamani et al. [8] generates some outliers that are globally valid but locally

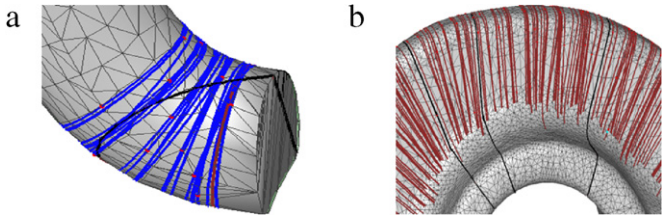


Fig. 9. Spurious PCS: (a) a spurious PCS obtained at the end of sweep section, (b) three spurious PCS obtained due to failure in angle threshold of a dense mesh model.

are not part of the expected swept volume. Fig. 9 shows the outliers (black curves) generated along with a large number of good PCS. Outliers may be generated: (i) When, stopping criteria fails at certain points due to high noise and uneven mesh density in underlying mesh model (Fig. 9(a)), (ii) At the ends of the parent sweep section (Fig. 9(b)), (iii) Where there is no underlying sweep evidence. On the large number of parts tested, the number of outliers varies between 5% and 15% of the total PCS. These bad PCS have to be removed before any further processing can be done and specially for automation.

In next the section, we present an improved version of our previous algorithm to compute the prominent cross-section. In subsequent sections we present an approach towards identifying and removing these locally invalid PCS.

4.1. Algorithm to compute PCS

In our previous approach [8], the computation of PCS is initiated at a seed point from a single input plane direction: $K_{min} \times N$ or $K_{max} \times N$ as chosen by the user. Now, unlike our previous approach, two PCS's are computed at each point from the two possible initial planes. For example, the initial plane shown in Fig. 10 corresponds to $K_{max} \times N$. In most of the cases, where we have only one possible sweep direction, both the initial planes converge to the same final PCS. However, in few cases, such as in closed shapes (ellipsoid), presence of intersecting features (a cylinder with a hole), when the seed point lies at the boundary of the underlying sweep (as shown in Fig. 11), there could be more than one possible sweep directions and we can obtain two different PCS.

Out of these two PCS we select the PCS having lower value of least fit error on final Gauss map. This is derived from the definition

Algorithm 1 PCS Computation (Mesh)

for a point $p_i \in$ sampled points **do**
  PL_i = Identify two starting plane (PL_1 and PL_2) at point (p_i) corresponding to max and min curvature.
  for For each plane PL_i **do**
    repeat
      Identify point normals N_i which intersect with (PL_i).
      Obtain normals $N_j \in N_i$ around p_i without any sudden change in angle.
      Plot point normals N_j on sectional Gauss map.
      PL_i = Find best fit plane on the Gauss map.
    until The PL_i converges to the threshold.
  end for
  PL = Min(error(PL_1), error(PL_2))
  Generate PCS using above PL at the point (p_i)
end for

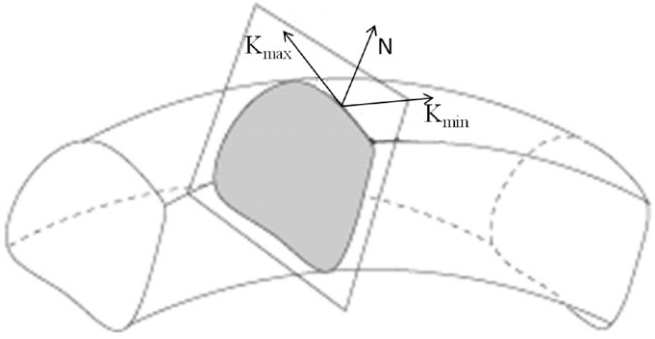


Fig. 10. Computation of initial plane at a given seed point. Above plane corresponds to $K_{max} \times N$.

of PCS in [8], that the angle between normals and its cutting plane should be constant. Hence, all such normals on Gauss map should lie completely on the fit plane as shown in Fig. 7(f). Thus given the points on the Gauss map and their corresponding fit plane, the error value represents the fit quality. Fig. 11 shows two PCS obtained from two possible initial planes ($K_{min} \times N$ and $K_{max} \times N$) at a same seed point. In the same figure, the corresponding Gauss map shows the plot of surface normal at all the points of intersection between PCS plane and mesh model. The PCS part of the parent sweep (brown in color) has a more uniform and constant distribution of normal on the Gauss map and therefore has a much lower fit error than second PCS (blue in color). This lower least fit error hypothesis has been confirmed by experimental results observed in a wide variety of CAD models that were tested (visual results can be found in Section 7). Consequently, the inclusion of a second input plane direction resolves the issue of extracting of a PCS that might seem locally compatible, but globally incorrect (i.e. not accurately representing a sweep cross section). Furthermore, in case an inaccurate PCS does happen to pass the least fit error test, they will be detected and removed by the outlier removal algorithm described in Section 4.3. The time complexity of this algorithm is $O(NM)$. Where N is the total number of sampled points and M is the number of iterations.

This modified algorithm removes the dependency on the user choice of initial plane ($K_{min} \times N$ or $K_{max} \times N$). It also improves the quality of result by choosing the PCS having lower fit error on the gauss map. This improvement is only effective in cases where more than one PCS can exist at a given seed point. Hence, the improvement in result varies across mesh models.

4.2. Similarity vector between PCS

We need a distance measure to create a global manifold of PCS and we define one based on the geometric properties of PCS.

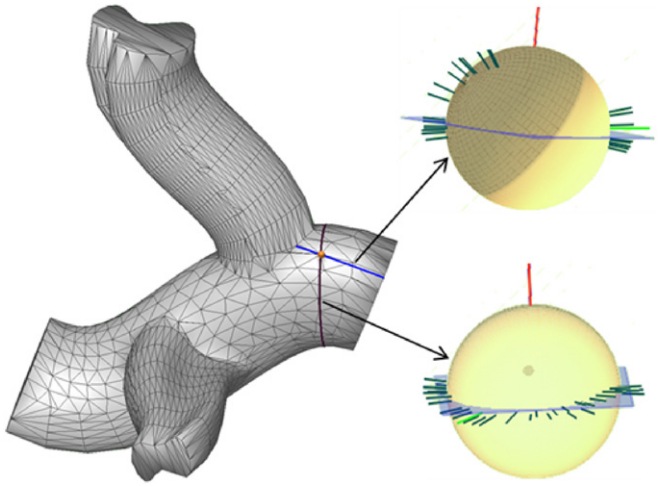


Fig. 11. Two different PCS obtained at same seed point from two different initial planes.

Between a pair of PCS: P_i and P_j, we compute a similarity vector V_{ij}, whose elements are composed of translation, rotation, scaling and shape similarity. Amato et al. [29] explains the importance of these elements and Mitra et al. [30] and Pauly et al. [31] uses similar measures to detect structure symmetry and regularity. For any two PCS P_i and P_j, we transform P_j into P_j' such that the dissimilarity between P_j' and P_i is minimum. P_j' can be represented in terms of P_j as:

$$P_j' = S \times P_j \times R + T \tag{7}$$

where T is the translation, R is the rotation and S is the scale between P_i and P_j. R, T and S are calculated to minimize the error ϵ below based on the work done in least-sq fitting of 3D points by Arun et al. [32]

$$\epsilon^2 = |P_i - P_j|^2. \tag{8}$$

The only dissimilarity remaining between P_i and P_j' is the dissimilarity between their shapes. We label dissimilarity measure of shape between two PCS as D, which captures the shape difference. For example, Root Mean Square Distance (RMSD), captures the difference in shape between two sets of points. Finally, using all the above measures (T, R, S, D) we define a similarity vector V_{ij} between to PCS P_i and P_j as:

$$V_{ij} = (T_{ij}, R_{ij}, S_{ij}, D_{ij}). \tag{9}$$

As each component of this similarity vector measures different geometric properties, they are quantified within numerical ranges that differ from one another. Therefore it is likely that one component can dominate the other three simply because it is based on a range that exists between large numerical values. Normalization of the components enables us to quantify each component on a common scale, where the relative influence of each component is accurately represented. Eq. (10) illustrates the range within which such normalization can be accomplished. It has been empirically determined that the ranges that provide the best normalization are as follows: (a) 0–1 for D, (b) 0 to $T_{max}/2$ for T, where T_{max} represents the maximum translation observed within all the points of a given PCS while transforming to another PCS, (c) 0°–90° for R, and (d) 0–10 for S.

$$(D = 1) \approxeq \left(T = \frac{T_{max}}{2}\right) \approxeq (R = 90°) \approxeq (S = 10). \tag{10}$$

We compute the shape and scale factor similar to the procrustes analysis based on [33]. We represent the PCS as a curve formed by

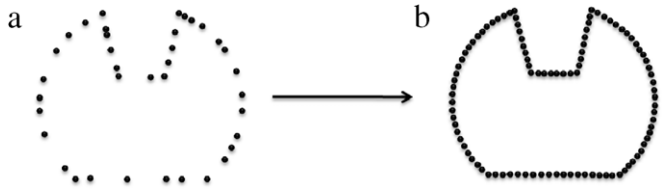


Fig. 12. Sampling PCS: (a) each point represents the intersection of mesh edge with final PCS plane, (b) sampled n ($n = 100$) equally spaced points.

3d-points X_i on a 2d-plane (Fig. 12(a)). To obtain the above four elements, we first re-sample X_i into n ($n = 100$) equally spaced points (PCS $= \{X_i | i = 0, 1, 2, \ldots, 100\}$) as shown in Fig. 12(b). Re-Sampling is done by adapting the algorithm presented by Wobbrock et al. [34]. This step ensures that while calculating the vector elements the number of points contained in a PCS is uniform across the model. Also, by maintaining a high value for point density we minimize the error component while calculating the shape similarity between any two PCS. The steps to compute the D and S are explained below.

1. Compute the centroid of each PCS as $C_i = \frac{\sum X_i}{n}$ and $C_j = \frac{\sum X_j}{n}$. Position the centroid at origin as, $X_i = X_i - C_i$ and $X_j = X_j - C_j$.
2. Calculate the size of two PCS as $N_i = (\sum (X_i^2))^{\frac{1}{2}}$ and $N_j = (\sum (X_j^2))^{\frac{1}{2}}$ and normalize them by its size $X_i = \frac{X_i}{N_i}$ and $X_j = \frac{X_j}{N_j}$ where new X_i and X_j represents the scale free PCS.
3. Compute covariance matrix $A = X_i^T X_j$.
4. Perform singular value decomposition (SVD) of the covariance matrix $A = U \Delta V$.
5. Compute a measure of the shape similarity $D_{ij} = 1 - (trace(\Delta))^2$.
6. The ratio of N_i and N_j represent the scale factor $S_{ij} = trace(\Delta) \times \frac{N_i}{N_j}$.

During the above computation we can also obtain the rotation matrix as VU^T. However, VU^T obtained, depends on the start point of two PCS, which may be anywhere within our input. Therefore, we calculate the rotation similarity as $R_{ij} = 1 - |n_i \cdot n_j|$, where n_i and n_j represents the unit normal vectors of the plane containing two initial PCS P_i and P_j. Translation similarity is the Euclidean distance between centroid of two PCS, $T_{ij} = \|C_i - C_j\|$. The time complexity to obtain the similarity vector is $O(N^2)$. Where N is the total number of PCS obtained on a mesh model.

In the following sections, we use this similarity vector to identify outlier PCS's and it is also used to stitch these locally similar PCS to compute a global distance.

4.3. Removal of PCS outliers

In the next section we propose an algorithm that maps PCS to points in d-dimensional space while preserving their neighborhood distances. This algorithm performs a neighborhood analysis to create global embedding of PCS. Therefore, it is important that each PCS should have rich neighborhood, which is ensured by obtaining a large number of PCS distributed uniformly throughout the mesh. Also, outliers can significantly affect the embedding of PCS, therefore the input data should also be free from outliers as much as possible.

We define outliers as the PCS that are not part of the local sweep component. Therefore, our goal is to remove PCS that are locally dissimilar. The probability of an outlier appearing as a neighbor of good PCS is very low because it is not locally similar to the neighboring cross-sections. We used the distance measure $D = \|V_{ij}\|^2 = (T_{ij})^2 + (R_{ij})^2 + (S_{ij})^2 + (P_{ij})^2$, to calculate the k-nearest neighbor for each PCS, ($k \approx 10$). The number of times a PCS appears as a neighbor of any other PCS is defined as its 'neighborhood frequency'. We plot a histogram of the cross-sections along the x-axis with respect to their neighborhood frequencies along the y-axis. While testing a large number of parts, a threshold value (neighborhood frequency $\approx$ 6) is used to filter the outliers. Fig. 13(b) shows the results obtained after applying the above approach on parts shown in Fig. 13(a).

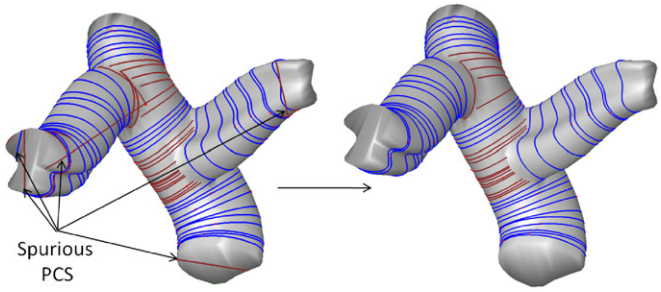


Fig. 13. Removing outlier PCS using neighborhood analysis.

The algorithm proposed in the next section is based on neighborhood formation. On filtering out the outliers by using the neighborhood frequency, we ensure that the PCS, which are not a part of the neighborhood formation of other PCS, get filtered out. This step is quite effective in removing the PCS which are not part of any local sweep sections. However, there may be some PCS that are not part of local sweep component but pass this step as they appear in groups that are locally similar to each other. Such PCS will finally appear as separate sets once the PCS are grouped together to represent different swept volumes. Fig. 13 shows the result on a standard part. The part in Fig. 13(a) contains few spurious PCS along with large number of good PCS. The spurious PCS (as indicated by the arrow) are locally dissimilar and stand out from their neighbors. Hence, they will not appear in the list of k-nearest neighbors of any good PCS. Subsequently, such PCS fall well below the threshold limit of neighborhood frequency and gets filtered out (Fig. 13(b)).

5. PCS embedding

To achieve a global understanding of the shape, the locally similar cross-sections have to be joined together to form continuous sweep components. The set of cross-sections that make up these components possess the property that they are locally similar. We cannot cluster the PCS directly since the similarity vector developed in Section 4 is locally linear and hence represents local but not the true global similarity between any two PCS. For example, the centroidal distance between distant PCS as shown in red in Fig. 14, is not a correct distance measure when we look at the complete mesh model. The correct distance can be computed when centroidal distances of all the PCS lying between them are added together consecutively.

To obtain a global embedding, we map each cross-section to a point in d-dimensional space, such that the neighborhood similarity between a cross-section and its neighbor is preserved and the resulting distribution is also a better representative of global distances between non-neighboring PCS (See Fig. 15). This is achieved by using an unsupervised algorithm that determines a distribution of high dimensional manifold in a single coordinate system of lower dimensionality by using local similarity measures. Among the four elements of the similarity vector, T and R are 3×3 transformation matrices while S and D are scalar quantities. This configuration of the similarity vector is equivalent to a row vector that comprises of all the basis elements of the T and R matrices along with elements representing S and D. This implies that each PCS is represented as a single point embedded in a high

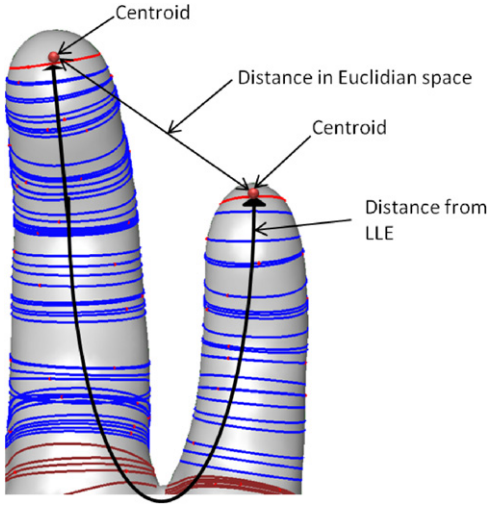


Fig. 14. Centroid distance in embedded space compared with same distance in original Euclidean space. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

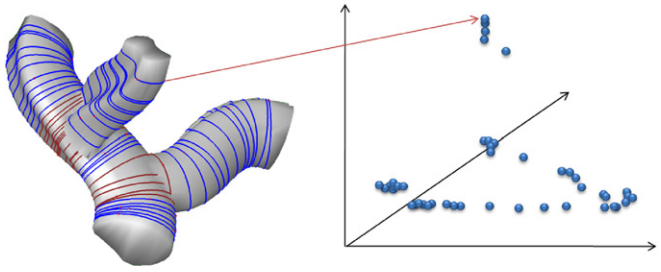


Fig. 15. Neighborhoods preserving embedding of PCS obtained from LLE. Each PCS gets mapped to a point in d-dimensional space. The local covariance between PCS and its neighbor is preserved in mapped space.

dimensional space, since the distance metric between two PCS is found within this space. The analysis of such high dimensional data can be made computationally feasible through the dimensionality reduction facilitated by the above unsupervised algorithm.

To obtain a global representation of PCS, we first construct the neighborhood of each PCS by finding their K-nearest ($K \approxeq 10$) neighbors. We take $\|V_{ij}\|$ as a distance measure to find K-nearest neighbors of PCS P_i by selecting K smallest values of $\|V_{ij}\|\, j \in [1, N]$, where N is total number of PCS on the mesh model. $\|V_{ij}\|$ will be close to zero for a PCS and its neighbors as all the four factors of the similarity converge to zero. For distant PCS, at least one out of these four should stand out. Only these K-nearest neighbors will take part in reconstruction of point corresponding to P_i. We first assign weight w_{ij} to each of these neighbors such that the reconstruction error E_i is min.

$$E_i = \sum \left| P_i - \sum w_{ij} \times P_j \right|^2. \tag{11}$$

E_i is minimized by applying two constraints: (i) Only neighbors contribute to the reconstruction of P_i so w_{ij} is zero for all other PCS. (ii) $\sum w_{ij} = 1$ (for more details, please refer [3]). Based on these two constraints, weights are obtained by solving the above least-square problem as explained in [3]. We address this least-square problem as follows:

The first step of the solution is to obtain a local covariance matrix C. This is obtained by shifting the ith point to origin to get the relative distance vector between the point and its neighbor. In our case, we already have a relative vector V_{ij}, each element of which represents the relative distance between P_i and its K-nearest neighbors $P_j\ j = 1, 2, 3, \ldots, K$. Therefore, local covariance matrix C for P_i is obtained as:

$$C_{K\times K} = [V_{i1}, V_{i2}, V_{i3}, \ldots, V_{iK}]^T [V_{i1}, V_{i2}, V_{i3}, \ldots, V_{iK}]. \tag{12}$$

K weights corresponding to P_i are obtained as $K \times 1$ vector W_i by solving equation below:

$$CW_i = I \tag{13}$$

where, $I_{K\times 1}$ is a unit vector. Next step is mapping each PCS P_i to a low-dimension vector $\overrightarrow{X_i}$ such that the mapping is neighborhood preserving. This is achieved by choosing $\overrightarrow{X_i}$ to minimize the cost function below:

$$\Phi(X) = \sum_i \left| \overrightarrow{X_i} - \sum_j W_{ij} \overrightarrow{X_j} \right|^2 \tag{14}$$

where, W_{ij} is the weight corresponding to PCS P_i and its K-nearest neighbor P_j as calculated above. The above cost function is subjected to two constraints: (i) The resultant vector will be centered at origin i.e. $\sum_i \overrightarrow{X_i} = \overrightarrow{0}$. (ii) To avoid degenerated solutions the embedded vector should have unit covariance $\frac{1}{N}\sum_i \overrightarrow{X_i} \otimes \overrightarrow{X_i} = I$, where I is $d \times d$ identity matrix. Finally, the low-dimensional embedding is obtained by finding the eigenvectors corresponding to d ($d \approxeq 5$) smallest eigenvalues of the sparse matrix M_{ij} as explained in [3]. M_{ij} is calculated as shown below:

$$M_{ij} = \Delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} \times W_{kj} \tag{15}$$

where $\Delta_{ij} = 1$ if $i = j$ and 0 otherwise. W_{xy} represents the weight corresponds to yth nearest neighbor of PCS P_x. The d eigenvectors when combined together represents $d \times N$ matrix where ith column represents a d-dimensional vector $\overrightarrow{X_i}$ which corresponds to PCS P_i.

Time complexities of different steps in this algorithm is $O(KDN^2) + O(DNK^3) + O(dN^2)$ [35]. Where N is the total number of PCS embedded, K is the number of neighbors ($\approxeq$10), D is the input dimension (=4) and d is the output dimension ($\approxeq$5). In our method K, D and d can be considered as constant. Hence the total time complexity of above algorithm is $O(N^2)$.

These steps map each PCS to a d-dimensional vector, while preserving the neighborhood distribution among them. The Euclidean distance between any two point's X_i and X_j in this new space is representative of a global relative distance between their corresponding PCS P_i and P_j. Now these points can be grouped together to obtain global sweep segments.

6. Constructing sweep components

The previous step maps the PCS to a point in d-dimensional space and provides us a measure of global distance in terms of Euclidean distance between points. In the next step we cluster the d-dimensional points in order to acquire the global sweep components. A large number of different clustering algorithms such as k-means and k-center [36] heavily depend on the randomly chosen initial seed points. They also require the potential number of clusters to be pre-specified, which is unknown in our case, as we are initially unaware of the number of volumetric sweep components in a mesh model. Furthermore, above mentioned algorithms consider only the nearest neighbors to form a cluster. This is not appropriate in our case as we are looking to cluster even the distant PCS if they belong to the same continuous sweep component. To address this problem, we need a clustering algorithm that is assumption-free and produces optimal clusters after considering the arrangement of all the sweep components in a globally consistent manner.

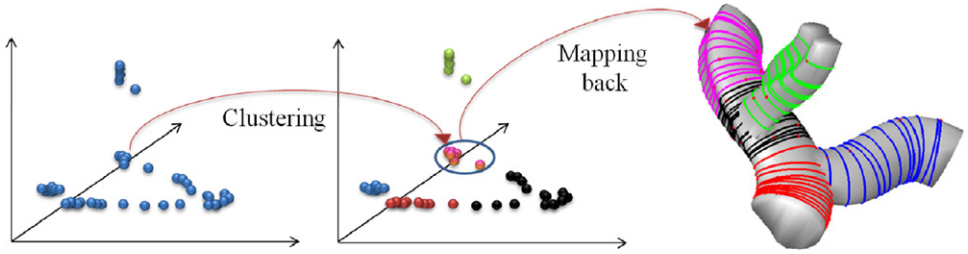


Fig. 16. Clustering of points in embedded space using AP algorithm and mapping back each point to corresponding PCS to obtain final sets shown in different colors on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

We derived our method from a message passing clustering algorithm called Affinity Propagation (AP). The AP described in [4] is a recently proposed algorithm that takes the measures of similarity (distance) between pairs of points as the input. It begins with considering all data points as possible cluster centers and iterates through, by passing real valued messages to trim down the number of clusters until final clusters gradually emerge. At any given stage of iteration, it passes messages between possible cluster centers and their children stating their relative affinity towards each other, based on which the cluster centers and their children are modified for next iteration.

We used a modified version of Affinity Propagation know as Adaptive Affinity Propagation (AAP) clustering [37], which also takes as input a collection of real-valued similarities between data points. AAP improves upon AP by eliminating oscillations and can obtain better quality clusters. In our adaptation, similarity $S(i, k)$ (computed using Eq. (16)) indicates how well the PCS P_k is suited to be the parent for PCS P_i and obtained by calculating Euclidean distance (as shown below) between point X_i and X_k in the d-dimensional manifold obtained in previous section. Time complexity of AP is $O(N^2 \log N)$. Where N is the total number of input data points. AAP has higher time complexity than AP as compared in [37].

$$S(i, k) = \left\{ \sum_{j=1}^{d} (X_{ij} - X_{kj})^2 \right\}^{1/2}. \tag{16}$$

This algorithm does not require a pre-specified number of clusters as input and also generates the different potential sets of output clusters in one run. Also, the message passing procedure is a global approach, as messages are exchanged between all the pairs of points simultaneously and based on these exemplars are selected. Each cluster of points represents a volumetric sweep segment when mapped back to corresponding PCS on the mesh object as shown in Fig. 16.

7. Results and discussion

In Section 3, we outlined the complete pipeline of our work. Initially, uniform samples of vertices v_i $i \in [1, k]$ are obtained on mesh model by using a k-means clustering algorithm [38] and then we compute the PCS P_i corresponding to each vertex v_i. We then propose a similarity vector V_{ij} between every possible pair of PCS P_i and P_j where $i, j \in [1, k]$ and $i \neq j$. Using similarity vector V_{ij} we first clean up the PCS which are not taking part in neighborhood formation and obtain $P_j \subset P_i$. This step helps us in two ways: Firstly, we get rid of spurious cross-sections and secondly, we make sure the input to the embedding algorithm does not contain any outlier since it can affect the outcome of algorithm and subsequent clustering process. Again, we use the same similarity vectors for the remaining PCS P_j to map each of them to a point in d-dimensional space using LLE. Finally, we cluster these points and obtain different sets of PCS, each set independently representing a unique possible sweep component. Table 2 demonstrates the results on CAD parts having different shape and mesh density.

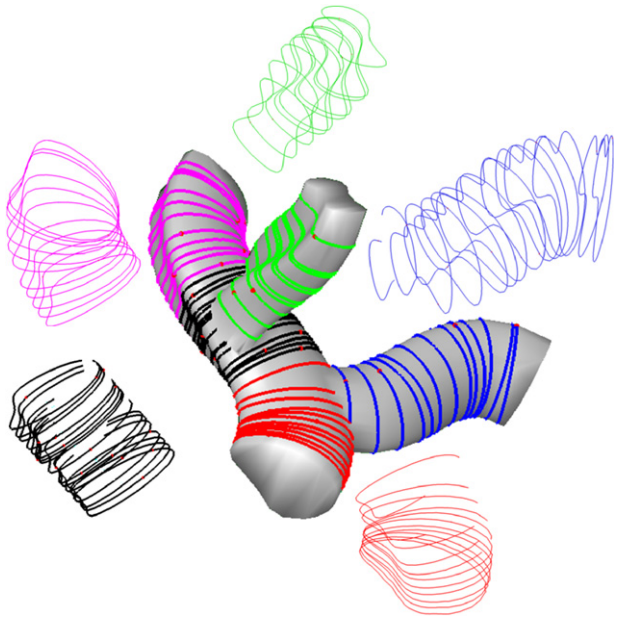


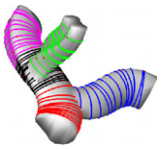
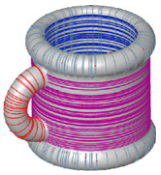
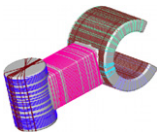
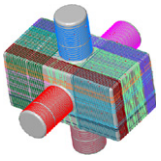
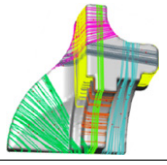
Fig. 17. Results obtained on Tree model distance in original. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

7.1. User created CAD models

The model in Fig. 17 contains 3 intersecting sweep segments of different shapes and sizes. Our method identifies 5 sweep segments, where one of the sweep is broken into three different segments (red, black and pink) at two regions of intersections. The sets in black and red contain partial PCS capturing sweep intersections and maintaining the continuity of sweep sections.

Fig. 18 illustrates results on a part having intersecting sweeps of different shapes. It has 3 sweep components and we obtain 7 sets of PCS, out of which 3 sets represent different segments of same sweep component as shown in Fig. 18(c). Both the front and back cylindrical sections have two different possible representations as shown in Fig. 18(d, c) and Fig. 18(a, e) respectively. Note that set (d) in front section and set (a) in the back section contain a high density of PCS, while the PCS in set (c) and (e) in the respective sections are less densely populated. The reason for such a difference in number is because we generate two PCS on every seed point, but select only the PCS that has the least fit error on Gauss map. Also, the distribution of normals is more continuous in the region corresponding to the PCS in sets (d) and (a) than to those PCS in sets (c) and (e), resulting in least fit error on the Gauss map. Therefore, sets (a, b, d) combined together represent the complete model having 3 different sweep segments. The PCS density can be used as a criterion to resolve the ambiguity when one or more sweep components represent the same region of a part. However, denser PCS does not necessarily mean the best sweep component representation. Hence, user interaction will be required to choose the best set in such a scenario.

Table 2
Results obtained on parts with different shapes and mesh densities.

Parts	No. of vertices	No. of PCS	No. of neighbors in embedding (K)	Embedded dimension (d)	No. of sets obtained	Time (s)	Comments
	1508	64	10	5	5	7.98	Sweep with intersections. Black and red PCS are partial PCS at intersections.
	5668	289	12	4	4	36.65	Set of PCS in blue color represents sweep where material is removed.
	55,490	560	10	5	7	923.73	More than one set of PCS (front cylinder) for same sweep segment capturing different design intent.
	33,990	760	12	5	14	713.86	Large number of sets obtained in rectangular section representing same sweep segments in different direction.
	11,984	201	8	5	7	240.28	High threshold value used to remove spurious PCS, resulting in uncovered mesh regions.

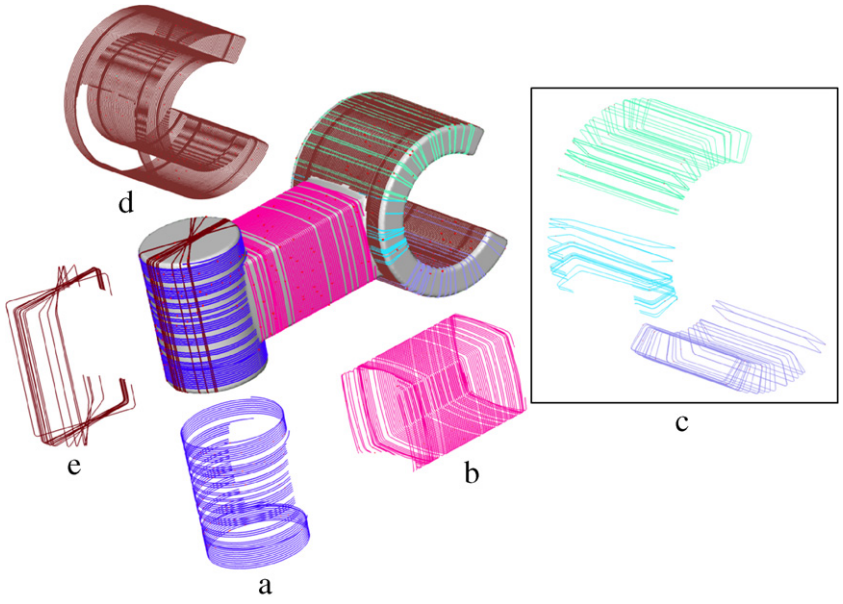


Fig. 18. Different design intent represented in (d) and (c) for same sweep segment.

Fig. 19 shows a cup having two major sweep components, a handle and a body. We obtain the handle as one set (a) and three other sets as (b, c, d) representing different sweeps in the body of the cup. We also get a sweep where material is removed as shown in (b). The information of material direction (in this case outside the circle) can be used to determine the attribute of sweep, whether material is added or removed. Here (a, c) together represent complete part (handle and body) and (b, d) represent different sweeps in the body of part.

Fig. 20 shows sweep interactions resulting in breakage of 3 sweeps (2 cylinders and 1 cuboid) into 10 different smaller sweep components. The cuboid is intersected from four sides, resulting in

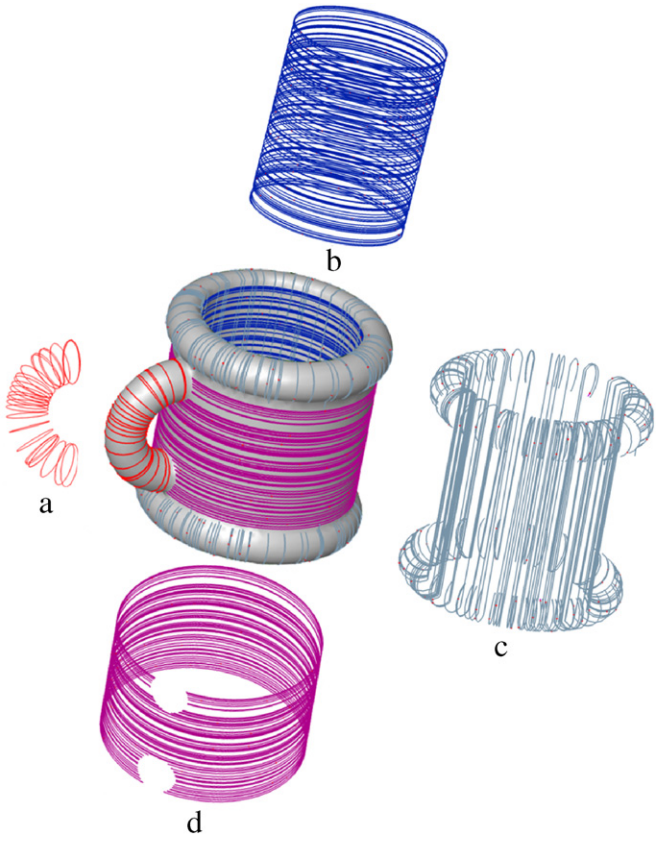


Fig. 19. Sets of PCS obtained on a mug. Blue set represents a sweep which is removing material. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

segmentation of one sweep component into a number of smaller components. Sets (a, f, e, j) capture the complete cuboid and sets (i, d) capture it partially, representing a different design intent. Sets (b, c, g, h) capture the cylindrical sections running across the cuboid. Few sets are rejected and not shown separately as they contain sparse and non-uniformly spaced PCS.

Fig. 21 shows sets obtained on a Fan Disk. This part is obtained after large number of sweep operations and contains few regions where there is enough evidence of sweep. This is one of the limitations of our method; regions without any sweep evidence cannot be represented as generalized sweep segment. For example, the set in green color is a group of PCS not representing any sweep. Also, due to large number of sweep operations, the number of spurious PCS is increased and their subsequent removal leave significant uncovered region. Hence, in this case, our algorithm cannot produce a complete representation of the object, but instead can identify potential sweeps present in the model which may represent the design intent or at least may be useful for further editing the model.

7.2. Scanned CAD model

In this section we demonstrate the results obtained on three scanned point cloud data—rocker arm, an engine component and a twisted drill bit. The point clouds have been obtained from CGRG Point Cloud Repository [39]. These data sets have been processed for noise removal, flaw correction, hole-filling and basic symmetry-based repairing. The point clouds were first converted to 3D triangular mesh models using a commercial tool, Geomagic. However, in order to reduce the computation time, these highly dense models were re-meshed and reduced in size intelligently, such that the density of triangles is higher in the curved regions than in the planar regions. Table 3 shows the results of the algorithms applied on mesh models obtained from the scanned point cloud data.

Fig. 22 shows the result obtained on a laser scanned rocker arm model. The part is obtained after converting scanned point cloud data to a triangulated mesh model. The part contains high noise levels which result in noise in the orientation of PCS in a sweep segment. For example PCS obtained in red color set (g) needs to be smooth before any further processing can be done. Also, few small features have been grouped together (set (e)) due to insufficient number of PCS in that region.

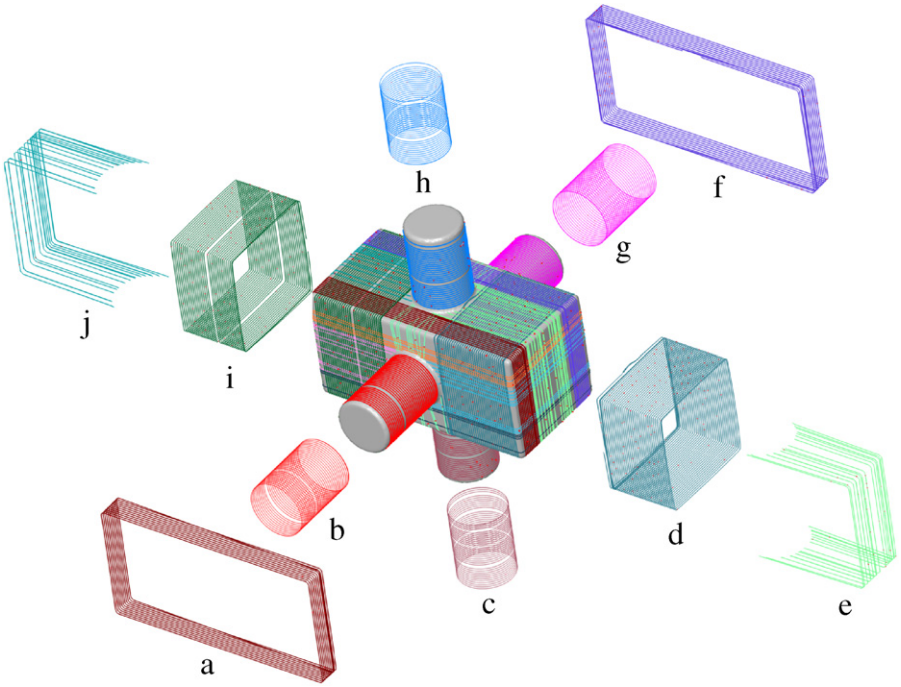


Fig. 20. Large number of sets obtained on a simple cube cylinder intersection.

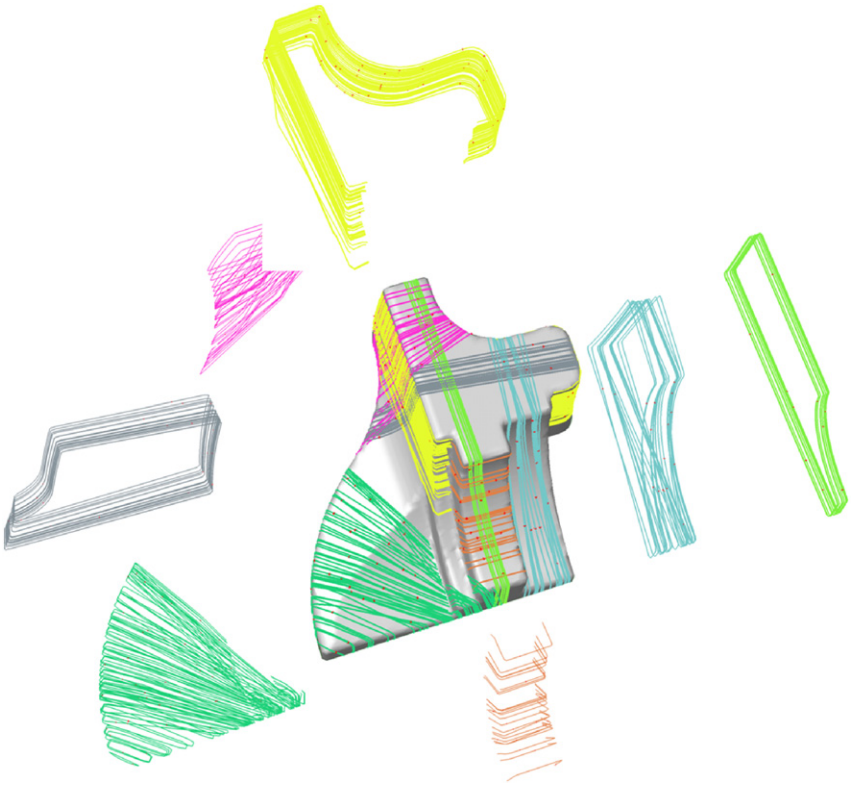


Fig. 21. Sets obtained on standard fan model with large number of sweep operations.

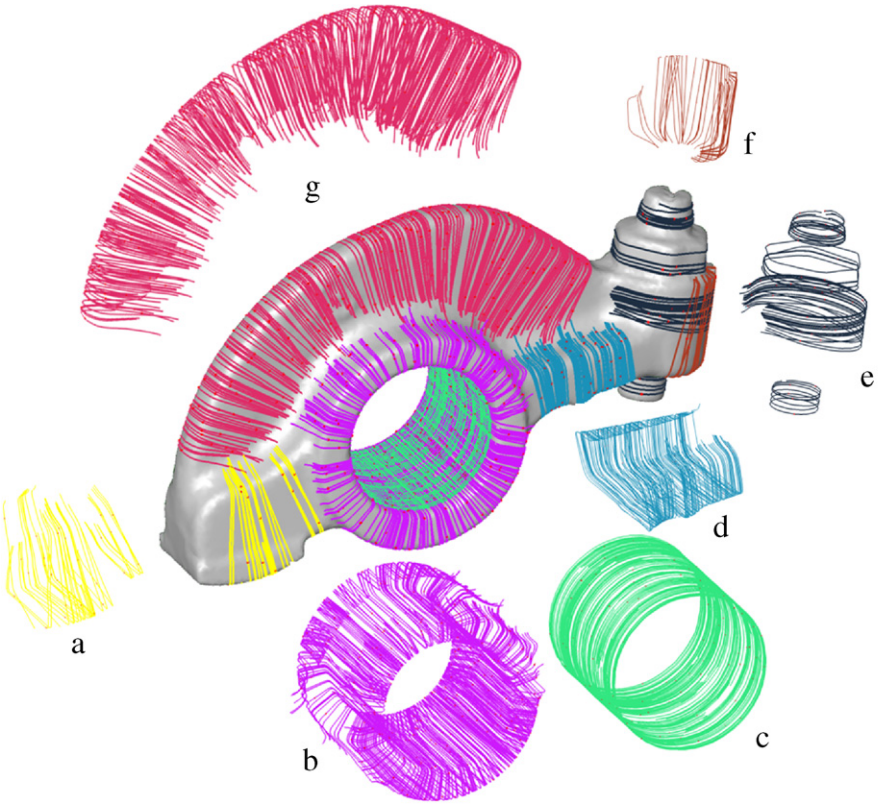
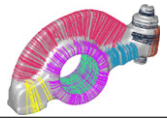
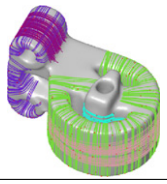


Fig. 22. Sets obtained on laser scanned Rocker Arm.

Table 3
Results obtained on mesh parts obtained from laser scanned point cloud data.

Parts	No. of vertices	No. of PCS	No. of neighbors in embedding (K)	Embedded dimension (d)	No. of sets obtained	Time (s)	Comments
	21,804	889	10	6	8	1242.3	PCS obtained on complete part without selecting any region.
	81,899	436	10	6	6	1418.38	PCS obtained by selecting the region having sweep evidence.
	80,523	296	8	4	4	464.2	PCS obtained separately on small feature at the top on the bolt.

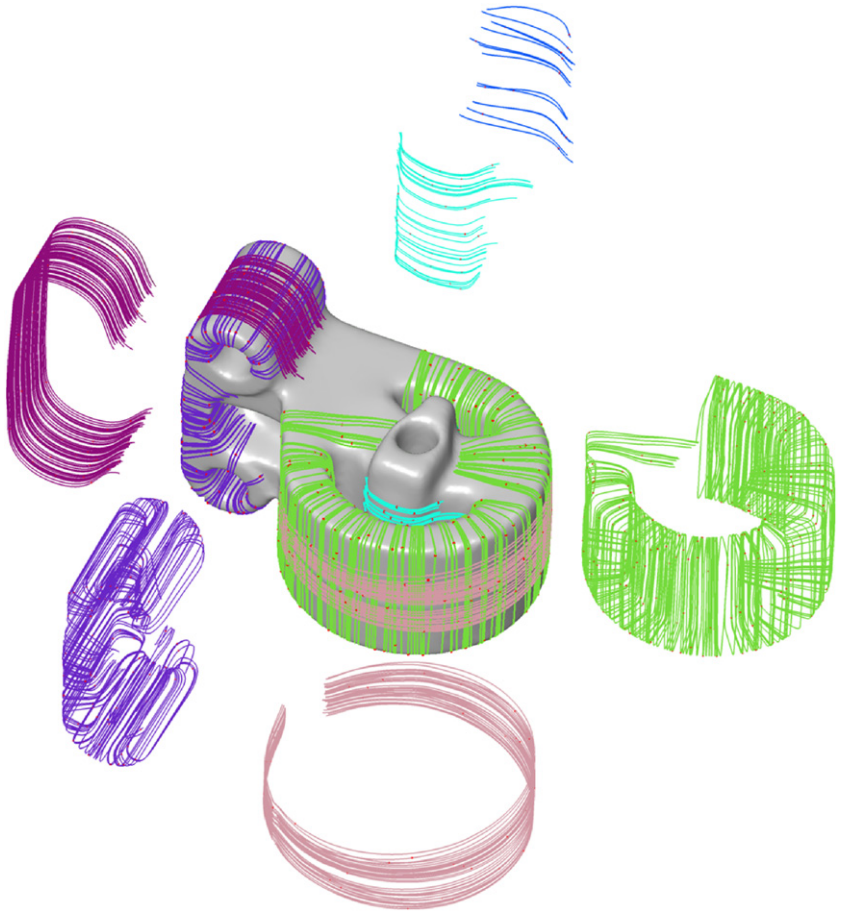


Fig. 23. Sets obtained on mesh engine part obtained from scanned point cloud data.

Fig. 23 shows a scanned engine part. The part was scanned with a Handyscan EXASCAN laser scanner (accuracy 40 μm) and in one piece. The point cloud was post processed for holes and error with a reverse engineering tool suite. Prior to scanning, the part had to be sprayed with a thin white coat. We obtained PCS on a selected region of the parts where there are enough evidence of sweep. We are able to extract the majority of sweep sections as shown in the figure. Similar to the previous cases, we obtain two different design intents for the same region. Also, despite our limitations that we do not have enough sweep evidence, we were able to extract sufficient sweep segments with the help of some user interaction for selecting a region of interest for targeted PCS extraction.

Fig. 24 shows the result of the algorithms applied on a scanned twisted drill bit. The part was scanned using a 3D Base laser scanner with rotational platform (0.05 mm accuracy) and in two pieces. The point cloud was post processed for holes and error with a reverse engineering tool suite. This part contains small features on the head and thread at the bottom. As we do not control the PCS density, a sufficient number of PCS is not obtained in those regions around small features. To address this issue, currently we obtain PCS on small features separately, with the user specifying

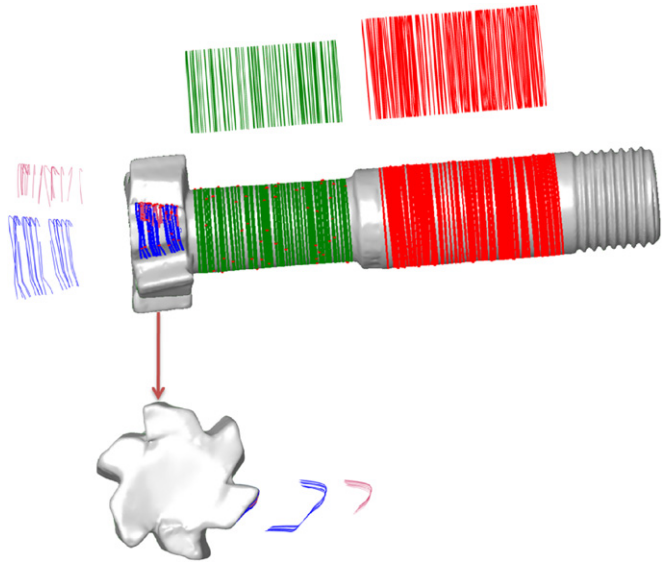


Fig. 24. Sets obtained on mesh bolt part obtained from scanned point cloud data.

the target locations and then add them to the PCS obtained in the other regions before clustering. Fig. 24 shows combined results obtained by selecting small region on the top (Blue and brown colored PCS) and region at the bottom (Red and green PCS). We combined the PCS from both the results and then performed the subsequent clustering.

Our method is sensitive to sweep and a part should have sufficient evidence of a sweep for our method to extract sufficient number of locally similar PCS in a region. Boolean operations such as shells can destroy the evidence of original sweep and the result might not contain sweep evidence that can be extracted by our method. Also, even if there is sweep evidences the resultant sweep can have a totally different cross-section and trajectory as compared to the originally designed part. Therefore, our method will yield a possible design for a model or in some cases multiple designs which could be totally different from original design intent. We plan to overcome such problems through higher level user interactions for bridging the computational and design intent gaps.

Currently we introduced some level of user interaction and explored them for parts shown in Figs. 23 and 24. User interactions were used for (1) varying the density of PCS in different regions and (2) avoiding PCS creation in regions not having sweep evidence. In the future, we will adaptively control the density of PCS created in a specific region. This will help us by having sufficient number of PCS in regions having small features and also by avoiding PCS creation in the regions having no sweep evidence. Reverse modeling of physical parts poses additional challenges. The possible decompositions of a physical part are neither obvious nor unique and the underlying shape may be concealed by noise in the scanning process and(or) by added-on details. For example in Fig. 23, two sets of sweeps are obtained for the same region in the part suggesting two possible decompositions of the geometry. This issue can be resolved by including the user in the reverse modeling process, allowing the user to choose the best set for recreating the CAD model. The time complexity of different steps are presented in previous sections. The combined complexity of all the steps is the complexity of the slowest step, which is the last step to obtain clusters of points $O(N^2 \log N)$.

7.3. CAD model reconstruction

In this section, we describe the use of PCS to reconstruct a CAD model of a gas turbine airfoil (Fig. 25) in order to exemplify its reverse CAD modeling capabilities. When represented as a CAD model, the primary geometry of an airfoil can be generalized as a single sweep segment. It is imperative that the cross sectional curves, incrementally distributed along the sweep direction of the airfoil body, be provided to a CAD modeler while developing such a representation. These cross sections ideally lie on planes normal to the sweep direction of the airfoil body. The PCS extracted from the airfoil mesh meet such criteria and serve as quintessential cross sections for the ensuing CAD model. Fig. 25(b) shows the extracted sweep component of the airfoil body from its mesh. The PCS data is then exported into *CATIA*^{TM} as a set of coordinate values of points lying on the PCS. In *CATIA*^{TM}, these points are fitted with interpolation splines (each representing a PCS), and a loft operation is performed across the resulting cross-sectional splines, as shown in Fig. 25(c). During reverse modeling, *CATIA*^{TM} prefers each PCS to have equal number of points, and the points in each PCS to be evenly spaced. To facilitate this, the points in the PCS are subjected to a uniform re-sampling algorithm [40] that takes into account a user defined PCS point density value. Furthermore, the quality and reconstruction rate are significantly improved if points in adjacent PCS are approximately aligned along common iso-parms. This is also accomplished through re-sampling.

Throughout the entire reverse modeling process, the reference coordinate system is kept consistent. As a result, the orientation of the final reconstructed model (Fig. 25(c)) can be overlapped with that of the original model in *CATIA*^{TM} without having to manually register the two. Fig. 25(d) illustrates the results of an error analysis conducted using 'Distance Analysis' tool, a part of the 'Freestyle Shape Workbench' in *CATIA*^{TM}. Here, the orientation of the original reference model (Fig. 25(a)) is registered with that of the final reconstructed model (Fig. 25(c)) to determine geometric deviations occurring in the reconstructed model. The color codes indicate the extent of deviation within a specific region. The recorded deviations are less than 0.5 mm. The deviations result from the smoothening effect on PCS curves that occur during uniform re-sampling. However, the build time for the reconstructed model in this case is significantly curtailed, since the organized PCS points accommodate lofting criteria of 3D CAD modelers and require less user intervention during reconstruction. The process of exporting the PCS data, generated from a point

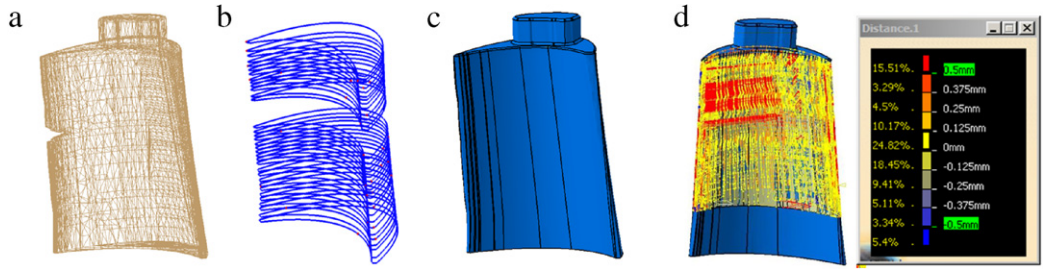


Fig. 25. AirFoil body reverse modeling. (a) Acquired point cloud and generated mesh, (b) extracted sweep component, (c) reverse modeled airfoil in *CATIA*^{TM}, (d) results of error analysis between reconstructed and original model.

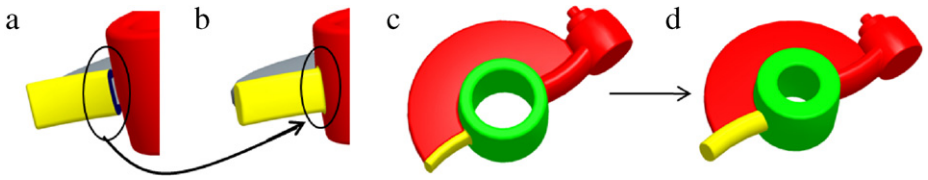


Fig. 26. Construction of CAD model. (a) During reconstruction, the features, $f1$ (yellow) and $f2$ (red) are not automatically merged (blue region). The user can interact directly to fix it (b). As the features are defined parametrically, making these changes is straightforward. (c) Completely reconstructed model. (d) For redesign, the user had reduced the diameter of the hole (green), increased width and length of the arm (yellow) parametrically. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

cloud, to an existing CAD modeler and transforming it into a coherent and fully parameterized CAD model can be automated with the use of API based macros that are specific to the CAD modeler being used. Doing so can preclude the need for any significant user input during the process, and can thus curtail the time taken to carry out the overall pipeline that transforms a point cloud to a CAD model.

For parts with multiple sweep components, each component can be reverse modeled using the above approach. The CAD model thus generated will closely approximate if not entirely the original scanned part. The problems can arise at those places where two or more swept features interact, for example in Fig. 26(a). User interaction would be required in such a scenario to merge the interacting features Fig. 26(b). As the features are defined parametrically, making these changes will be easy and straightforward for any native *CATIA*^{TM} user. The final CAD model is shown in Fig. 26(c). The generated model can now be edited for redesign. For example, if the diameter of the hole in rocker arm (Fig. 26(c)) needs to be modified to accommodate the new shaft, the user can do so by updating the associated parameter and the result is shown in Fig. 26(d).

8. Conclusion

In this paper, we have developed a method to extract volumetric information from a mesh model that has sufficient evidence of constituent swept volumes. We describe two new algorithms derived from Locally Linear Embedding (LLE) and Affinity Propagation (AP) for organizing and clustering PCS data into sweep components, which form the basic building blocks of the re-created CAD model. The LLE algorithm analyzes the cross-sections (PCS) using their geometric properties to build a global manifold in an embedded space. The AP algorithm then clusters the local cross sections by propagating affinities among them in the embedded space to form different sweep components. We have demonstrated the results on different CAD mesh models having intersecting sweep components. We have also demonstrated the reconstruction of a parameterized CAD model of an Airfoil body using *CATIA*^{TM}. The Airfoil represented a model with a single sweep component. We will also compare our method with a standard approach in terms of computation time and accuracy. We posit that our work will help in related problems such as symmetry detection, constrained deformation of scanned mesh parts, mesh model repair, remanufacturing of damaged parts, and product quality inspection. Our current method takes a triangular mesh, obtained from building connectivity amongst points on laser scanned point cloud data, as the input. Although a mesh representation allows for easier geometry processing, a significant amount of time is still spent on converting the point cloud to a mesh. The reconstruction of a part with multiple sweep components and intersections between them, poses numerous challenges which is a research topic in itself and is part of a future work. We will also extend our method to use the point cloud information directly like in [41]. Furthermore, we intend to build a benchmark of laser scanned parts represented as point clouds and to also expose further challenges in the field of reverse 3D modeling.

Acknowledgements

This material is based upon work supported by the National Science Foundation Division of Information and Intelligent Systems (NSF IIP) under Grant No. 0917959 for 3DHub. Earlier work was supported by General Electric$^{(R)}$. Any opinions, findings,and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Vrady T, Martin RR, Cox J. Reverse engineering of geometric models—an introduction. Computer-Aided Design 1997;29(4):255–68. doi:10.1016/S0010-4485(96)00054-1. Reverse Engineering of Geometric Models; URL http://www.sciencedirect.com/science/article/B6TYR-407P9G8-2/2/aa4777d3669980f8d07b01e8937baf15.

[2] Ye X, Liu H, Chen L, Chen Z, Pan X, Zhang S. Reverse innovative design—an integrated product design methodology. Computer-Aided Design 2008;40(7):812–27. doi:10.1016/j.cad.2007.07.006. URL http://www.sciencedirect.com/science/article/B6TYR-4PB0PS9-1/2/88d301d68836d4ba6158dee1f5454ec2.

[3] Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. Science 2000;290(5500):2323–6. doi:10.1126/science.290.5500.2323. http://www.sciencemag.org/cgi/reprint/290/5500/2323.pdf ; URL http://www.sciencemag.org/cgi/content/abstract/290/5500/2323.

[4] Frey BJ, Dueck D. Clustering by passing messages between data points. Science 2007;315(5814):1136800–97. doi:10.1126/science.1136800. URL http://www.sciencemag.org/cgi/content/abstract/1136800v1.

[5] Abdel-Malek K, Jingzhou Y, Blackmore D, Ken J. Swept volumes: fundation, perspectives, and applications. International Journal of Shape Modeling 2006;12(1):87–127. URL http://search.ebscohost.com/login.aspx?direct=true\&db=aph\&AN=21498051\&site=ehost-live.

[6] Cox T. M. Multidimensional scaling. London: Chapman and Hall; 1994.

[7] Tenenbaum JB, Silva VD, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science 2000;290(5500):2319–23. doi:10.1126/science.290.5500.2319. http://www.sciencemag.org/cgi/reprint/290/5500/2319.pdf ; URL http://www.sciencemag.org/cgi/content/abstract/290/5500/2319.

[8] Sellamani S, Muthuganapathy R, Kalyanaraman Y, Murugappan S, Goyal M, Ramani K, et al. Pcs: prominent cross-sections for mesh model. Computer Aided Design and Applications 2010;7(4):601–20. URL https://engineering.purdue.edu/PRECISE/Publications/PCSProminentCrossSectionsforMeshModels/CADA-ProminentCrossSections-Final.pdf.

[9] Petitjean S. A survey of methods for recovering quadrics in triangle meshes. ACM Computing Surveys 2002;34(2):211–62. http://doi.acm.org/10.1145/508352.508354.

[10] Vrady T, Facello MA, Terk Z. Automatic extraction of surface structures in digital shape reconstruction. Computer-Aided Design 2007;39(5):379–88. doi:10.1016/j.cad.2007.02.011. URL http://www.sciencedirect.com/science/article/B6TYR-4N7RWBD-1/2/7e09e0f5289432e41098f00d3e4de3eb.

[11] Yang P, Schmidt T, Qian X. Direct digital design and manufacturing from massive point-cloud data. Computer-Aided Design and Applications 2009;6(5):685–99. URL http://www.sciencedirect.com/science/article/B6TYR-4PB0PS9-1/2/88d301d68836d4ba6158dee1f5454ec2.

[12] Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3D mesh segmentation. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 2009;28(3).

[13] Lavou G, Dupont F, Baskurt A. A new cad mesh segmentation method, based on curvature tensor analysis. Computer-Aided Design 2005;37(10):975–87. doi:10.1016/j.cad.2004.09.001. URL http://www.sciencedirect.com/science/article/B6TYR-4DNW4GK-1/2/508a457e680b94fe8bae496f7dc8ae27.

[14] Attene M, Falcidieno B, Spagnuolo M. M.: hierarchical mesh segmentation based on fitting primitives. The Visual Computer 2006;22(3):181–93.

[15] Shamir A. A survey of mesh segmentation techniques. Computer Graphics Forum 2008;27(6):1539–56. URL http://dx.doi.org/10.1111/j.1467-8659.2007.01103.x.
[16] Stamati V, Fudos I. A feature based approach to re-engineering objects of freeform design by exploiting point cloud morphology. In: Proceedings of the 2007 ACM symposium on solid and physical modeling. SPM'07, New York, NY, USA: ACM; 2007. p. 347–53.
[17] Weiss V, Andor L, Renner G, Várady T. Advanced surface fitting techniques. Computer Aided Geometric Design 2002;19:19–42.
[18] Mortara M, Patan G, Spagnuolo M, Falcidieno B, Rossignac J. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. Algorithmica 2003;38(1):227–48. http://dx.doi.org/10.1007/s00453-003-1051-4.
[19] Mortara M, Patanè G, Spagnuolo M, Falcidieno B, Rossignac J. Plumber: a method for a multi-scale decomposition of 3*d* shapes into tubular primitives and bodies. In: SM'04: proceedings of the ninth ACM symposium on solid modeling and applications. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association; 2004. p. 339–44.
[20] Agathos E, Pratikakis I, Perantonis S, Sapidis N, Azariadis P. 3*d* mesh segmentation methodologies for cad applications. Computer Aided Design and Applications 2007;4(6):827–41.
[21] Zhang Y, Paik J, Koschan A, Abidi MA. A simple and efficient algorithm for part decomposition of 3*d* triangulated models based on curvature analysis. In: Proceedings of the international conference on image processing, III. 2002. p. 273–6.
[22] Zuckerberger E, Tal A, Shlafman S. Polyhedral surface decomposition with applications. Computers and Graphics 2002;26(5):733–43. doi:10.1016/S0097-8493(02)00128-0.
URL http://www.sciencedirect.com/science/article/B6TYG-46X803H-D/2/241800c25f3f2ff79383a07f58385b83.
[23] Katz S, Leifman G, Tal A. Mesh segmentation using feature point and core extraction. The Visual Computer 2005;21(8–10):649–58.
[24] Benk P, Ks G, Vrady T, Andor L, Martin RR. Constrained fitting in reverse engineering. Computer Aided Geometric Design 2002;173–205.
[25] Li M, Langbein FC, Martin RR. Detecting design intent in approximate cad models using symmetry. Computer-Aided Design 2010;42(3):183–201.
[26] Shapira L, Shamir A, Cohen-Or D. Consistent mesh partitioning and skeletonisation using the shape diameter function. The Visual Computer 2008;24:249–59.
[27] Ke Y, Fan S, Zhu W, Li A, Liu F, Shi X. Feature-based reverse modeling strategies. Computer-Aided Design 2006;38(5):485–506. doi:10.1016/j.cad.2005.12.002.
URL http://www.sciencedirect.com/science/article/B6TYR-4JHMS4G-1/2/184143068fd4ac741c00c72469dc75da.
[28] Yilmaz O, Gindy N, Gao J. A repair and overhaul methodology for aeroengine components. Robotics and Computer-integrated Manufacturing 2010;26:190–201.
[29] Amato N, Burchan O, Lucia B, Dale K, Jones C, Vallejo D. Choosing good distance metrics and local planners for probabilistic roadmap methods. IEEE Transactions On robotics and Automation 2000;16(4):442–7.
[30] Mitra NJ, Guibas LJ, Pauly M. Partial and approximate symmetry detection for 3*d* geometry. ACM Transactions on Graphics 2006;25(3):560–8. http://doi.acm.org/10.1145/1141911.1141924.
[31] Pauly M, Mitra NJ, Wallner J, Pottmann H, Guibas LJ. Discovering structural regularity in 3*d* geometry. In: SIGGRAPH'08: ACM SIGGRAPH 2008 papers. New York, NY, USA: ACM; 2008. p. 1–11. http://doi.acm.org/10.1145/1399504.1360642.
[32] Arun KS, Huang TS, Blostein SD. Least-squares fitting of two 3-*d* point sets. IEEE Transactions on Pattern Analysis and Machine Intelligence 1987;9(5):698–700. http://dx.doi.org/10.1109/TPAMI.1987.4767965.
[33] Seber G. Multivariate observations. New York: J. Wiley and Sons; 1984.
[34] Wobbrock JO, Wilson AD, Li Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In: UIST'07: proceedings of the 20th annual ACM symposium on user interface software and technology. New York, NY, USA: ACM; 2007. p. 159–68. ISBN: 978-1-59593-679-2 http://doi.acm.org/10.1145/1294211.1294238.
[35] Hui K, Wang C. Clustering-based locally linear embedding. In: Pattern Recognition, 2008. ICPR 2008. 19th International Conference on; 2008, p. 1–4. doi:10.1109/ICPR.2008.4761293.
[36] Agathos E, Pratikakis I, Perantonis S, Sapidis N, Azariadis P. Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley symposium on mathematical statistics and probability; 1967, p. 281–97.
[37] Wang K, Zhang J, Li D, Zhang X, Guo T. Adaptive affinity propagation clustering. CoRR, 2008 abs/0805.1096.
[38] Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY. An efficient *k*-means clustering algorithm: analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence 2002;24(7):881–92. http://dx.doi.org/10.1109/TPAMI.2002.1017616.
[39] Fudos I, Stamati V, Vasilakis A. CGRG point cloud repository, Computer Science Department, University of Ioannina, Greece, 2010.
[40] Murugappan S, Sellamani S, Ramani K. Towards beautification of freehand sketches using suggestions. In: SBIM'09: proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling. New York, NY, USA: ACM; 2009. p. 69–76. http://doi.acm.org/10.1145/1572741.1572754.
[41] Kobbelt L, Botsch M. A survey of point-based techniques in computer graphics. Computers and Graphics 2004;28(6):801–14.