



Special Section on CAD & Graphics 2019

Autoencoder-based part clustering for part-in-whole retrieval of CAD models



Lakshmi Priya Muraleedharan, Shyam Sundar Kannan, Ramanathan Muthuganapathy*

Advanced Geometric Computing Lab, Department of Engineering Design, Indian Institute of Technology Madras, Chennai 600036, India

ARTICLE INFO

Article history:

Received 9 March 2019

Revised 28 March 2019

Accepted 29 March 2019

Available online 12 April 2019

Keywords:

Segmentation

Part-based

CAD mesh model

Hyperbolic points

Autoencoder

Unsupervised part clustering

ABSTRACT

Part-in-whole retrieval (PWR) is an important problem in the field of computer-aided design (CAD) with applications in design reuse, feature recognition and suppression and so on. Initially, we present a non-parametric (and hence threshold independent) algorithm for segmenting CAD models (represented as meshes) which does not require any user intervention. As there is no labelled segmented dataset available for part clustering, we propose the use of autoencoders, one of the approaches used in deep networks along with hierarchical clustering. The features for autoencoder is derived from the Gauss map of the segments. The autoencoder network is then trained and validated using a hierarchical clustering-based approach that generates a dictionary of labels for each segment. PWR is then done by testing a query model with the network that retrieves models having the query as their subset. Comparison of the segmentation algorithm with the state-of-the-art approaches indicate that it performs better or on par. The algorithm was also tested for noisy models. Results of the part clustering and PWR are also presented for models from a CAD dataset along with the discussions.

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

Given a query model, three-dimensional shape search (3DSS) retrieves models from a dataset that are similar to the query model. 3DSS of CAD models has become an important problem for more than a decade due to digital archiving. Though digital archiving has been made easier because of the advent of newer hardware technologies, collecting data of CAD/Engineering models is a cumbersome task. Many of them cannot be made public due to the proprietary nature of design data. In the field of CAD/Engineering models, Engineering Shape Benchmark (ESB) is a popular database [1] consisting of 801 models. Another popular database for CAD models is Regli [2] which has a couple of hundreds of models. From both of these public databases, it is evident that digital archiving of CAD models is a difficult task as opposed to graphical models, for e.g. ShapeNet (which has millions of models, see [3] for more details). Jayanti et al. [1] have also noted that multimedia / graphical model datasets are not well suited for engineering domain because engineering models have high genus, discontinuities like sharp edges and vertices, assembly of individual parts etc.

3DSS is a very important problem for engineering design process such as fetching right information for design is an arduous task and designers spend over 60% of their time searching for them [4]. To aid the task of searching, shape signatures have been developed. Most of the shape signatures are tuned for global search that retrieve models that are globally similar. Iyer et al. [5] have demonstrated the capabilities of various global shape signatures for the retrieval of CAD models. There are a few shape signatures that are popular in the area of graphical models, e.g., Osada's shape distributions [6], Fang and Wong [7]'s heat-kernel approach, rotation invariant spherical harmonic representation (SHR) by Kazhdan et al. [8], etc. Both [6,7] are global signatures, where as SHR [8], as they have mentioned, does not work well for axis-aligned objects, and Engineering (CAD) models are majorly axis-aligned.

Gunn [9] has observed that about 40% of the new designs could be built from an existing design and 40% from modifying an existing design. Ullman [10] has indicated that a large percentage (75% or sometimes, more than that) of design reuses existing knowledge for the new product development. Considering this, retrievals that are locally similar rather than globally similar ones would be more useful. Part-in-whole retrieval (PWR), where a part is given as a query model and it retrieves models that contains the query part would be a valuable addition for the design process.

PWR problem typically uses a grouping-based approach which can be done using one of the following [11] - local descriptors, view-based, segmentation-based. Local descriptors use

* Corresponding author.

E-mail address: mraman@iitm.ac.in (R. Muthuganapathy).

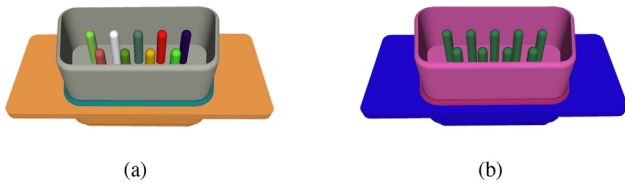


Fig. 1. (a) Segmented CAD model. (b) Classified segments of the same model.

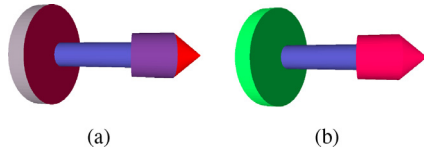


Fig. 2. Segmented CAD model. (a) Patch-based segmentation of a CAD model. (b) Part-based segmentation of the same model.

neighborhood information of a point. Finding optimal correspondences using local descriptors usually results in higher time complexity. View-based approach generates a set of 2D images from multiple viewpoints in conjunction with region-based signatures such as Zernike moment. Selecting best set of views for this approach is the usual bottle neck and various researchers have attempted to address this problem. Segmentation-based approach first decomposes a model into a set of parts/components which are then classified and then used for part retrieval. For example, Fig. 1(a) shows a segmented CAD model which are then used to classify them (Fig. 1(b)).

As retrieving engineering models need to emulate the thinking of engineers, who learn simpler concepts and compose them to represent more abstract ones [12] and also a CAD model is usually built using a set of Boolean operations on many parts termed as primitives, perhaps, segmentation-based approach is more suited for PWR of CAD models.

Segmentation, in general, can be classified into patch-based (or surface-based) and part-based (or volume-based). Patch-based segmentation yields surface patches as output (Fig. 2(a)) whereas part-based segmentation generates volumetric information (Fig. 2(b)). Traditional segmentation algorithms for graphical models (e.g., SDF [13], WC Seg [14] and for other algorithms, see [15]) have been found to be not working well for low-poly CAD model as input [16]. Also, algorithms that work for CAD mesh models require user-intervention/manual parameter setting (see Agathos et al. [17]). For applications such as PWR, it is desirable to have an automatic algorithm for part-based segmentation.

In this paper, we present an algorithm for PWR of CAD models represented as mesh. Initially, an algorithm for the part-based segmentation is proposed that requires no user input values. The algorithm uses hyperbolic points in the model along with graph theoretical approach such as strongly connected components to segment the model. With the advent of deep learning techniques, we propose an unsupervised part clustering algorithm based on Autoencoder and agglomerative clustering as there exists no database of segmented parts of CAD models with ground truth information. Autoencoder helps in reducing the size of the feature space as well. Training and validation are then performed using a hierarchical clustering approach which is eventually used in PWR. The key contributions of our approach are

- Using Hyperbolic points and Strongly Connected Components to identify segment boundaries enabling a non-parametric (and hence threshold-independent) approach for segmentation.
- An unsupervised part clustering of the segmented parts using a Gauss map based feature extraction and Autoencoder based learning method combined with Agglomerative clustering.

2. Related works

In this section, we mention works only related to segmentation, part clustering and PWR of CAD models (please see [18] for a learning-based approach for grouping and labeling in graphical models). For segmentation, we categorize the algorithms based on patch-based and part-based.

2.1. Patch-based CAD model segmentation algorithms

A clustering approach called HFP [19] detects sets of triangles that form a plane, a sphere or a cylinder. The algorithm requires the information of the number of clusters as input. Clearly, the number of segmented surfaces vary with the number of clusters. Extension of the primitive approach to thin-plate CAD models using parallel planes has been presented in [20]. Gauss map and Hough transform based clustering for finding the sparse and dense regions has been proposed in [21] by Xiao et al. for segmenting CAD models.

Curvature tensor based segmentation by Lavoe et al. [22] converts the original triangulated CAD mesh model into a set of patches represented by subdivision surfaces. Gao et al. [23], presented an approach for feature suppression where the CAD mesh model is segmented by an improved watershed segmentation algorithm, constructing the region-level representation required by feature recognition. Sunil and Pande in [24], has developed a method to segment free form surfaces like sheet metal into various regions by using various geometric properties such as curvature etc. and recognizing a variety of protrusion and depression features.

2.2. Part-based CAD model segmentation algorithms

Zhang et. al [25] uses a Gaussian curvature based threshold approach for segmenting into parts. Perception-based approach, proposed in [26] uses a threshold-based minimum/normal curvature approach for part-based segmentation of CAD models. Using minima of electrical charge density distribution, an algorithm for part-based segmentation has been proposed in [27]. Other algorithms [28–30] also demonstrated the segmentation for CAD models but required some amount of user intervention.

2.3. Classification of CAD models

A neural network approach to classify 3D prismatic parts has been proposed in [31] based on the projected views from a model. The research presented in [32,33] perform classification of the CAD Models using histogram representation and a k -Nearest Neighbour (k NN) classifier. A classification approach based on Support Vector Machines (SVMs) has been presented in [34] using surface curvatures. The idea of SVMs was also used by Hou et al. [35] along with a combination of moment invariants, principal moments and geometric ratios. Jayanti et al. [1] have tested various signatures for the classification of CAD models. Though neural network or learning-based approaches have been in place, application of deep learning to CAD models is very rare (except [12] which focuses on whole CAD model).

2.4. Part-in-whole Retrieval (PWR) of CAD models

Using multilevel decomposition of models via. HFP [19], a thesaurus-based approach for PWR has been presented in [36]. PWR in [37,38] use graph construction for B-Rep CAD models along with segmentation. Partial retrieval of B-Rep CAD models in [39,40] use face adjacency graph (FAG) and patch-based segmentation. Graph matching is an NP-complete problem and hence usually inefficient. Almost all the methods use B-Rep as input for

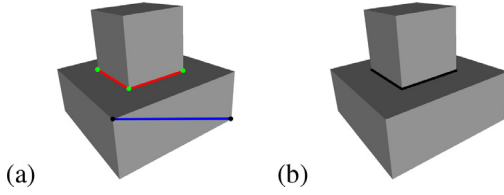


Fig. 3. Illustration of basic concepts (a) Hyperbolic Vertices (green), Hyperbolic Edges (red) and Flat edges (blue). (b) Segment Boundary (black edges). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

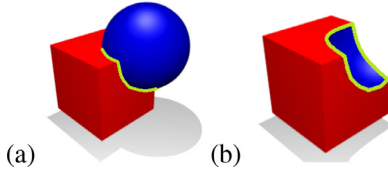


Fig. 4. Concave discontinuity formed (green boundary) (a) Union of two parts. (b) Difference of two parts. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

retrieval of CAD models and have to rely on patch-based segmentation.

3. Segmentation of CAD mesh models

3.1. Basic concepts

1. Concave edge (E_C): An edge is concave if the angle between two of its incident triangles is less than π (angle measured from the outside).
2. Hyperbolic vertex (V_H): A vertex V with Gaussian Curvature $K < 0$ in the input mesh model as shown in Fig. 3(a) with green dots.
3. Hyperbolic edge (E_H): A concave edge connecting two hyperbolic vertices in the mesh model as shown in Fig. 3(a) in red.
4. Flat edge: An edge is a flat edge if its two incident triangles lie on the same plane as shown in Fig. 3(a) in blue.
5. Segment Boundary: A boundary between two segments from a CAD mesh model (black edge in Fig. 3(b)).

3.2. Characterizing the segment boundaries

In this section, we discuss the motivation for characterizing the vertices of a mesh using hyperbolic points for finding segment boundaries, which in turn guides the proposed segmentation algorithm.

In Constructive Solid Geometry (CSG), CAD models are built using Boolean operations such as union (\cup), intersection (\cap) and difference ($-$). When two parts intersect they form a contour of concave discontinuity as per transversality regularity [41]. In the case of join operation (\cup) between two components, the concave discontinuity actually forms the boundary between the two parts as shown in Fig. 4(a) (green boundary). The vertices in this region have one of their principal curvature (k_{\min}) negative because of this concave discontinuity, but the other principal curvature (k_{\max}) is positive because of the convexity in the original part. Since k_{\max} and k_{\min} of the vertices in this region are of opposite signs, the vertices in this region becomes hyperbolic vertices.

Similarly, when difference operation ($-$) is performed between two parts, some region from a part is removed (as shown in Fig. 4(b) (green boundary)). This will also give rise to concavity and

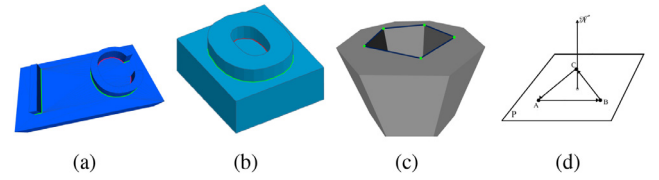


Fig. 5. (a) Hyperbolic edges in the segment boundary - green color and Non-Hyperbolic edges in the segment boundary - pink color. (b) Hyperbolic edges in the segment boundary - green color and Non-Hyperbolic edges in the segment boundary - pink color. (c) Edges connecting two hyperbolic points within the same segment (blue edges). (d) Normal N of the supporting plane P of the $\triangle ABC$. The side of the normal N gives the region above the plane P . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the points in that region become hyperbolic. All the triangles incident on these hyperbolic vertices lie on the same part and there is no segment boundary. Hence, all the hyperbolic vertices need not lie on a segment boundary.

When a convex polygon is extruded with a positive depth (as shown in Fig. 5(a)), it is similar to the union of two parts. However, when the sketch profile extruded is a concave polygon ('C' as shown in Fig. 5(a)), all the vertices in the segment boundary are not hyperbolic (the vertices in outer curve becomes hyperbolic vertices).

In case of extruding a profile with an all-concave inner edges (Fig. 5(b)), it can be noted that the inner set of edges is not hyperbolic and the outer set of edges is hyperbolic. In order to prevent over-segmentation we use the condition that atleast one edge on the boundary loop has to be a hyperbolic edge. In the case of extrusion with a negative depth (Fig. 5(c)), all hyperbolic vertices in the model need not lie on a segment boundary. Hence, in our approach, we use hyperbolic points in conjunction with concavity.

3.3. Extraction of segment boundaries and segmentation

In this section, the boundaries between two parts which guide the segmentation process are identified. Initially, from the input CAD mesh model, the principal and the Gaussian curvatures at all the vertices are computed (using CGAL [42]) and the hyperbolic vertices V_H are identified. The paper makes few assumptions on the input models such as: they have consistent orientations, they do not have seams along the hyperbolic edges and that they do not have self intersections.

3.3.1. Finding the boundary edges

In order to find the segment boundaries, the set of edges which lie on the boundary region are needed. The set of edges that characterize the boundary are divided into concave edges and hyperbolic Edges. The edges in the input CAD mesh model which connect two hyperbolic vertices are found. Then, it is checked whether the edges lie in a region of concave discontinuity.

In order to check whether an edge lies in a region of concave discontinuity, the orientation of the two triangles (T_1 and T_2) incident on that edge are used. If a supporting plane P is constructed using one of the triangles (T_1), then the other triangle (T_2) lies either above, below or on the plane. In Fig. 5(d), a plane P is constructed using $\triangle ABC$. Let N be the normal for the plane. A triangle lies above the plane, if it entirely lies in the side of N and the triangle lies below the plane, if it lies on the side opposite to the direction of N .

3.3.2. Segment boundaries from hyperbolic edges

Only when the triangle T_2 lies entirely above the plane P , a concavity is formed between the two triangles. When there is a concavity between the two triangles, the common edge between two

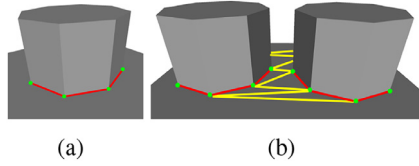


Fig. 6. (a) Edges connecting two hyperbolic points forming segment boundary (red edges). (b) Edges connecting two hyperbolic points between two segments (yellow edges). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

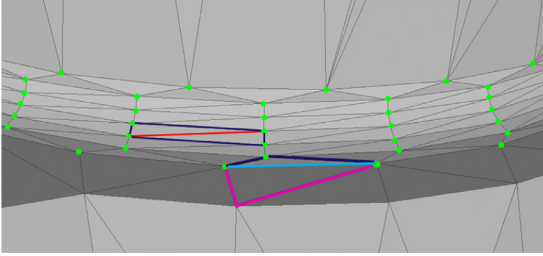


Fig. 7. Hyperbolic points (green dots) on a filleted region. Red edges are hyperbolic edges but non-candidate edges. Blue edges are hyperbolic edges which are candidate edges. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

triangles lies on a region of concave discontinuity and the edge is a part of a segment boundary. In Fig. 6(a), the edges corresponding to this concave region are highlighted in red and they define a segment boundary. All such edges are hyperbolic edges and are assigned to a new edge set E_H (a set of hyperbolic edges).

When two segments are close by, there might be edges in the input mesh which directly connect the two segments (yellow edges in Fig. 6(b)). Though the end points of those edges are hyperbolic vertices, it is evident that they do not form any segment boundary, since they connect two independent segments. As the edges connecting two independent segments are non-concave, these edges are not added to E_H . It can be noted that this condition is applicable even when the base surface is a curved one.

Similarly, the blue edges in the Fig. 5(c), connect two hyperbolic points but the triangles incident on these edges belong to the same segment. Hence, when the triangle T_2 entirely lies below the supporting plane P constructed using the triangle T_1 , the edge is not assigned to E_H , since that edge does not correspond to a segment boundary. The triangles T_1 and T_2 belong to the same segment in this case.

3.3.3. Segment boundaries of concave regions

In case of filleted regions that are concave, all the vertices on the region will be hyperbolic points (green dots in Fig. 7). If all hyperbolic edges are considered as candidate edges E_{can} , then it will generate an oversegmented region. The edges connecting the filleted region to the base region has to be identified. Such edges will have a set of hyperbolic edges towards only one side. From the set of concave edges E_C , select the edges with its two incident vertices which are hyperbolic vertices V_H . In each of those edges, find the two incident triangles and consider all its six edges. If all of them are hyperbolic edges E_H , then those edges belong to a curved region and has to be avoided. In case if out of the six edges on two incident triangles, two of those edges are non-hyperbolic, then the concave edge in consideration will be a candidate edge E_{can} . In Fig. 7, considering the red edge in a filleted region which is concave, its two adjacent triangles are highlighted with blue edges. All the six edges arising from those two triangles are hyperbolic (blue and red edges). Hence the red edge will not be added to E_{can} . The cyan edge which is a concave edge has two adjacent trian-

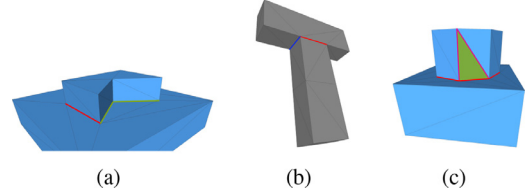


Fig. 8. (a) The hyperbolic edges are shown in red and the edges corresponding to the concave region of the segment boundary found are shown in green. (b) Incomplete Loop: The blue edge highlight the unused concave edge and the red edge highlight the flat edge part of the expected boundary. (c) Region Growing Process: The red edges highlight the segment boundary. The seed triangle picked is highlighted in green. The edges highlighted in pink are the edges enqueued with respect to the seed triangle. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

gles, only four vertices are hyperbolic (edges highlighted in purple and cyan) and two edges are non-hyperbolic (edges highlighted in pink), hence the cyan edge will be added to E_{can} .

3.3.4. Segment boundaries from strongly connected components SCCs

All the hyperbolic edges E_H existing in the mesh model will be added to the set of candidate edges E_{can} . From all such candidate edges E_{can} , a directed graph G is constructed. The direction of the edges for the construction of G are taken from a subset of half edges [43] in the input model. The direction of half edges in the graph G is chosen such that only one of the half edge of every edge is selected in a consistent direction.

From the directed graph G , various strongly connected components SCCs present are found. Since all the segment boundaries are disconnected and all the edges in a segment boundary are connected, the SCCs give the segment boundaries E_{SB} .

3.3.5. Segment boundaries from solitary edges

Even though Section 3.3.3 delineates a procedure to extract segment boundaries, there are a few concave and hyperbolic edges that are not part of any strongly connected component could potentially form further segment boundaries.

For example, in Fig. 8(a), a concave sketch profile is extruded over the base block with a positive depth. Not all the edges along the likely segment boundary are hyperbolic edges. Hence a strongly connected component is not formed. The set of such edges which are not part of any strongly connected component are defined as Solitary Edges E_{SO} (red and green edges in Fig. 8(a)). In the figure, the red edges are concave edges with its two incident vertices being hyperbolic.

In order to find segment boundaries from such edges, the following procedure is adopted: From the input model, the set of concave edges E_C and hyperbolic edges E_H which are already not a part of any previously found strongly connected components are included in E_{SO} . Once the set of solitary edges E_{SO} is formed, we construct a directed graph $G(V, E_{SO})$ (using the half edges in the same manner as explained in Section 3.3.4), and the SCCs existing in the graph are found. From the identified SCCs, only those loops which has atleast one hyperbolic edge is added to E_{SB} to prevent over segmentation (as explained in Section 3.2, Fig. 5(b)).

In Fig. 8(a), E_C is shown in green and E_H in red. All of these edges which were not considered previously would be added to E_{SO} . These edges would form segment boundary E_{SB} after the SCC is extracted from $G(V, E_{SO})$. With these edges in E_{SB} they define an additional set of segment boundaries.

3.3.6. Segment boundaries from incomplete loop

In some of the cases, although some concavity occur at the boundary of the region, it may happen that the boundary region

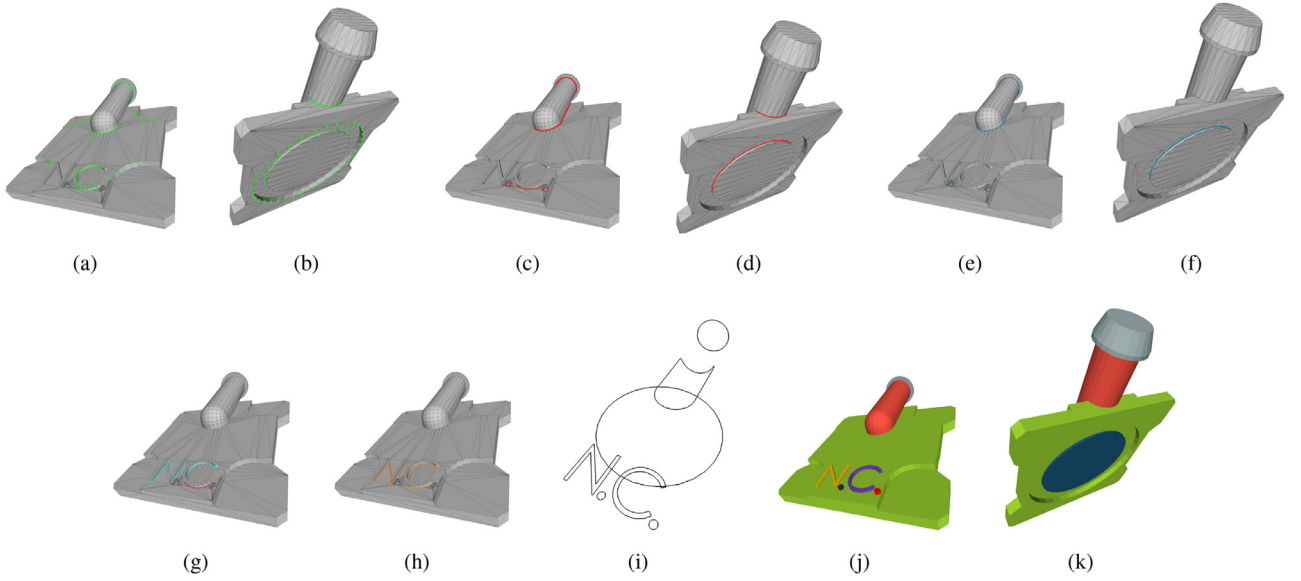


Fig. 9. Illustration of the Algorithm. (a) V_H in green. (b) V_H in green. (c) Hyperbolic edges in red. (d) Hyperbolic edges in red. (e) Blue edges indicate strongly connected component. (f) Blue edges indicate strongly connected component. (g) Solitary edges in pink, Concave edges in cyan. (h) Strongly connected component using E_{SO} in orange. (i) Segment boundaries. (j) Segmented output. (k) Segmented output. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

may not be entirely concave. Hence part of the supposed boundary may lie on a flat region attached to the base segment. In order to separate such regions we need to find out the unused hyperbolic edges until the current stage and make use of them as a cue to the boundary extraction. For example in Fig. 8(b), the blue edges are the unused concave edges and red edges are flat edges part of the expected boundary. Here the set of red edges and blue edges combined will form a segment boundary. Let E_{unhyp} be such unused hyperbolic edges. We construct planes P_{sup} from the adjacent facets of all E_{unhyp} . Then we find all the edges lying on P_{sup} satisfying few criteria viz the edges shall not be a convex edge, it has to be a flat edge and both the facets incident to the edge should not lie on the supporting plane simultaneously. A list of all such edges are found and another graph is constructed to form an SCC which will find the segment boundaries. All such boundaries found are added to the universal segment boundary list.

3.4. Extracting the segments

After the segment boundaries are identified from the strongly connected components, the various segments are found using a flooding-based region growing method. In order to find the triangles corresponding to a segment, a seed triangle T_s incident on an edge in a strongly connected component is picked. In Fig. 8(c), the segment boundary is highlighted in red and the seed triangle picked is highlighted in green. From T_s , the segment is expanded through the edges using a queuing approach. The edges which are part of any strongly connected component are not enqueued, ensuring that the segment does not grow beyond the segment boundaries. The edges highlighted in pink in Fig. 8(c), incident on the seed triangle are enqueued. The region growing process is continued using the triangles incident on the edges enqueued and the triangle which are already a part of that segment are not considered.

This process sub-divides the model into two segments. The segmentation process is repeated iteratively for all the identified strongly connected components, yielding various segments in the model. The pseudo-code of the algorithm is given in Algorithm 1.

3.5. Illustration of segmentation algorithm

A step by step illustration of the working of our algorithm using a sample CAD mesh model is given in Fig. 9. The Gaussian curvature for all the vertices in the input model are computed and the hyperbolic vertices V_H are identified (green vertices in Fig. 9(a) and 9(b) (Algorithm 1 step 2)). From these hyperbolic vertices, the hyperbolic edges E_H are found (edges in red in Fig. 9(c) and 9(d) (Algorithm 1 step 6)). These hyperbolic edges E_H are used to find strongly connected components (blue edges in Fig. 9(e) and 9(f) (Algorithm 1 step 8)) which defines the segment boundaries. It can be noticed that not all the hyperbolic edges are a part of the previously found strongly connected components. These edges are classified as Solitary edges E_{SO} (pink in Fig. 9(g) (Algorithm 1 step 9)). The strongly connected components found using E_{SO} are shown in orange colored edges in Fig. 9(h) (Algorithm 1 step 10). A list of strongly connected components which define the segment boundaries are found (Fig. 9(i) (Algorithm 1 step 14)). From these segment boundaries, the model is segmented using a flooding-based region growing method. The segmented result of our algorithm for the given input model is shown in Fig. 9(j) and 9(k) (Algorithm 1 step 28).

4. Clustering of segments

An autoencoder [44] is trained with the subparts which are segmented from a CAD model. The feature representation of the segments of each CAD model are fed into an autoencoder network to find a compressed representation of the same. We are proposing a Gauss map [45] based feature extraction method to extract features from each segment and feed it into the network. The part clustering section of the Fig. 10 describes the entire process.

4.1. Gauss map based feature extraction

The segments could be aligned in various orientations in their original models. Hence a uniform representation of these parts has to be found to make them orientation invariant. It has been shown in [8] that principal axis based approaches are more suited for

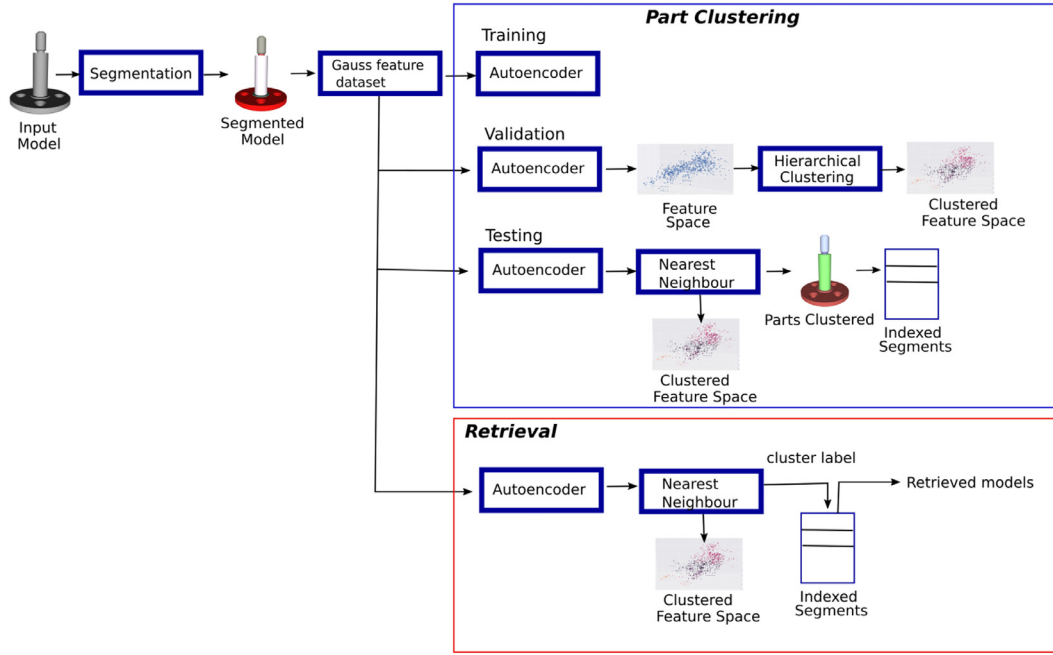


Fig. 10. Block diagram for training the model, constructing bag of labels and part-in-whole retrieval.

Algorithm 1: CAD Segmentation Algorithm.

Input: Input mesh data, $M = \{V, E\}$.
Output: Segmented mesh $S = \{S_1, S_2, \dots, S_N\}$.

- 1: Compute $K \forall V \in M$.
- 2: $V_H \leftarrow$ hyperbolic vertices, $E_C \leftarrow$ Concave edges.
- 3: $E_H \leftarrow \{e_{i,j}\}$ incident on V_{H_i}, V_{H_j} and a concave edge.
- 4: For each E_C , extract six e 's of two incident Δ s
- 5: **if** one of Δ_1 or Δ_2 has $|E_H| < 2$ **then**
- 6: $E_{Can} \leftarrow E_C$
- 7: **end if**
- 8: $E_{SB} \leftarrow$ SCC of $G(V_H, E_{Can})$
- 9: $E_{SO} \leftarrow E_C - E_{SB}$
- 10: $E_{SB} \leftarrow$ SCC of $G(V, E_{SO})$
- 11: For each $e \in E_{unhyp}$ construct supporting planes P_{Sup} from its incident Δ s
- 12: $E_{Sup} \leftarrow$ all e lying on $P \in P_{Sup} \mid e$ is flat and non-convex
- 13: $E_{SB} \leftarrow$ SCC of $G(V, E_{Sup})$
- 14: $E_B \leftarrow \bigcup_{i=1}^{|E_{SB}|} E_{SB} \mid \text{atleast one } E \text{ in } E_{SB} \text{ is } E_H$
- 15: **for each** $scc \in SCC$ **do**
- 16: $S' \leftarrow$ segment where scc belongs
- 17: $S_i \leftarrow \emptyset$ and $S_{new} \leftarrow \emptyset$
- 18: Pick Δ_{seed} incident on an E in scc
- 19: $edges \leftarrow E \in \Delta_{seed}$
- 20: **for each** $edge \in edges$ **do**
- 21: **if** $edge \notin E_i$ **then**
- 22: Flood through Δ s incident on $edge$
- 23: **end if**
- 24: **end for**
- 25: $S_{new} \leftarrow \Delta$ s reachable from Δ_{seed}
- 26: $S' \leftarrow S' - S_{new}$ and $S_i \leftarrow S_i + S_{new}$
- 27: **end for**
- 28: **return** $S = \bigcup_{i=1}^{|E_B|+1} S_i$

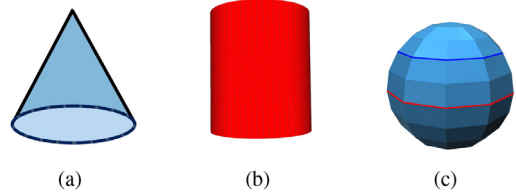


Fig. 11. Gauss map of corresponding shapes (a) Conical sub-part (b) Cylindrical sub-part (c) Mapping of face normals on Gauss map. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

axis-aligned models. Motivated by that statement, we use the following procedure; In order to transform each part we calculate the oriented bounding box of each sub-part and find the longest axis of this box. The angle θ between the longest axis and Z axis is found and the object is rotated by θ thereby aligning the major axis of each part to Z axis. The property that the oriented bounding box is to rest on its largest face is also satisfied. The aligned model might face away from or towards the origin after this processing. So in order to preserve uniformity the part boundary (if single) is always made to face the origin. If there are two boundaries then the largest loop is made to face the origin. Thus all the segments across all the models are uniformly aligned before building the training dataset.

Given a surface X lying in R^3 , the Gauss map is a continuous map $N: X \rightarrow S^2$ such that $N(p)$ is a unit vector orthogonal to X at p , namely the normal vector to X at p . Every facet in the segment is mapped to a facet in discrete version of the Gauss map, sampling with 102 vertices as shown in Fig. 11(c). Since the variability of the CAD part is less compared to organic shapes, a $102(=10 \times 10 + 2)$ sampling of the gauss sphere would be sufficient to capture the geometry. A k-d tree is formed with the Gauss Map and every facet normal in a segment is mapped to the points in the Gauss map using the nearest neighbor algorithm. A $[102 \times 1]$ feature vector is created for every segment which denotes the Gauss sphere vertices and once a facet is mapped to a vertex, the area of the facet is added to its corresponding position in the vector. The cone

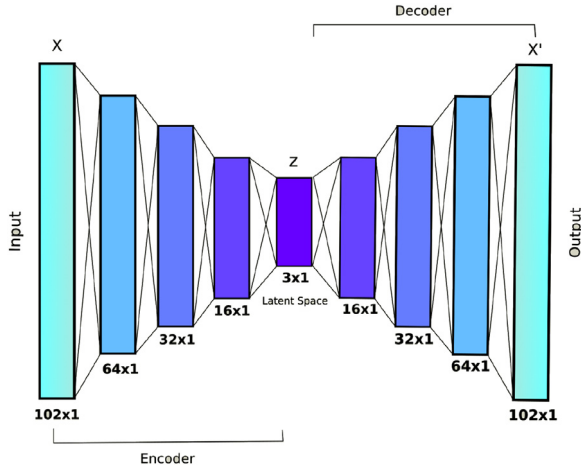


Fig. 12. Architecture of the autoencoder network.

in Fig. 11(a) is mapped to the blue color in Gauss sphere and the cylinder in Fig. 11(b) is mapped to red color. Finally the feature vector is normalized to get a representation which gives the percentage of area on the surface oriented to each direction. Later the features are normalized globally. Every segment is thus represented with a $[102 \times 1]$ feature vector.

4.2. Autoencoder-based feature mapping

The feature vectors of each segments are individually fed into an autoencoder for training. Autoencoders are simple learning circuits which aim to transform inputs into outputs with the least possible amount of distortion. The architecture of the proposed autoencoder is as described using the Fig. 12. The hidden layers will compress the input data with a better representation than the raw data. Stacked autoencoder has several hidden layers. However, the number of hidden layers is always experiential. An autoencoder neural network is a typical unsupervised learning algorithm to train the output to be equal to the inputs. Our autoencoder architecture has seven hidden layers in which the fourth hidden layer is used to find the latent space representation Z . It has an encoder and a decoder part which tries to reconstruct the input vector X at the output layer such that:

$$\Psi : X \rightarrow Z \quad \text{and} \quad \Phi : Z \rightarrow X \quad (1)$$

$$\Psi, \Phi = \arg \min_{\Psi, \Phi} \|X - (\Psi \circ \Phi)X\|^2 \quad (2)$$

4.2.1. Training the network

The hyperparameters which are to be predefined before training the autoencoder are code size, number of layers, number of nodes per layer, and the loss function. The autoencoder network has seven hidden layers and an input and output layer each for the input vector and the reconstructed vector. Both the encoder and decoder are fully connected neural networks. The number of nodes in each layer is defined as in Fig. 12. The arctan (\tanh) is used as the activation function. $[3 \times 1]$ is the code size. Mean squared error (MSE) is used as the loss function since the outputs are real-valued and Adam optimization [46] is used to minimize the loss function. Other hyperparameters involved are learning rate, number of epochs and batch size. The learning rate scales the weight vector in order to minimize the loss function. A learning rate of 0.005 is used and the value has been determined through experimentation. Epochs are the number of times an entire data set is passed to the network for training and 100 as the number of epochs provided

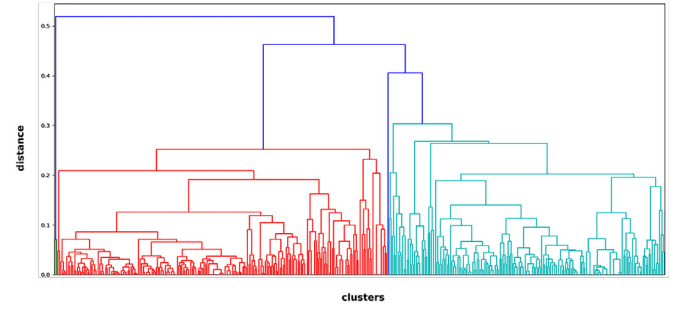


Fig. 13. Dendrogram constructed for hierarchical clustering.

the best results. The batch size is set as 1. The training dataset is created using our segmentation algorithm which is a subset of various standard datasets which consists of number of segments > 1 . We constructed a segmented dataset which was an outcome of our segmentation algorithm consisting of 156 models. The segment dataset contains 623 parts which are segments extracted from the models in ESB. Now a training dataset of size 102 is constructed by manually selecting models such that they are unique in their geometry. Rest of the models are used for validating the algorithm. Finally, the part clustering is performed on this entire set.

4.3. Hierarchical clustering on feature space

The Autoencoder will reduce the feature vector into a lower dimensional feature space of $[3 \times 1]$ vector size. Now we have to classify the feature vector of each part in this feature space. Hierarchical clustering method is for part clustering in a bottom-up agglomerative fashion. In agglomerative method each individual part begins as a separate cluster and at each step clusters are merged together based on a distance measure. The linkage criteria determine the metric used for the merge strategy. The metric used for linkage in our approach is *Euclidean* and the linkage is *Average*, which minimizes the average distance of each observation in the two sets. This merging will stop when it reaches a threshold value τ given by the user. In order to view the clusters and the inter-cluster distance we make use of a dendrogram as shown in Fig. 13 to visualise the possible clusters and fixing the τ . From experimenting with our dataset we have finalised the value of $\tau = 0.05$. Finally our feature space will be divided into N number of clusters. In our dataset used for clustering, we observed a total of 101 number of clusters. Using this clustered feature space we used the Nearest Neighbour algorithm to assign a cluster label to each subpart in a query model. Once the clustering of subparts has been done to various category, a bag of features for each model is constructed with the number of subparts belonging to each category $FS = [1, 2, \dots, N]$.

4.4. Part-in-whole retrieval

Part retrieval will consider a sub-part of any model and retrieve all the models which consist of this constituent part in it. In order to perform this operation, the part under consideration is fed to the trained autoencoder network and a reduced feature representation in $[3 \times 1]$ space is found. Now this reduced feature is mapped to the hierarchically clustered feature space of the trained data using the Nearest Neighbour algorithm. The cluster p where the part belongs to is found which is $p \in [1, 2, \dots, N]$. Using this category p , we can retrieve all the models from the indexed database, where $p > 0$ in $FS = [1, 2, \dots, p, N]$. The retrieval section of the Fig. 10 describes the process.

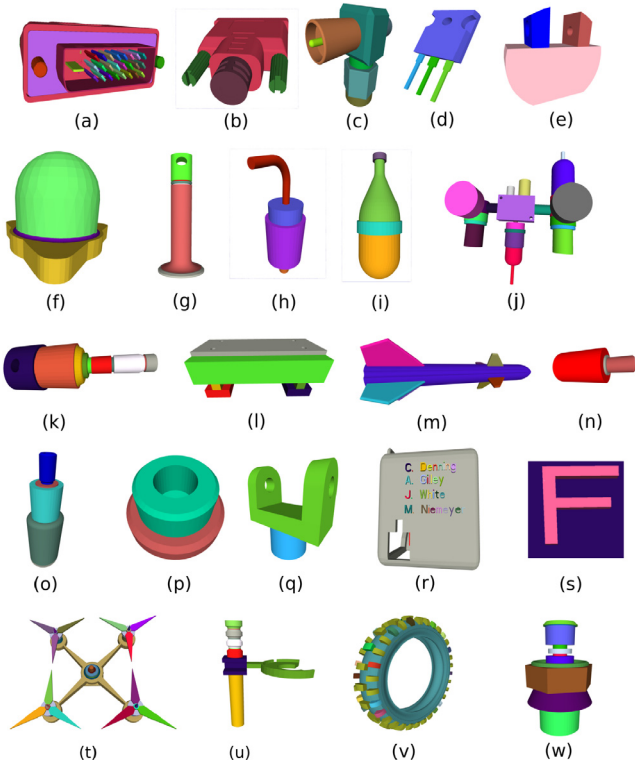


Fig. 14. Results of our segmentation algorithm.

5. Results and discussion

5.1. Segmentation

The proposed segmentation algorithm (Algorithm 1) has been implemented using C++ and CGAL [42]. The algorithm has been extensively tested on various CAD models from benchmark data sets like Engineering Shape Benchmark (ESB) [1], National Design Repository [2] and models from online repositories like GRABCAD [47].

To test the segmentation algorithm, a data set of various engineering parts and assemblies are chosen. The segments identified by our algorithm were manually verified with the segmented results based on human perception. The chosen models have various characteristics like multiple genus, non-planar boundary, fillets, tiny segments etc.

Fig. 14(a), 14(b) and 14(c) show the segmented results of a DVI connector and SMA Tee adapter. In Fig. 14(d), 14(h), 14(i) and 14(w), the segmented results of various simple models like a MOSFET, collector, pressure switch, spark plug etc. are shown.

Features like volumetric fillets are also segmented into separate segment by the algorithm (purple in Fig. 14(f)). In Fig. 14(v) the treads of the tire model are segmented. In Fig. 14(l) and 14(e), the parts with boundaries falling on flat edges are segmented. Hence we are able to demonstrate the boundary detection of parts with at least one hyperbolic edge on it.

The segmented results of various complex assemblies like a quad-copter and an air-pump has been shown in Fig. 14(t) and 14(j). The fins of the missile in 14(m) are segmented. The algorithm can segment small features like characters embossed on the model as in Fig. 14(r). In Fig. 14(s), a zoomed in portions of our result is shown. The letter embossed on the model is separated into individual segments. The results given in Fig. 14 exhibit its ability to segment the model into meaningful components.

Input mesh model	WC Segmentation	SDF	HFP	Our method
Bearing_Post_56_Back			Clusters=4 	
RedLantern			Clusters=8 	
a-arm2			Clusters=3 	
SparkPlug			Clusters=21 	

Fig. 15. Comparison of our Segmentation algorithm with other existing works.

5.1.1. Comparison with existing approaches

Fig. 15 shows the comparison of our approach with other methods. Comparison was done with the approaches where code was available publicly (SDF [13], WC Seg [14] and HFP [19]).

For models Bearing_Post_56_Back, a-arm2, RedLantern and SparkPlug the SDF segmentation tends to under-segment the model thereby making it unable to extract the prominent features of the assembly. In models Bearing_Post_56_Back and SparkPlug the HFP method tends to over-segment the model while selecting the appropriate number of clusters in order to capture the prominent parts. In RedLantern, the button is not segmented as a separate entity in WC segmentation, SDF and HFP, but our method is able to identify the button as a separate entity and also avoid over-segmentation. The WC segmentation has under segmented the model in RedLantern and the segmentation is not meaningful in a-arm2. All of these models are segmented correctly into meaningful parts by our method. The primitive-based approaches like HFP are limited to planes, cylinders and spheres and also requires the user to select the primitives based on observation and select the number of segments. Using HFP for segmentation requires multiple trials for selecting the parameters and coming up with the best-suited values for our expected result.

Our algorithm does not depend on any parametric values as used in many approaches. Many of the current approaches [25,26] used parameters like angle threshold, constraints on geodesic and angular distances, values for the size of the search region etc. Our approach being non-parametric does not need any experience for using the algorithm and obtaining the desired results. Many methods [19] use multiple parameters which need previous experience in tuning them, to obtain the appropriate results. In general, our approach has segmented the models on par with the existing ones or better, without manual intervention or using threshold values.

5.1.2. Noisy input

As a mesh model can be imported from different sources, it is possible that there can be noise in the data. To test the algorithm for noisy inputs, ReMESH 2.1 [48] (an editor for manifold triangle meshes) was used to generate them. Noise is varied by distributing the Gaussian noise over the model in the normal direction and it is performed by increasing the percentage of bounding ball radius (BBR) of the mesh model [48]. Fig. 16 shows the results for models with varying noise specified by BBR. For BBR = 10 to 100, Fig. 16(c)–(f) show that the algorithm has extracted all the segments correctly. For BBR = 200, only one segment was found correct (Fig. 16(g)). For BBR = 500, none of segments were found

Table 1

Running Time of Segmentation. V - number of vertices in the input mesh model, F - number of facets in the input mesh model, RT(s) - running time in seconds, NS - number of segments captured.

Part		14(a)	14(c)	14(d)	14(i)	14(h)	14(w)	14(f)	14(j)	14(t)
Input	V	3505	47593	198	3202	1010	1560	1343	5640	4797
	F	7006	95182	396	6400	2016	3116	2678	11284	9586
RT		0.4	1.7	0.009	0.35	0.05	0.24	0.15	0.7	0.84
NS		50	8	4	4	5	10	4	26	32

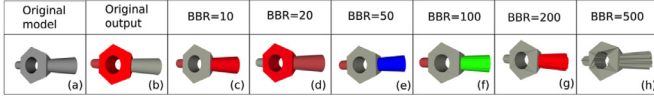


Fig. 16. Performance of our algorithm on Models with varying amount of Noise. (a) Original Model without noise (b) Original model segmented (c)–(f) Models with BBR=10 to 100 are segmented correctly (g) Model with BBR=200 extracts only 2 segments (h) Model with BBR=500 fails to segment.

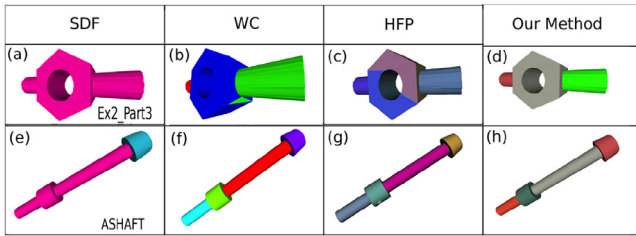


Fig. 17. Comparison of our method with other approaches on Noisy input model (BBR=100).

correctly (Fig. 16(h)). Table 17 compares performance of SDF, WC segmentation, HFP and our method on input models with BBR=100. SDF gives under segmented result for noisy input data. WC segmentation and HFP will not always give the correct segmentation under the influence of noise. The use of strongly connected components enabled the algorithm to handle noisy models, where a manual intervention or parameter setting could be very cumbersome.

5.1.3. Run time for segmentation

Our current implementation of the segmentation algorithm on average takes less than a second for about 5K vertices. Table 1 shows the running time for various CAD mesh models in Fig. 14 along with details of the mesh.

5.2. Part clustering

The part clustering has been implemented using Python 3, Py-Torch [49] and Scikit-learn [50]. All the implementations were carried out on a system running Ubuntu 18.04 Operating System. The system has an Intel Xeon CPU with 32GB RAM and an NVIDIA Tesla K20 GPU with 4GB RAM.

Results of the part clustering algorithm are given in Fig. 18. The outputs generated by algorithm is manually verified with the segmented results based on human perception since there is no labeled benchmark available for part clustering. The results demonstrate the ability of the algorithm to classify similar objects into the same category. Parts of similar classes are represented by the same color. The colors are pre-set randomly.

Fig. 18(a) shows the letters embossed on a surface. All the alphabets are segmented and classified based on shape similarity. We can see that the letters such as n, l, e etc are grouped into appropriate classes. Letters C and G are misclassified into same class due to their geometric similarity and due to less training data. In

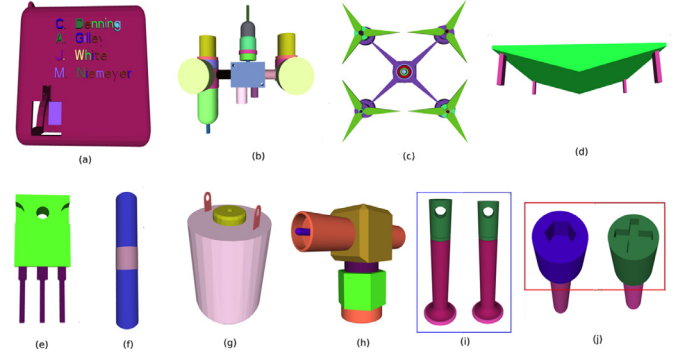


Fig. 18. Results of clustering of segments.

Fig. 18(g) the motorcasing pins are classified into same category. In ballcatcher Fig. 18(d) all the legs in the model are correctly classified. All the pins in the model 1(b) are correctly classified. In Mosfet Fig. 18(e) all the three pins are classified as same set. Similarly in SMATEEAdapter Fig. 18(h), the adapter plugs are correctly classified as same category. Valves in Fig. 18(i) are classified into similar parts across these models.

In Fig. 18(j), although overall the parts look similar, the head part has a slight difference in the geometry of the groove, hence both are classified as different objects.

Gauss-map based feature is able to encapsulate the geometry of the part as well as the aspect ratio of it. Since the feature is normalized later the scaling up or down of models does not influence it. Geometrically similar shapes (eg: Cylinder) having different aspect ratio will generate different features and lead to different clusters.

5.2.1. Run time for part clustering

The training time required for the autoencoder was approximately 30 min. Training is only a single time operation and hence its run time does not affect the part clustering process which may be attempted for different models at various times. The part clustering of a model on average takes 0.01 s.

5.3. PWR of CAD models

The part-in-whole retrieval algorithm is tested using the segmented database which we constructed. The outputs generated by part clustering algorithm is manually verified with the segmented results based on human perception since there is no benchmark available for part retrieval of CAD models. The results of our algorithm are given in Fig. 19. The results demonstrate the ability of the algorithm to retrieve the objects containing the query part as a constituent part. The query part contained across various retrieved results will be colored uniformly. The retrieved results can be manually verified to be of similar geometric characteristics as the query. Since the Gauss-map based feature that we proposed gives weightage to the shape orientation as well as the normalised size of the part to be classified, only shapes containing parts with


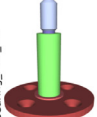
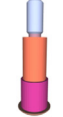
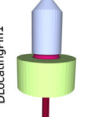
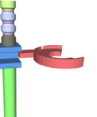

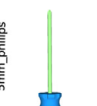

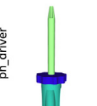


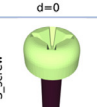

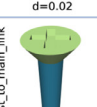


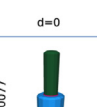
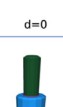
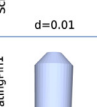

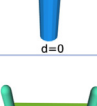

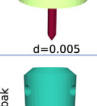


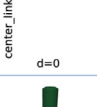
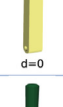
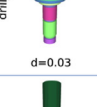
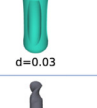
Query	Results				
(a)					
		d=0	d=0.01	d=0.025	d=0.03
(b)					
		d=0	d=0	d=0.02	d=0.02
(c)					
		d=0	d=0	d=0.01	d=0.03
(d)					
		d=0	d=0	d=0.005	
(e)					
		d=0	d=0	d=0.03	d=0.03
(f)					
		d=0	d=0	d=0.005	d=0.01

Fig. 19. Results of part retrieval. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

similar size and shape will be classified into same category and hence retrieved.

Fig. 19 shows each query part and the corresponding retrieved models which are shown in increasing order of their distance measure in the feature space. In Fig. 19(a) the query model is a pin like object, and the retrieval gives all the models (parts in ice blue) containing this part. As the distance increases in the feature space the similarity between the query and the model reduces. We can see that in DLocatingPin1, the part is almost similar but with minor difference towards its base. The final object retrieved Shift_Rod_R_L_prt has two parts in it which are almost similar to the query.

In Fig. 19(b) the query model is a screwdriver head and the result (parts in light green) shows that all the retrieved models contain a similar head part which is cylindrical in geometry. Our database contained few similar models with a rectangular cross section too, but they are clustered differently and hence not retrieved.

Fig. 19(c) retrieved similar models (parts in fluorescent green) with similar groove pattern. The first two objects are exactly similar and the third object is retrieved due to the shape of the groove on top on the segment. The final object retrieved is interesting to look at. Although it looks very different from the query, the raised por-

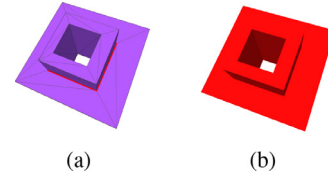


Fig. 20. Limitation of the algorithm: (a) Input model with segment boundary in red. (b) Segmented model.

tions on the cylindrical section does present a geometric similarity to the query model and hence the result.

For the pin query given in Fig. 19(d) the first two models retrieved has exactly similar parts (in sky blue) and the fourth model retrieved is the handle of the screwdriver which is to some extent similar to the cylindrical pin query. The third model retrieved is not exactly a cylindrical pin by functionality, but it got classified into the pin category due to its geometric similarity.

For the part query given in Fig. 19(e), all the models retrieved has exactly similar part contained in the whole model, since all have been correctly classified.

While experimenting with the retrieval, we have observed that whenever a part query is given, the first model retrieved is exactly the whole model containing the query part. The subsequent models retrieved will vary from the query based on the part clustering. Sometimes geometrically similar parts may get grouped together rather than their functionality since our objective is to retrieve geometrically similar parts with the query.

Since ESB is a benchmark which is based on the functionality of each shape and there is no benchmark provided for a part category in each model, it is difficult to evaluate the result automatically. Also to the extent of our knowledge, there exist no other benchmark for a query set with respect to part-in-whole matching. Hence, we rely on validating the results using human perception.

5.3.1. Run time for PWR

The part retrieval of models is quite fast and requires around 0.01 s on average. The training time required for the Autoencoder was approximately 30 min. The training time does not affect the retrieval algorithm.

5.4. Limitations and future work

A limitation of our segmentation method is that when there is a single through hole passing through two parts, the model will not be segmented. For example, in Fig. 20(a) though the segment boundary between the two blocks is found (magenta edges), the region growing process extends the segment through the triangles in the hole region and reaches the other segment. Hence, the two blocks are not segmented and they remain as a single segment as in Fig. 20(b). Triangle subdivision techniques can be included to handle the cases, where there is a single through hole passing through two parts. We are also exploring applications like feature suppression of CAD mesh models.

Also in clustering of the subparts, since the clustering is based on a threshold τ given by the user, setting τ requires some visualization of the dendrogram resulting from the feature space. The result of the clustering will depend a lot of value of τ and it has to be set appropriately. If it is too high, dissimilar clusters may merge together into a single class. if it is too low, then similar parts will be split into multiple clusters.

6. Conclusion

We have developed and demonstrated a threshold-independent non-parametric algorithm for the part-based segmentation of CAD

mesh models without any user input. Experiments on various CAD models showed that the algorithm was not only very fast but also very robust, capturing all the segments. It was shown that Gauss map based feature along with autoencoders and hierarchical clustering can be used for an unsupervised part clustering of CAD models. Finally, part-in-whole retrieval of CAD models demonstrate the utility value of our approach. Limitations and future work have also been discussed.

Acknowledgments

The authors' acknowledge the funding support by Sconce Solutions India Pvt. Ltd. for the segmentation part of this work.

References

- [1] Jayanti S, Kalyanaraman Y, Iyer N, Ramani K. Developing an engineering shape benchmark for CAD models. *Comput Aided Des* 2006;38(9):939–53.
- [2] Regli W. CAD model datasets of national design repository; 2004. <http://edgcsdrexed.edu/repository/>
- [3] Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, et al. ShapeNet: an information-rich 3D model repository Technical Report. Stanford University, Princeton University, Toyota Technological Institute at Chicago; 2015. arXiv: 1512.03012[cs.GR]
- [4] Leizerowicz W, Bilgic T, Lin J, Fox MS. Collaborative design using WWW. In: *Proceedings of WET-ICE*; 1996.
- [5] Iyer N, Jayanti S, Lou K, Kalyanaraman Y, Ramani K. Three-dimensional shape searching: state-of-the-art review and future trends. *Comput Aided Des* 2005;37(5):509–30. doi:10.1016/j.cad.2004.07.002. Geometric Modeling and Processing 2004
- [6] Osada R, Funkhouser T, Chazelle B, Dobkin D. Shape distributions. *ACM Trans Gr* 2002;21(4):807–32. doi:10.1145/571647.571648. <http://doi.acm.org/10.1145/571647.571648>
- [7] Fang Y, Wong E. 3d deep shape descriptor. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*; 2015. p. 2319–28. doi:10.1109/CVPR.2015.7298845.
- [8] Kazhdan M, Funkhouser T, Rusinkiewicz S. Rotation invariant spherical harmonic representation of 3d shape descriptors. In: *Proceedings of the Eurographics/ACM SIGGRAPH symposium on geometry processing, SGP '03*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association; 2003. p. 156–64. ISBN 1-58113-687-0. <http://dl.acm.org/citation.cfm?id=882370.882392>.
- [9] Gunn TG. The mechanization of design and manufacturing. *Sci Am* 1982;247(3):114–31. <http://www.jstor.org/stable/24966684>.
- [10] Ullman DG. *The mechanical design process: Part 1*. McGraw-Hill; 2010.
- [11] Liu Z-B, Bu S-H, Zhou K, Gao S-M, Han J-W, Wu J. A survey on partial retrieval of 3D shapes. *J Comput Sci Technol* 2013;28(5):836–51. doi:10.1007/s11390-013-1382-9.
- [12] Qin F-W, Li L-y, Gao S-M, Yang X-L, Chen X. A deep learning approach to the classification of 3D CAD models. *J Zhejiang Univ SCI C* 2014;15(2):91–106. doi:10.1631/jzus.C1300185.
- [13] Shapira L, Shamir A, Cohen-Or D. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis Comput* 2008;24(4):249. doi:10.1007/s00371-007-0197-5.
- [14] Kaick OV, Fish N, Kleiman Y, Asafi S, Cohen-OR D. Shape segmentation by approximate convexity analysis. *ACM Trans Gr* 2014;34(1):4:1–4:11. doi:10.1145/2611811.
- [15] Shamir A. A survey on mesh segmentation techniques. *Comput Gr Forum* 2008;27:1539–56. doi:10.1111/j.1467-8659.2007.01103.x.
- [16] Attene M, Katz S, Mortara M, Patane G, Spagnuolo M, Tal A. Mesh segmentation - a comparative study. In: *Proceedings of the IEEE International Conference on Shape Modeling and Applications* 2006, SMI '06. Washington, DC, USA: IEEE Computer Society. ISBN 0-7695-2591-1. 7–
- [17] Agathos A, Pratikakis I, Perantonis S, Sapidis N, Azariadis P. 3D mesh segmentation methodologies for CAD applications. *Comput Aided Des Appl* 2007;4(6):827–41.
- [18] Wang X, Zhou B, Fang H, Chen X, Zhao Q, Xu K. Learning to group and label fine-grained shape components. *ACM Trans Gr* 2018;37(6):210:1–210:14. doi:10.1145/3272127.3275009. <http://doi.acm.org/10.1145/3272127.3275009>
- [19] Attene M, Falcidieno B, Spagnuolo M. Hierarchical mesh segmentation based on fitting primitives. *Vis Comput* 2006b;22(3):181–93. doi:10.1007/s00371-006-0375-x.
- [20] Geng C, Suzuki H, Yan D-M, Michikawa T, Sato Y, Hashima M, et al. A thin-plate CAD mesh model splitting approach based on fitting primitives. In: *Colomosse J, Grimstead I, editors. Theory and practice of computer graphics*. The Eurographics Association; 2010. ISBN 978-3-905673-75-3.
- [21] Xiao D, Lin H, Xian C, Gao S. CAD mesh model segmentation by clustering. *Comput Gr* 2011;35(3):685–91. <https://doi.org/10.1016/j.cag.2011.03.020>. Shape Modeling International (SMI) Conference 2011
- [22] Lavoué G, Dupont F, Baskurt A. A new CAD mesh segmentation method, based on curvature tensor analysis. *Comput Aided Des* 2005;37(10):975–87. doi:10.1016/j.cad.2004.09.001.
- [23] Gao S, Zhao W, Lin H, Yang F, Chen X. Feature suppression based CAD mesh model simplification. *Comput Aided Des* 2010;42(12):1178–88. doi:10.1016/j.cad.2010.05.010.
- [24] Sunil V, Pande S. Automatic recognition of features from freeform surface CAD models. *Comput Aided Des* 2008;40(4):502–17. <https://doi.org/10.1016/j.cad.2008.01.006>.
- [25] Zhang Y, Paik J, Koschan A, Abidi MA, Gorsich D. 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*, 3; 2002. III–273–III–276 vol.3
- [26] Koschan AF. Perception-based 3D triangle mesh segmentation using fast marching watersheds. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2; 2003. II–27–II–32 vol.2
- [27] Wu K, Levine MD. 3D part segmentation using simulated electrical charge distributions. *IEEE Trans Pattern Anal Mach Intell* 1997;19(11):1223–35. doi:10.1109/34.632982.
- [28] Lee Y, Lee S, Shamir A, Cohen-Or D, Seidel H-P. Mesh scissoring with minima rule and part salience. *Comput Aided Geom Des* 2005;22(5):444–65. doi:10.1016/j.cagd.2005.04.002. Geometry Processing
- [29] Benhabiles H, Lavou G, Vandeborre J-P, Daoudi M. Learning boundary edges for 3D-mesh segmentation. *Comput Gr Forum* 2011;30(8):2170–82. doi:10.1111/j.1467-8659.2011.01967.x.
- [30] Liu R, Zhang H. Segmentation of 3D meshes through spectral clustering. In: *Proceedings of the twelfth pacific conference on computer graphics and applications*; 2004. p. 298–305. doi:10.1109/PCCGA.2004.1348360.
- [31] Wu MC, Jen SR. A neural network approach to the classification of 3D prismatic parts. *Int J Adv Manuf Technol* 1996;11(5):325–35. doi:10.1007/BF01845691.
- [32] Ip CY, Regli WC, Sieger L, Shokoufandeh A. Automated learning of model classifications. In: *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications, SM '03*. New York, NY, USA: ACM; 2003. p. 322–7. ISBN 1-58113-706-0. doi:10.1145/781606.781659.
- [33] Ip CY, Regli WC. Content-based classification of CAD models with supervised learning. *Comput Aided Des Appl* 2005a;2(5):609–17. doi:10.1080/16864360.2005.10738325.
- [34] Ip CY, Regli WC. Manufacturing classification of CAD models using curvature and SVMs. In: *Proceedings of the international conference on shape modeling and applications, (SMI' 05)*; 2005b. p. 361–5.
- [35] Hou S, Lou K, Ramani K. SVM-based semantic clustering and retrieval of a 3D model database. *Comput Aided Des Appl* 2004;2:155–64.
- [36] Ferreira A, Marini S, Attene M, Fonseca MJ, Spagnuolo M, Jorge JA, et al. Thesaurus-based 3D object retrieval with part-in-whole matching. *Int J Comput Vis* 2010;89(2–3):327–47. doi:10.1007/s11263-009-0257-6.
- [37] Bai J, Gao S, Tang W, Liu Y, Guo S. Design reuse oriented partial retrieval of CAD models. *Comput Aided Des* 2010;42(12):1069–84. doi:10.1016/j.cad.2010.07.002.
- [38] Li Z, Zhou X, Liu W. A geometric reasoning approach to hierarchical representation for B-rep model retrieval. *Comput Aided Des* 2015;62:190–202. doi:10.1016/j.cad.2014.05.008.
- [39] Tao S, Huang Z, Ma L, Guo S, Wang S, Xie Y. Partial retrieval of CAD models based on local surface region decomposition. *Comput Aided Des* 2013;45(11):1239–52. doi:10.1016/j.cad.2013.05.008.
- [40] Tao S, Wang S, Chen A. 3D CAD solid model retrieval based on region segmentation. *Multimed Tools Appl* 2017;76(1):103–21. doi:10.1007/s11042-015-3033-3.
- [41] Hoffman D, Richards W. *Parts of recognition*. *Cognition* 1983;18:65–96.
- [42] The CGAL Project. CGAL user and reference manual. 49 ed. CGAL Editorial Board; 2016. <http://doc.cgal.org/4.9/Manual/packages.html>
- [43] Berg MD, Cheong O, Kreveld MV, Overmars M. *Computational geometry: algorithms and applications*. 3rd ed. Santa Clara, CA, USA: Springer-Verlag TELOS; 2008. ISBN 3540779736, 9783540779735
- [44] Rumelhart DE, Hinton GE, Williams RJ. Parallel distributed processing: explorations in the microstructure of cognition. In: *Learning Internal Representations by Error Propagation*, vol. 1. Cambridge, MA, USA: MIT Press; 1986. p. 318–62. ISBN 0-262-68053-X. <http://dl.acm.org/citation.cfm?id=104279.104293>
- [45] do Carmo MP. *Differential geometry of curves and surfaces*. Prentice Hall; 1976. ISBN 978-0-13-212589-5.
- [46] Kingma DP, Ba J. Adam: a method for stochastic optimization. *CoRR* 2014;abs/1412.6980. arXiv: 1412.6980
- [47] Meybaum H, Narusk I. GrabCAD; 2017. <https://grabcad.com/>
- [48] Attene M, Falcidieno B. ReMESH: an interactive environment to edit and repair triangle meshes. In: *Proceedings of the SMI*. IEEE Computer Society; 2006. p. 41. ISBN 0-7695-2591-1.
- [49] Paszke A, Gross S, Chintala S, Chanan B, Yang E, DeVito Z, et al. Automatic differentiation in pytorch. In: *Proceedings of the NIPS-W*; 2017.
- [50] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. *J Mach Learn Res* 2011;12:2825–30.