Computer Aided Geometric Design 52–53 (2017) 231–246

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

A primitive-based 3D segmentation algorithm for mechanical CAD models

Truc Le, Ye Duan *

Department of Computer Science, University of Missouri, Columbia, USA

ARTICLE INFO

Article history:
Available online 24 February 2017

Keywords:
Primitive-based segmentation
Profile curve
RANSAC
Set-cover problem
Orientation

ABSTRACT

This paper presents a novel segmentation algorithm for mechanical CAD models (represented by either mesh or point cloud) constructed from planes, cylinders, cones, spheres, tori and easily extendable to surfaces of revolution. Our proposed approach differs from existing techniques in the following aspects. First, by assuming that common mechanical models only have a limited number of dominant orientations that their primitives are either parallel or orthogonal to, we narrow down the search space for detecting the primitives to the automatically estimated major orientations of the input model. Second, we employ a dimension reduction method which transforms the problem of detecting 3D primitives into the classical 2D problems such as circle and line detection in images. Third, we generate an over-complete set of primitives and formulate the segmentation as a set cover optimization problem. We demonstrate our method's robustness to noise and show that it compares favorably with state-of-the-art solutions such as the RANSAC-based (Schnabel et al., 2007) and GlobFit (Li et al., 2011) approaches on many synthetic and real scanned examples.

1. Introduction

Computer-aided design (CAD) models are geometric models which are a conceptually higher level and more accurate representation of an object than mesh or point cloud. In normal cases, engineers and designers create CAD models for printing, machining or other manufacturing operations in many areas including automotive, shipbuilding, aerospace industries, industrial and architectural design, prosthetics, and many more. However, during the process, the original CAD models may be lost so people would like to reconstruct them from meshes or point clouds being scanned from the real objects. The reconstruction process refers as reverse engineering and it has ample applications (Beniere et al., 2011, 2013; Tenorth et al., 2013). The reverse engineering saves engineers and designers time in a way that they do not need to create a whole CAD model from scratch but instead get it reconstructed from a real model followed by modification.

Segmentation of a mesh or point cloud is at the heart of reverse engineering and it is also a fundamental problem of computer graphics which has been extensively studied over the past several years. In general, the segmentation problem is ill-posed and no objective measurement does exist for universally assessing segmentation quality. Judging the quality of a segmentation is application dependent (Shamir, 2008; Chen et al., 2009) and it is very difficult to propose an approach capable of segmenting all various kinds of models.

* Corresponding author.
E-mail addresses: tdlxqb@mail.missouri.edu (T. Le), duanye@missouri.edu (Y. Duan).

http://dx.doi.org/10.1016/j.cagd.2017.02.009
0167-8396/

Primitive type recognition and primitive fitting are key issues for mechanical CAD model segmentation (Varady et al., 1997). Schnabel et al. (2007) developed an efficient RANSAC-based (Fischler and Bolles, 1981) framework for recognizing planes, spheres, cylinders, cones and tori. This approach is quite fast and robust to outliers. However since the RANSAC-based approach only looks for local cues it may result in over or under-segmentation or even wrongly identified primitive types because its estimation of the primitive parameters is sensitive to noise of both the sampled points' positions and normals (Isack and Boykov, 2012) (see examples in Figs. 10 and 11 for its failure cases). This issue also applies to other methods that use low level information such as Gaussian or mean curvature (Zhang et al., 2002; Zuckerberger et al., 2002).

Cohen-Steiner et al. (2004) proposed the variational shape approximation method that employs linear approximation, or planes, to segment the model. Later, Wu and Kobbelt (2005) extended this algorithm to cylinders, spheres and rolling ball surfaces. This variant also tends to over-segment the regions between patches and is sensitive to noise as it usually creates many new proxies for the noisy regions. Yan et al. (2006, 2012) used iterative quadratic surface fitting for segmenting the meshes of CAD models as well as free-form geometry, and the experimental results seem to be very promising. This method uses quadratic surfaces to approximate all kinds of primitives (plane, sphere, cylinder, cone, etc.). The results obtained from this approach can be adjusted easily as the user can insert and merge quadratic surfaces interactively. However for CAD objects, quadratic surfaces are sometimes too flexible because they may over-fit the underlying surface and create an undesired segmentation. In addition, the segmentation quality is heavily dependent on both the number and the positions of the initial seeds.

Recently, researchers have been trying to unearth global information from the 3D models. Li et al. (2011) designed the GlobFit method which improves RANSAC's output by exploring the global relationship among primitives such as parallelism, orthogonality, co-axis, equality, etc. The authors pose the discovery of a global relationship as a constrained optimization problem where the objective function is the least-squares fitting term and the constraints are the relations among primitives. This method, as its name suggests, focuses on fitting rather than segmentation. It is able to refine primitives' parameters such as location and orientation to globally align them. However since the global relationship is extracted during a post-processing stage, it is highly dependent on and limited by the initial primitives output by RANSAC. If the initial primitive types provided by RANSAC are wrong, it is very difficult if not impossible for the algorithm to correct them (see Fig. 1). Monszpart et al. (2015) proposed the RAPter approach to extract a regular arrangement of planes from point cloud (of man-made scene). Their main technical contribution is a formulation that balances the dominant scene orientations and the less-dominant orientations as the internal integrity. Demir et al. (2015) segment and detect similarities within an existing 3D architectural model by casting the segmentation problem as a weighted minimum set cover over an input triangle soup, and maximize the repetition of similar segments to find a best set of unique component types and instances. Golovinskiy et al. (2009) attempted to apply machine learning concepts to recognize semantic objects in 3D point clouds of urban environments. However, it is difficult to propose an appropriate shape feature for all kinds of objects, especially for the geometric primitives in the context of CAD models. Moreover, the requirement for moderate to large training data in 3D cannot always be met.

In this paper, we propose a novel segmentation framework for mechanical CAD models that overcomes some of the limitations of the existing work. Our method can deal with planes, cylinders, cones, spheres, tori and can be easily extended to surfaces of revolution. Based on the observation that common mechanical models only have a limited number of dominant axes around which the primitives are constructed (i.e. the primitives are either parallel or orthogonal to one of these axes), we estimate the dominant directions of the model by detecting *poles* and *rings* formed by the point cloud's normals. Note that the idea of using the normals to estimate the major directions has been studied before (Reisner-Kollmann et al., 2011; Qiu et al., 2014), but people have only explored incorporating major orientations for planes in building scans (Reisner-Kollmann et al., 2011) or cylinders in pipelines (Qiu et al., 2014). We estimate them for all planes, cylinders, cones, tori and use them to generate an over-complete set of various segmentations of the CAD models. The estimated major directions significantly reduce the search space and degrees of freedom for the subsequent primitive detection and enhance the robustness as well as the accuracy of the segmentation. We then convert the 3D primitive detection into a sequence of 2D circle and line detection by slicing the point cloud along each main direction, finding circles in each slice, constructing and segmenting the profile curve. The final segmentation is formulated as a set cover optimization where "items" are over-segmented planar patches and "subsets" are the over-complete set of detected primitives obtained from the profile curve analysis.

The proposed dimension reduction approach makes our method more robust to noise. For example, suppose we need to find a cylinder among 100 3D-points with 20% outliers. To estimate a cylinder with 5 degrees of freedom, we need at least 5 points, and hence, the solution space is $\binom{100}{5} = 75{,}287{,}520$. Because there are only 80 points that actually belong to the cylinder, the number of correct tuples is $\binom{80}{5} = 24{,}040{,}016$. It means that no matter what detection algorithm we use, it has to distinguish 24,040,016 tuples from the total of 75,287,520 tuples, which leads to the successful detection probability $\frac{24{,}040{,}016}{75{,}287{,}520} \approx 31.93\%$. Now consider an equivalent 2D circle detection with 100 2D-points with the same 20% outliers. At least 3 points are required to calculate a 2D circle. Following similar computation, we obtain the successful detection rate $\frac{82{,}160}{161{,}700} \approx 50.81\%$, which is significantly higher than that of the cylinder. Moreover, slicing the model, detecting circles in each slice and analyzing the profile curve altogether make our approach robust. In fact, a stack of many co-centered circles (of the same radius) provides more confidence of the presence of a cylinder in 3D than a fragile direct detection of a cylinder in 3D. Our main contributions include:

Fig. 1. Failure case of RANSAC-based approach (Schnabel et al., 2007) and GlobFit (Li et al., 2011). **First column**: Input model (point cloud); **Second column**: Segmentation from RANSAC-based approach is locally optimal (black lines indicate the axes of the primitives); **Third column**: GlobFit's optimization can align the parallel primitives but increase the fitting error and fail to recover the correct segmentation; **Last column**: Our segmentation. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

- A global model major orientation estimation of the mechanical model which is done early in the pipeline instead of in a post-processing step.
- A dimension reduction approach that transforms the 3D primitive detection problem into the classical 2D problems such as circle and line detection in image.
- A formulation of the segmentation as a set cover problem.

Our proposed algorithm is more robust than the RANSAC-based approach (Schnabel et al., 2007) due to the fact that we globally identify the model's major orientations, incorporate a dimension reduction approach, produce an over-complete set of geometric primitives and find the optimal segmentation as a set cover optimization. Comparing with the GlobFit (Li et al., 2011), the segmentation result of our algorithm is not dependent on the quality of the initial segmentation result and is much faster. The remainder of this paper is organized as follows. Section 2 briefly reviews other related works on 3D segmentation. Section 3 describes our novel segmentation method in detail. Experimental results are shown in Section 4 along with a side-by-side comparison with some of the state-of-the-art methods including RANSAC (Schnabel et al., 2007) and GlobFit (Li et al., 2011). Section 5 concludes our work. In this paper, we use random colors to differentiate parts of a segmentation result. To visualize the primitive types, we use red, green, blue, cyan, olive and purple for planes, cylinders, cones, spheres, tori and surfaces of revolution, respectively. The unlabeled regions are in golden color.

2. Other related works

The CAD models are often represented with either point clouds or meshes. The class of segmentation algorithms applied for point clouds and that applied for meshes actually overlap. The point cloud-based approaches usually can handle mesh as well by sampling points from a mesh, but generally not the other way around. Shamir (2008), Agathos et al. (2007) and Theologou et al. (2015) give a detailed survey on mesh segmentation techniques. Many segmentation approaches are usually formulated as an energy minimization problem and various error measures have been proposed in the literature to define the energy. These measurements quantify the properties such as planarity of various forms, higher degree geometric proxies (cylinders, cones, spheres, etc.), dihedral angles between triangles (Xiao et al., 2011), curvatures (Gaussian curvature or mean curvature) (Lavoue et al., 2005), geodesic distances on a mesh, slippage, symmetry, convexity, medial axis, shape diameter (Shapira et al., 2008) and motion characteristics (Shamir, 2008). According to Shamir (2008), a majority of mesh segmentation algorithms can be divided into five classes: region growing, hierarchical clustering, iterative clustering, spectral analysis and implicit methods, some of which can also be applied for point cloud. These algorithms make use of the error metrics described above to group "similar" triangles into segments.

Region growing, the simplest of all possible segmentation methods, is locally greedy and very fast (Vieira and Shimada, 2005; Jagannathan and Miller, 2007). Region growing includes single-source (where the growing starts from a single seed and stops before growing from another seed) and multiple-source region growing. Variants of the latter include the watershed method (Mangan and Whitaker, 1999) and the distortion-minimizing flooding algorithm (Cohen-Steiner et al., 2004) which cleverly controls the growing based on a priority queue. Researchers apply hierarchical clustering in both a bottom-up and top-down fashion. In the bottom-up hierarchical clustering (Attene et al., 2006; Garland et al., 2001), the algorithm starts with every triangle as a separate segment. At each iteration, the best fitting geometric primitive (plane, cylinder, sphere, cone, torus) approximating the triangles among every pair of adjacent segments determines the merging process, until hitting a required number of segments. In the top-down approach, the partition is achieved by finding the best boundary between parts at each step (Katz and Tal, 2003). Iterative clustering methods such as the K-means method and mean shift (Yamauchi et al., 2005) usually cannot be applied directly for recognition of mechanical parts. Spectral graph theory-based segmentation algorithms (Shi and Malik, 2000) demonstrate some success in image segmentation but cannot be directly applied to CAD segmentation because the construction and analysis of the affinity matrix on millions of triangles is infeasible. The spectral approach also does not exhibit primitive information, which is important for mechanical parts. It has recently been proven that noisy mesh normals significantly affect shape recognition results (Yi et al., 2014). Yi et al. (2014) developed an iterative slippage analysis which is less affected by normals' noise. This method works well for small to medium-sized models but does not scale well for large-sized models. Implicit methods extract the contours and hence implicitly define the segmentation (Mitani and Suzuki, 2004; Lee et al., 2005). The random walk method (Lai et al., 2008) usually cannot be directly applied to segment CAD models, because it depends on the initial seeds and, more importantly, the probability computation is based on local cues such as Gaussian and mean curvatures which do not explicitly enforce the model's primitive structure. The heat walk algorithm (Benjamin et al., 2011) was designed for segmenting free-form objects and hence it yields bad segmentation results (over-segmentation, undesired boundary creation, non-primitive conforming segments) on CAD models.

The most relevant works to ours are the pipe-run extraction and reconstruction (Qiu et al., 2014), the generalized cylinder decomposition (Zhou et al., 2015) and the polycube map construction by (He et al., 2009). Qiu et al. (2014) capture the dominant cylindric shapes and reveal similarities between cylinders. They also use the circle detection to guide the primitive fitting, but their method is only applied for cylinders with T-joints, boundary joints or curved joints. Zhou et al. (2015) locally fit generalized cylinders (GCs) by computing the cylindrical term (a linear combination between the straightness of the local skeleton and the profile variation), then merge local GCs into non-local GCs and finalize the decomposition by solving the Exact Cover Problem. GCs, however, are too abstract to reveal the accurate structure of the object, which is

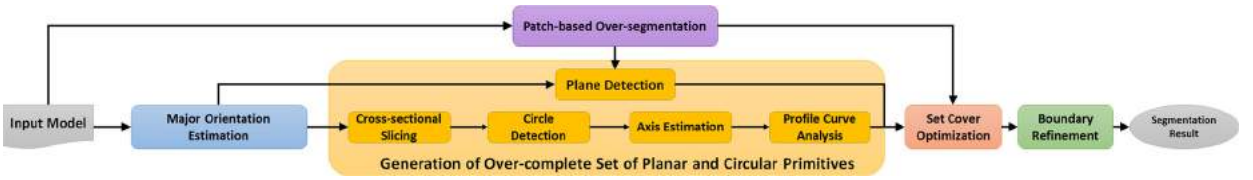

Fig. 2. Flowchart of our segmentation approach.

Algorithm 1 Our algorithm for segmenting a point cloud P.

1: Over-segmentation of the model into patches (Section 3.1).
2: Estimate the model's major orientations (Section 3.2).
3: Detect planes orthogonal to each of the major orientations (Section 3.3).
4: Generate an over-complete set of circular primitives (Section 3.4).
   – Slice the model along each of the major directions, generate slicing image and detect 2D circles.
   – Group adjacent circles with common centers into connected components. Estimate the axis location and orientation of each connected component by fitting line to the circle centers in the connected component.
   – Construct profile curve for each component and segment the profile curve into lines and circles.
5: Set cover optimization (Section 3.5).
6: (Optional) Refine segmentation boundary (Section 3.6).

essential in the CAD context. He et al. (2009) slice the model by horizontal planes which serves as a divide and conquer approach for polycube construction. However, these two later works are not designed for decomposing CAD models into geometric primitives. To the best of our knowledge, we are the first ones using the model's major directions to decompose the model and doing the dimension reduction to detect various types of geometric primitives.

3. Our approach

Fig. 2 and Algorithm 1 summarize the main pipeline of our segmentation framework with illustration on a model named *mechanical part* in Fig. 3. It consists of the following main stages: patch-based over-segmentation, model's major direction estimation, generation of an over-complete set of planar and circular primitives, set cover optimization and boundary refinement. In general, our approach first over-segments the model by planar patches. Then using the detected major orientations, we produce an over-complete set of (planar and circular) primitives and optimize them as a set cover problem. In our framework, two predefined thresholds, the distance (ϵ) and normal (α) enforce the tolerance between a primitive and the sampled point. We set ϵ to 0.5 percent of the length of the diagonal of the point cloud's bounding box and α to 20° for all of our tested models. In the subsequent sections, we will elaborate on each step.

3.1. Patch-based over-segmentation

This step takes as input the point cloud and produces a set of planar patches, which is essential for the optimization (in Section 3.5) as well as plane detection. A plane only has three degrees of freedom and is easier to be robustly detected than cylinder, cone, sphere and torus. We deploy the variational shape approximation (Cohen-Steiner et al., 2004) to linearly approximate the input model. A simple region growing is used to bootstrap the algorithm. We randomly pick a point s as a seed point and merge the neighboring points whose normals do not deviate more than α from the normal of s. The process is repeated until no more patches can be generated. Fig. 3B shows an example of patch generation from the *mechanical-part* model.

Nevertheless, it is possible that a patch crosses the boundary between two different primitives (Fig. 4 left). To solve this problem, we compute the crest lines (Yoshizawa et al., 2005) (a subset of curvature extrema) of the input model and use them as blocking markers to prevent the patches from growing across the primitives. As we can see in Fig. 4, the crest lines (middle) detected in the *carter* model improve the over-segmentation result (right).

3.2. Major direction estimation

A common property of a mechanical model is that it has a limited number of (usually from 1 to 3) main orientations around which the geometric primitives (planes, cylinders, cones, tori) reside. If these major orientations are known, it is possible to narrow down the search space for detecting the primitives because their degrees of freedom are reduced by two. In this paper, we are dealing with planes, cylinders, cones, spheres and tori (please note that spheres do not have any natural axis and we will talk about it later in Section 3.4). We refer the primitive's axis to the plane's normal and the cylinder, cone or torus's axis and it can be found from the distribution of the point cloud's normals. Ideally, the distribution of the plane's normals on a spherical Gaussian surface is a single point and that of a cylinder or a cone is a circle. In other words, if we can locate such special points and circles from the normals' distribution mapped on the Gaussian sphere, the model's major orientations can be derived in a straightforward manner.

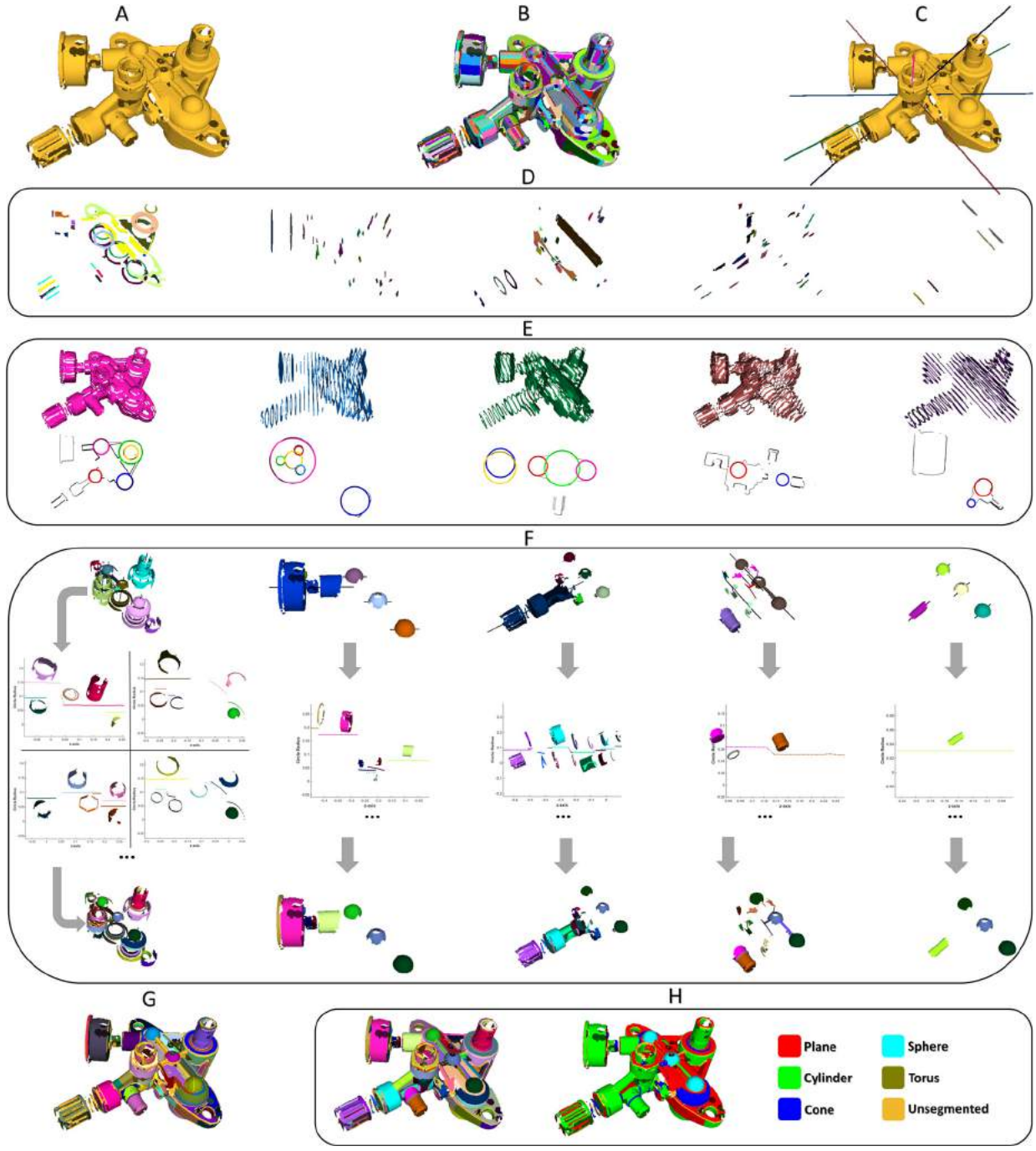

Fig. 3. Illustration of our approach on the *mechanical-part* model. **A**: Input point cloud; **B**: Patch-based over-segmentation; **C**: Major orientation estimation; **D**: Planes are detected from over-segmented patches from **B** and the major orientations from **C**; **E**: Cross-sectional slicing of the model along each orientation and circle detection on each slice; **F**: *top*: Group adjacent circles into connected components and for each component, fit line to the circle centers to estimate the circular primitive's axis location (and possibly refine the axis orientation); *bottom*: circular primitives are generated from the profile curve analysis (best viewed with electronic zoom-in or see Fig. 8 for more details); **G**: Set cover optimization; **H**: Final segmentation result after boundary refinement (*left*: segmentation colored by primitives; *right*: segmentation colored by primitive types). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

However, the real scanned models inevitably have noise. As a result, the distribution of the plane's normals becomes a high density circular region, or *pole*, and the circle obtained from the cylinder or cone's normals becomes a *ring* (Fig. 6B), which makes the model's major orientations harder to be revealed. To tackle the problem, we find the poles and rings separately. The poles are easier to be detected as the mean shift clustering (Cheng, 1995) can be applied, but we have made

Fig. 4. Crest lines prevent the patches from spanning multiple primitives. **Left**: patches (without crest lines); **Middle**: detected crest lines shown in black (Yoshizawa et al., 2005); **Right**: patches from *constrained* region growing. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

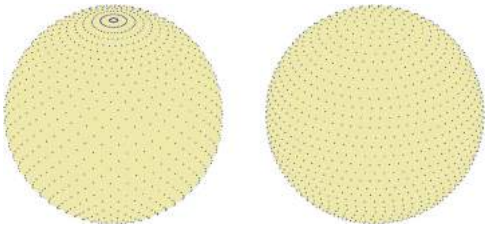

Fig. 5. Partition of a sphere into 1,600 regions by sub-dividing the spherical coordinates (**left**) and the equal-area partitioning (Leopardi, 2006) (**right**).

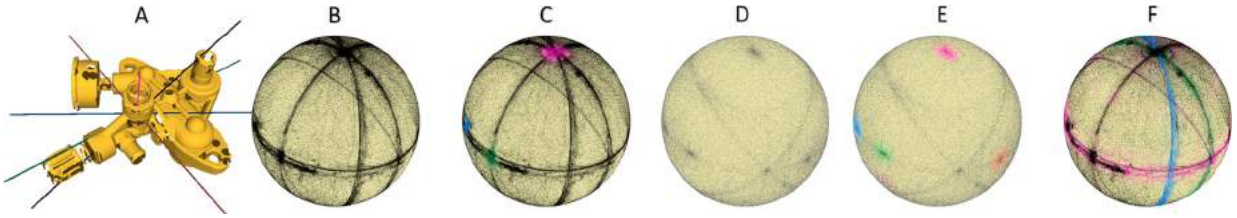

Fig. 6. Major direction estimation. **A**: Input model and the estimated major directions; **B**: Projection of point cloud's normals (black dots) onto the Gaussian sphere; **C**: the poles (in pink, dodger blue and dark green) of the normals' distribution; **D**: mapping of randomly chosen triples (a triple of three black dots in **B** corresponds to a gray dot); **E**: poles (in pink, dodger blue, dark green, brown and purple) in the mapped space; **F**: corresponding rings (in pink, dodger blue, dark green, brown and purple) of the normals' distribution. The final model's major directions are shown in **A**. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

two modifications. First, to initialize the starting position, we use the histogram method. The Gaussian sphere is divided into 1,600 regions, each of which represents a bin to accumulate the normals. There are many ways to achieve such division. The most naive method is using the spherical coordinates and sub-dividing the azimuth angle and the elevation (e.g. 40×40). Unfortunately, this leads to non-equal zones and distortions near the pole (see Fig. 5 left). As a result, some bins are bigger and have higher probability of receiving more votes while the bins near the poles are too fine to expose any local maxima. To overcome such problem, we use the equal-zone sphere partition (Leopardi, 2006) which ensures that each bin has the same surface area, hence have equal probability of receiving votes (Fig. 5 right). Second, we check the stability of the final modes. Upon convergence, each mode is randomly perturbed (it is shifted by some random noise) a few times (e.g. 20) and the mean shift is run again at the new starting location. If the mode converges to the same position, it is called stable and is accepted as a pole. Fig. 6C shows the poles detected from the normals' distribution from Fig. 6B.

The rings, on the other hand, are more challenging because a small deviation of the ring plane results in a relatively big change in its orientation. Yet we only need to know the rings' orientations, not to exactly locate every ring. Consequently, we choose three normals (black points in Fig. 6B) at random and compute the orientation of the plane passing through them. The process is repeated over a relatively large number of triples (e.g. a million triples). That is to say we have transformed the problem of finding the circular rings' orientations in the original Gaussian sphere into that of finding the poles in the new Gaussian sphere (see Fig. 6D). Hence the same procedure of recognizing poles in the preceding paragraph is deployed in the converted space (Fig. 6E and 6F). Moreover, another layer of verification is added. We project the point cloud onto a plane perpendicular to each orientation candidate and assert if there exists a circle in the projection image.

The list of poles and rings' orientations are unified together to eventually become the model's major directions (Fig. 3C). These orientations serve as a decomposition of the input model which reduces the search space for the planar and circular primitive detection described in the subsequent sections.

It is worth mentioning that we use conservative thresholds for the mean-shift clustering during the pole detection (both in the original and mapped space) to make sure that we do not miss any orientation. Our algorithm, however, can still work smoothly given redundant orientations as it would only take more time for detecting the circular primitives for each orientation. The major orientation detection for all tested models is shown in Fig. 7.

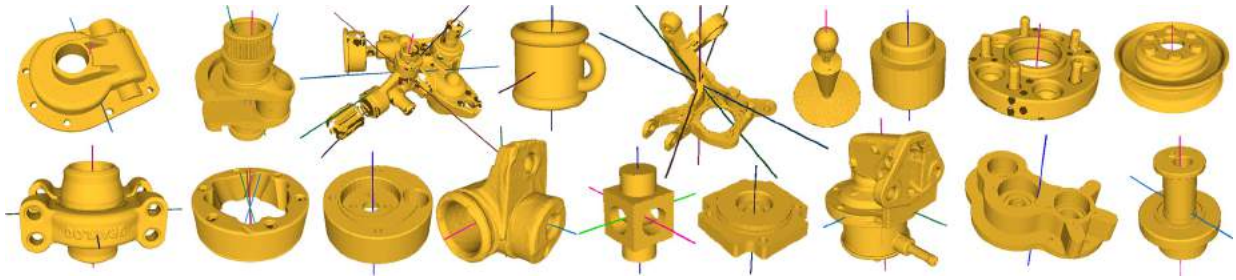

Fig. 7. The estimated major orientations for all tested models. Note that the colored axes represent orientations only, not location. See Fig. 1 (last column) for our detected axes. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

3.3. Plane detection

The planar patches obtained from the over-segmentation (Section 3.1) are planes of themselves. However, to increase the robustness of the plane detection, we further merge it under the constraint of the model's major orientation. More specifically, for each model's major orientation, we group the patches orthogonal to it. Then the z-value of each patch (with respect to the orientation and the origin) is calculated, which is clustered to get the planes orthogonal to each of the model's orientations. Fig. 3D shows examples of plane detection for five different orientations of the *mechanical part* model.

3.4. Generation of over-complete set of circular primitives

This step is to propose various hypothetical segmentations by generating an over-complete set of circular primitives using the detected major orientations. For the circular primitives such as cylinder, cone and torus, the intersection between each of them and a plane orthogonal to its axis is a circle. A plot of the circle's radius versus the displacement of the plane containing the circle is called the *profile curve*. The profile curve of a cylinder is a horizontal line segment and that of a cone is a oblique line segment. The profile curve of a torus is a circle whose center is off the z-axis. A sphere does not have any natural axis, but if given any axis passing through its center, its profile curve can be defined similarly to the case of cylinder, cone and torus. The profile curve of a sphere is also a circular arc but its center is on the z-axis. Exploiting this property, we convert the 3D circular primitive detection in the original model into the 2D circle detection in its cross-sectional projection along the primitive's directions followed by segmenting the profile curve into line segments and circular arcs.

3.4.1. Projection image generation and 2D circle detection

Starting with the model's major orientations, we uniformly slice the model along each of them (see Fig. 3E top row). The points whose normals are parallel to the axis orientation have been grouped into planes (Section 3.3) and thus are excluded here. The thickness of each slice, denoted by τ, depends on the point cloud's density. In all of our experiments, we choose $\tau = \epsilon = 0.5\%\, diagonal$, where *diagonal* is the length of the diagonal of the model's bounding box. There are at most $\frac{diagonal}{\tau} = \frac{diagonal}{0.5\%\, diagonal} = 200$ slices from each of the major orientations. Each slice defines a plane orthogonal to a major orientation and containing the projection of all points within τ-distance. Examples of projection images are shown in Fig. 3E second row in black pixels.

After the projection images are generated, we will proceed to detect the circles on these images. The 2D circle detection is a classical problem and it has been extensively studied for many decades (Illingworth and Kittler, 1988; Chen and Chung, 2001; Rad et al., 2003; Kim and Kim, 2001; Ho and Chen, 1995; Yuan and Liu, 2015; Cuevas et al., 2012; Akinlar and Topal, 2013). In general, one could use one of these methods as a sub-routine for this task. We apply our circle detection (Le and Duan, 2016) because it is much more robust than both the Circular Hough Transform and RANSAC approaches.

3.4.2. Primitive axis estimation

Section 3.2 only estimates the axis directions with no information about the axis location. After all the circles are detected in the preceding section, we will group the adjacent circles with nearby circle centers into connected components. For each connected component of 2D circles we conduct line fitting to all the circle centers in the connected component, which not only gives the axis location but also refines the axis orientation.

3.4.3. Profile curve analysis

After the axis is estimated, a profile curve is constructed for each connected component. The vertical axis represents the radii of the circles in the connected component and the horizontal axis shows the z-values of the circles' centers. Figs. 3F and 8 show examples of the *profile curves* obtained from groups of co-centered circles. Cylinders and cones will be represented by line segments while spheres and torus by circular arcs in the profile curves, respectively. Thus, detecting line segments and circular arcs in the profile curves is equivalent to detecting circular primitives in the 3D space.

A circular arc could also be approximated by multiple short line segments. To reduce the ambiguity between a circular arc and line segments, we propose to first extract the circular arcs from the profile curve. We use the same method

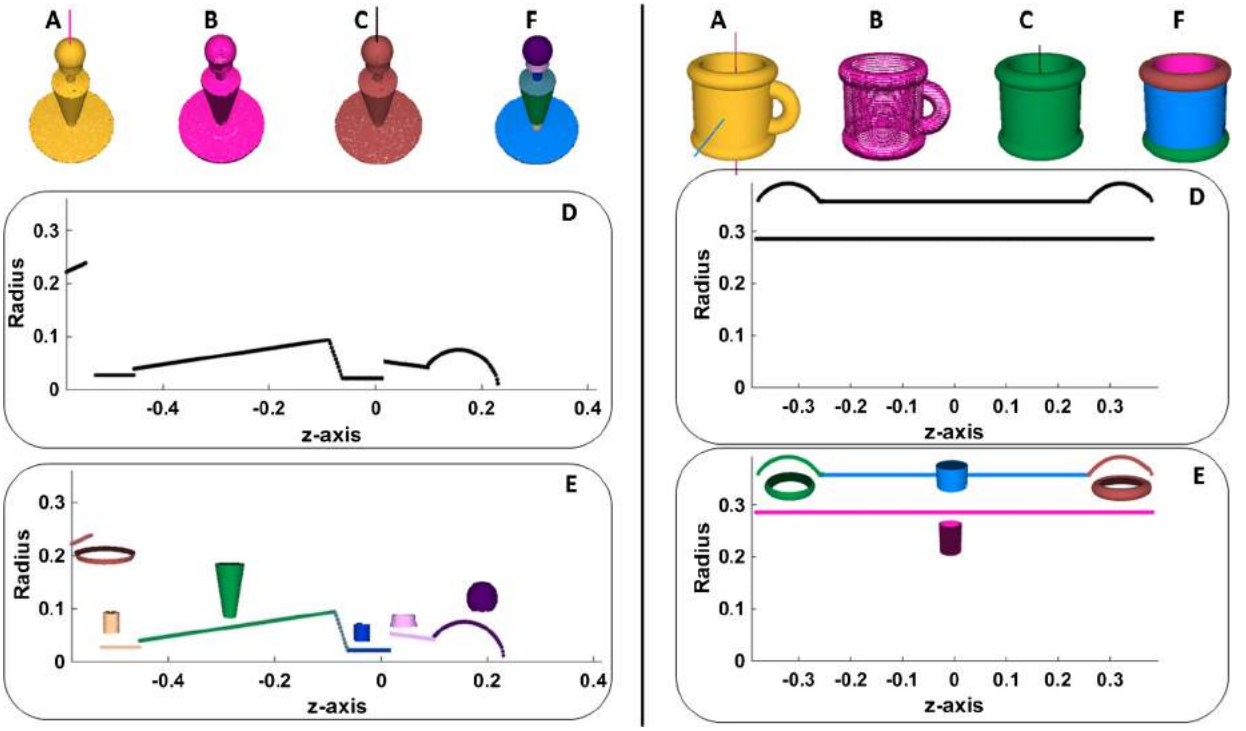

Fig. 8. Two examples of profile curve analysis. **A**: Input models and detected orientations; **B**: Model's slicing along the orientation; **C**: Group of adjacent detected circles; **D**: Profile curve from **C**; **E**: Segmentation of profile curve into line segments and circular arc; **F**: Detected primitives in 3D. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

(Le and Duan, 2016) to detect circular arcs. The circular arcs centered on z-axis correspond to spheres and those centered off z-axis correspond to torus (see Fig. 8).

Once the circular arcs are extracted, the remaining task is to detect all line segments in the remaining profile curve. In this paper, we employ a region growing algorithm for line segment detection. The region growing is performed based on the circle radius (vertical axis of the profile curve), the circle center's displacement (horizontal axis of the profile curve) and the circle's average angle (measured by the average over the angles between the 3D normals of the associated points and the major orientation) which serves as the *gradient* of the curve (in fact, this angle equals 90° minus the gradient of the curve). Other techniques such as the LSD (von Gioi et al., 2010) could also be used in place of the region growing. However, our experiments show that LSD sometimes could miss smaller line segments which our region growing algorithm correctly detects. Sample outputs of the line segments obtained from the profile curves are shown in Figs. 3F, 8E with their associated primitives.

Based on the profile curve information, the circular primitive's parameters can be easily computed. All circular primitives are extracted from each of the model's major orientations and individually fitted by the method in (Lukacs et al., 1998) which is known to be better than traditional least-squares fitting due to its robustness to noise and degeneration. An additional verification step in 3D based on distance and normal errors of the 3D points and the primitives is added for eliminating the bad quality (or spurious) primitives.

3.5. Set cover optimization

So far we have generated a linear approximation (planar patches) P of the model and all of the hypothetical circular primitives. The set of hypothetical primitives, H, includes both the planar patches and the circular primitives. Note that the hypothetical primitives may overlap on each other. A patch $p \in P$ is said to belong to a hypothetical primitive H_i if the majority (e.g. 80%) of its points belong to H_i. We use both distance and normal constraints (with ϵ and α tolerances) to verify the point-primitive's membership.

It can be seen that a segmentation is an assignment of each patch in P to each hypothetical primitive in H, or, in other words, a choice among all subsets of H covering P. Based on the principle of the *Minimum Description Length*, we define a good segmentation as a minimal subset of H that fully covers P. Let us further denote a set of binary indicator variables where $x_i = 1$ means the ith hypothetical primitive is selected in the minimal subset. The selection problem can be formulated by

Table 1
Comparison of primitive quality over processed models.

Model name	# of primitives			Coverage (percentage)			Distance error ($\times 10^{-3}$)			Normal error (degrees)		
	(I)	(II)	(III)	(I)	(II)	(III)	(I)	(II)	(III)	(I)	(II)	(III)
block	14	14	14	**99.98**	**99.98**	**99.98**	**0.08**	0.37	0.69	**1.24°**	1.33°	1.43°
casting	61	41	40	**98.80**	97.93	84.90	**0.33**	0.53	1.97	**2.28°**	3.53°	5.03°
coupling-down	44	74	67	**99.99**	76.54	80.83	0.27	**0.07**	0.34	**1.07°**	3.84°	1.83°
cover-rear	45	28	28	**100.00**	87.79	87.79	**0.04**	0.11	0.15	**0.51°**	1.38°	3.11°
crank	169	84	83	**99.34**	87.73	80.83	0.83	**0.14**	0.15	**1.68°**	1.75°	1.89°
grayloc	112	60	60	**93.77**	92.60	92.60	**1.69**	4.22	10.64	**6.14°**	10.72°	15.14°
lamp	34	29	29	**100**	96.31	96.31	**0.75**	1.23	1.02	**4.32°**	8.82°	12.04°
master-cylinder	78	32	32	**97.19**	93.81	93.81	**1.66**	2.98	10.61	**8.21°**	10.51°	12.80°
mech. part	179	59	59	**94.05**	76.17	76.17	**1.17**	7.68	7.54	**7.23°**	8.81°	11.32°
mug	5	6	n/a	**99.99**	99.98	n/a	**0.28**	0.34	n/a	**2.47°**	2.50°	n/a
oil pump	155	108	108	**96.00**	86.31	86.31	1.85	1.15	**1.14**	**8.08°**	8.21°	8.20°
pulley	49	49	49	**96.76**	95.25	95.25	**3.60**	4.23	4.49	**9.11°**	12.00°	12.02°
pump-carter	63	57	57	**98.61**	92.87	92.87	0.30	**0.16**	2.30	2.63°	**1.84°**	7.07°
rolling-stage	42	18	18	**99.94**	91.36	91.36	**0.32**	1.16	3.75	**3.43°**	3.79°	6.87°
shaft	12	12	39	**99.80**	84.31	70.71	**0.59**	1.83	1.81	4.17°	4.77°	**3.17°**
stator	12	12	n/a	**100.00**	99.99	n/a	**0.47**	0.80	n/a	**2.87°**	3.12°	n/a
stub-axle	165	91	91	**98.49**	90.05	90.05	**0.68**	1.19	5.02	**4.17°**	4.77°	7.00°
wheel	58	12	12	97.44	**98.02**	**98.02**	**0.21**	0.59	0.42	**3.57°**	6.52°	4.59°

(I): Ours; **(II)**: RANSAC; **(III)**: GlobFit.

$$\min \quad \sum_{i=1}^{|H|} x_i \tag{1a}$$

$$\text{subject to} \quad x_i \in \{0,1\} \qquad \forall i = 1, \ldots |H| \tag{1b}$$

$$\sum_{i\,:\,p \in H_i} x_i \geq 1 \qquad \forall p \in P \tag{1c}$$

The objective in (1a) clearly minimizes the number of selected hypothetical primitives. The constraint in (1c) asserts that the selection covers every element of the P. Another way of writing (1c) is $\bigcup_{i\,:\,x_i=1} H_i = P$.

The problem of identifying the smallest sub-collection from a set of collections whose union equals a known universe (set cover) is a classical NP-complete combinatorial problem (Korte and Vygen, 2012). We apply the randomized rounding algorithm (Vazirani, 2001) to solve this optimization. The algorithm first computes an optimal fractional solution x to the linear programming relaxation of the original binary programming. After that the fractional solution must be converted to an integer solution (and thus a solution of the original problem). The main conversion technique is to use randomization, and then to use probabilistic arguments to bound the increase in cost due to the rounding (based on the probabilistic method from combinatorics). Probabilistic arguments are used to show the existence of discrete structures with desired constraints in (1b) and (1c).

3.6. Boundary refinement

As a post-processing step, final primitives are grown to unlabeled regions using the *distortion minimizing flooding* algorithm (Cohen-Steiner et al., 2004; Yan et al., 2012) with normal deviation as the error metric under the constraint of distance (ϵ) and normal (α) tolerances. The reasons for doing this are two-fold: to fill holes between the primitives' boundaries and to fix missing circles in the earlier step. After that, we smooth the boundary between pairs of adjacent primitives using Graph-Cut-based technique (Boykov et al., 2001) similarly to Yan et al. (2012).

4. Experimental results

We run our algorithm on several mechanical models downloaded from the AIM@SHAPE Repository and Archive3D with various complications (see Table 1 for the list of our processed models). All experiments are executed on a PC with Intel Core i5 3.2 GHz CPU and 8 GB RAM. Our segmentation framework has three parameters: the distance (ϵ) and normal (α) tolerances for assessing a point on a primitive and the slice thickness (τ). They are fixed to the default values (ϵ is 0.5 percent of the diagonal of the point cloud's bounding box; $\alpha = 20°$; $\tau = \epsilon$). For comparison purpose, we apply the RANSAC-based segmentation (Schnabel et al., 2007) and the GlobFit optimization (Li et al., 2011) on the same data with parameters set to their default values.

Fig. 9 is the projection of the point cloud onto our segmented primitives, which is a rough reconstruction of the 3D models. As we can see, the projected points are highly consistent with the underlying surface, which illustrates the accuracy as well as the quality of our detected primitives.

Fig. 9. Projection of the point cloud onto the detected primitives obtained from our segmentation algorithm. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

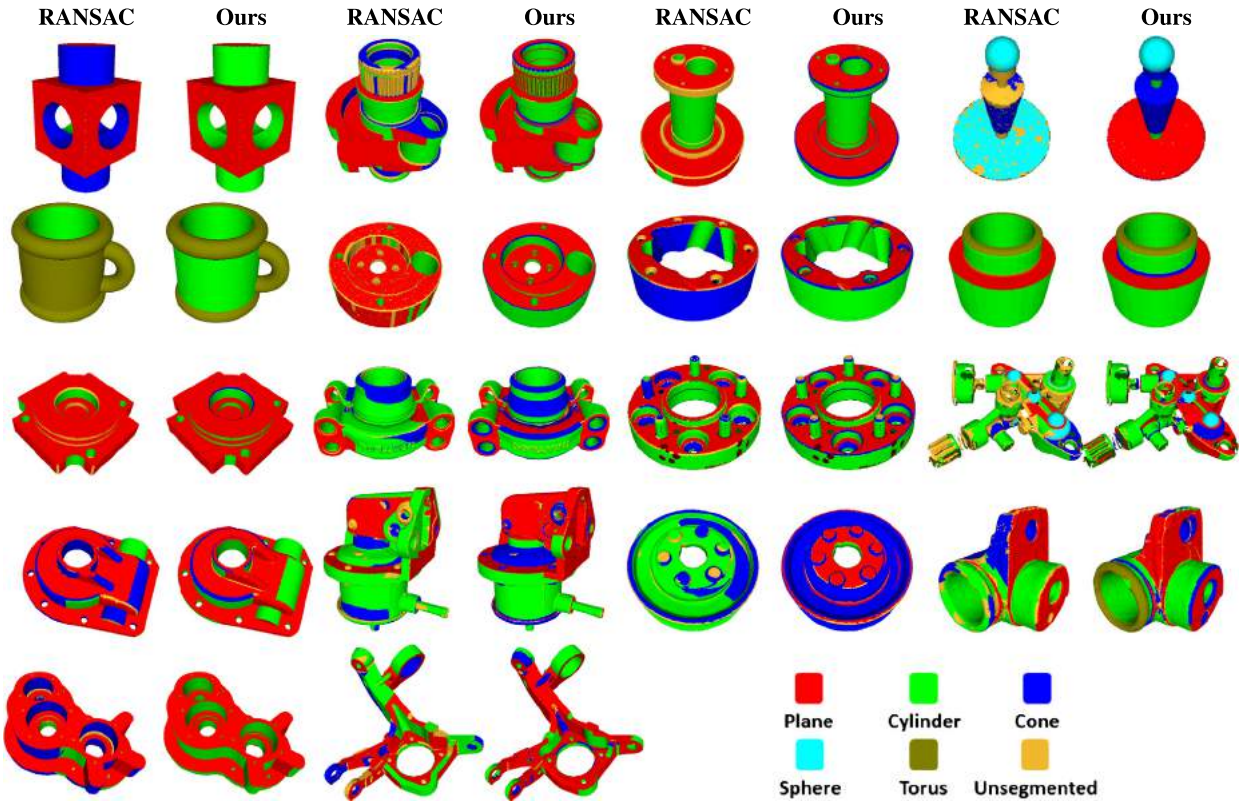

Fig. 10. Comparison primitive type recognition between our segmentation and RANSAC-based segmentation (Schnabel et al., 2007). **Odd columns**: RANSAC's result; **Even columns**: Our result. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Figs. 1 and 10 show a side-by-side comparison of our segmentation results with that of the RANSAC-based approach. We use random colors to differentiate between segmented primitives. For primitive type, we use red for plane, green for cylinder, blue for cone, cyan for sphere and olive for torus (gold color indicates unsegmented region). For simple and clean model such as the *block*, *cover-rear*, *crank*, *pump-carter* and *stator*, the RANSAC-based results are acceptable, though some primitive types are incorrect and some parts remain unsegmented. Moreover, RANSAC-based approach over-segments the *shaft* and *coupling-down* models. Our result, in contrast, not only identifies correct primitive types but also segments a better level of details.

More complicated models include the *grayloc*, *mechanical part*, *master-cylinder* and *stub-axle*. Our results are clearly better than the RANSAC's in terms of both level of details and segmentation quality. A lot of small parts with accurate primitive types can be well-captured by our approach while the RANSAC-based method typically either misses or detects with wrong primitive types. The *grayloc*, *mechanical part*, *master cylinder* and *stub-axle* are very challenging because there are many blending regions and a significant amount of noise on both the points' positions the normals. This is the case where the

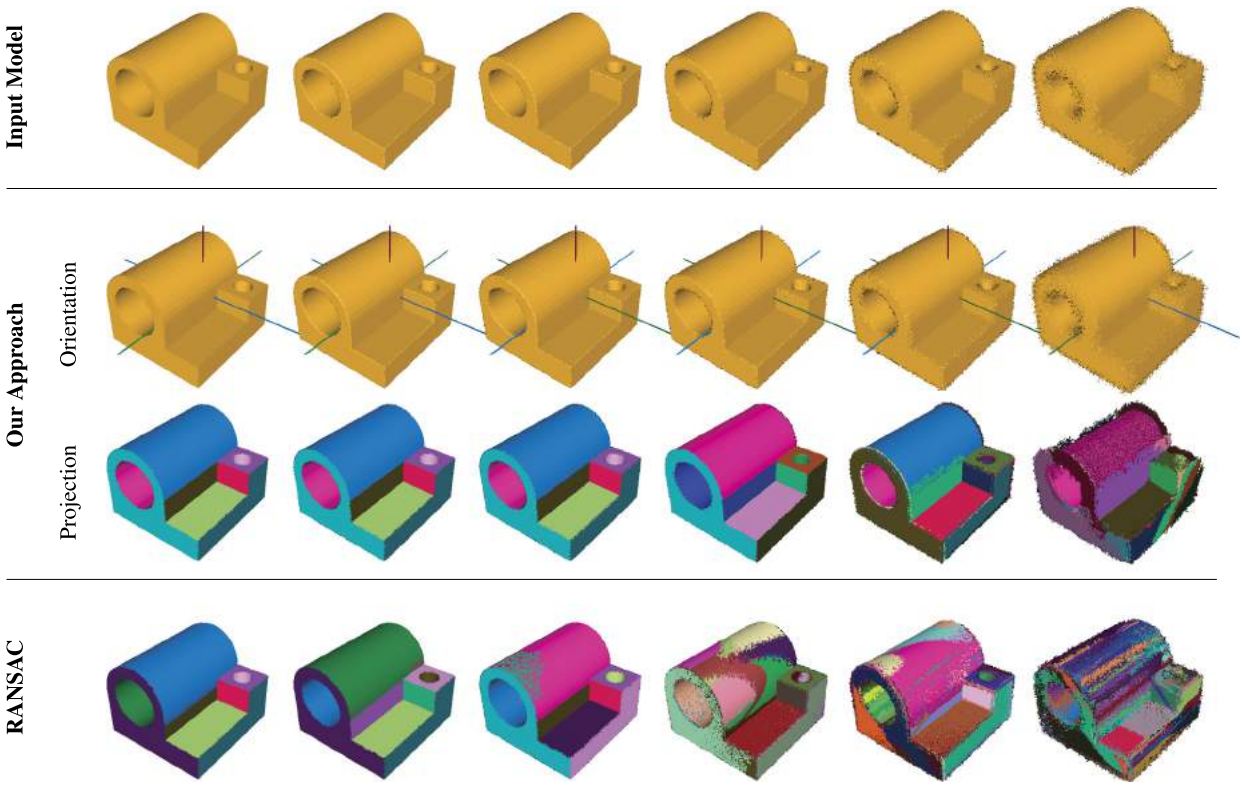

Fig. 11. Comparison of the robustness between our segmentation and RANSAC-based segmentation (Schnabel et al., 2007) on the *joint* model. Noise (on both points' positions and normals) increases from left to right, starting with a clean model. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

local-based segmentation approaches are often trapped by local sub-optimal primitives (and have no general way to fix them). On the contrary, our algorithm really shines as it exhibits well-defined structural primitives from large to small, even on the curvy surface such as the *master cylinder*.

Fig. 1 shows a side-by-side comparison of the quality of the segmentation results between our approach, the RANSAC-based approach and the GlobFit approach. We project the associated points onto the segmented primitives and display the primitives' axes by the black lines. As we can see our method demonstrates its strength in discovering the global relationship between the primitives because we restrict the search space for detecting the primitive and thus avoid the locally optimal primitives. On the other hand, RANSAC-based approach is still subject to local estimation and may produce locally optimal primitives. If the initial RANSAC's segmentation is wrong, it is very hard, if not impossible, for the GlobFit approach to correct it and sometimes GlobFit even makes it worse (e.g. the *grayloc*, *master-cylinder*).

Fig. 11 shows the comparison between the RANSAC-based method and ours on the *joint* model under different levels of noise with numerical statistics in Table 3. Starting with a clean model, we gradually increase the (Gaussian) noise characterized by the noise level σ. With the clean model (first column) ($\sigma = 0$) or very little noise (second column) ($\sigma = 0.01$), both approaches give consistently good segmentation results. However, when noise increases, RANSAC-based approach tends to produce local optimal primitives (and even wrong primitive types) while our dimension reduction segmentation approach is still robust to a certain degree. Even when the model is heavily corrupted by noise (the last column) ($\sigma = 0.2$), our model's major orientations are still correctly estimated and some of the primitives are extracted in good shapes.

In short, RANSAC-based method, as mentioned in the introduction often produces over or under segmentation and wrong primitive type. This is because the primitive parameters are calculated locally, which is extremely sensitive to noise, and the model selection is also local. Our approach, on the other hand, narrows the solution to the model's major orientations and by examining the profile curve obtained from the cross-sectional projection, our primitive's parameters are more accurately computed and the model selection is more robust. As a result, our approach can capture small details such as the conic transition between coaxial primitives (e.g. the transition between primitives in the *crank*, *mechanical part*, *master-cylinder* and *shaft* in Fig. 10).

For quantitative comparison, some error metrics such as the number of primitives, the coverage percentage, the distance error and the normal error are evaluated for our results, RANSAC's and GlobFit's (see Table 1). There is no doubt that our method yields the best overall quality with very low variance (we cover over at least 90 percent of the model, our distance error is less than 0.5 percent of the diagonal of the point cloud's bounding box and our normal error is less than 10°).

Table 2
Timings statistics (in seconds) of each step in our method on processed models.

Model name	# of points	(I)	(II)	(III)	(IV)	(V)	(VI)	(VII)	(VIII)	Total
block	850,873	1.01	10.52	0.13	54.98	0.23	21.15	1.31	20.23	109.56
casting	911,321	7.14	101.82	0.16	20.27	0.86	1.62	2.35	51.07	185.29
coupling-down	1,190,493	1.76	14.53	0.16	64.87	0.21	35.54	6.13	13.76	136.96
cover-rear	990,663	1.96	9.63	0.14	43.01	0.12	26.86	3.26	3.74	88.72
crank	1,213,649	9.96	18.58	0.29	26.91	0.84	12.34	14.01	43.76	126.69
grayloc	1,272,891	9.99	270.39	0.32	185.29	0.96	5.78	4.12	112.82	589.67
lamp	212,365	4.13	28.42	0.08	35.58	0.85	4.32	0.25	57.90	131.53
master-cylinder	1,244,445	9.25	70.87	0.26	277.55	0.98	138.36	23.75	56.54	577.56
mech. part	529,006	4.02	258.91	0.23	103.37	0.52	112.23	36.47	79.75	595.50
mug	758,597	3.45	85.74	0.10	65.42	0.41	39.64	1.34	3.63	199.73
oil pump	1,020,244	7.81	147.11	0.68	175.40	0.86	5.90	6.17	81.89	425.82
pulley	1,366,550	10.72	209.38	0.12	29.10	0.69	1.05	1.01	92.18	344.25
pump-carter	1,900,511	3.29	54.74	0.17	69.66	0.90	50.75	7.90	23.45	210.86
rolling-stage	920,433	7.23	49.47	0.15	68.51	0.36	65.52	6.38	45.34	242.96
shaft	1,102,147	2.71	54.67	0.15	83.76	0.12	13.52	6.91	32.45	194.29
stator	1,777,363	3.86	85.96	0.20	23.45	0.14	18.42	4.50	5.64	142.17
stub-axle	1,266,906	9.34	215.74	0.35	204.45	1.52	59.43	32.57	69.65	593.05
wheel	282,534	2.13	63.39	0.09	29.53	0.61	16.43	3.18	9.23	124.59

(I): Patch-based over-segmentation; **(II)**: Orientation estimation; **III**: Plane detection; **(IV)**: Circle detection; **(V)**: Axis estimation; **(VI)**: Profile curve analysis; **(VII)**: Set cover optimization; **(VIII)**: boundary refinement.

Table 3
Numerical comparison of primitive quality for Fig. 11.

Noise level	$\sigma = 0$		$\sigma = 0.01$		$\sigma = 0.02$		$\sigma = 0.05$		$\sigma = 0.1$		$\sigma = 0.2$	
	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)
# primitives	12	12	12	13	12	14	24	41	27	83	40	112
Coverage	**100**	**100**	**99.70**	99.52	**99.66**	98.61	**96.08**	95.87	**91.19**	88.39	**67.54**	60.14
Distance error	**0.17**	0.32	**0.84**	0.92	**1.61**	1.86	**2.26**	3.60	**2.46**	5.90	**2.76**	8.18
Normal error	**1.07**	1.28	**1.13**	1.15	**1.16**	1.19	**1.42**	2.04	**1.95**	4.09	**4.31**	7.72

(I): Ours; **(II)**: RANSAC.

Table 2 gives the performance of our algorithm with detailed timings on each step. The performance depends on the complexity of the model rather than its size. The axis estimation and profile curve analysis are very fast because in worst case scenario, they only have to deal with a relatively small input (i.e. grouping several hundreds of circles into connected components and fitting lines to each component for the axis estimation; the profile curve analysis processes as many as 200 points (see Section 3.4.1)). The cross-sectional circle detection is the most time-consuming stage in our framework. The Graph-Cut smoothing also accounts for a moderate amount of time. It is worth noticing that the major orientation estimation and cross-sectional circle detection steps are both highly amenable to parallelization and, hence can be made significantly faster. With our sequential implementation in an unoptimized `MATLAB` code, it takes less than 10 minutes to completely segment a 1M-point model.

The proposed method can be easily extended to surface of revolution such as hyperboloid, ellipsoid, paraboloid, etc. by extending the segmentation of the profile curve into polynomial curve segments or splines. Fig. 12 shows an example of the *bishop* model with cylinder, cone, sphere and surface of revolution together. Since surface of revolution is too flexible, it is sometimes ambiguous to differentiate between a surface of revolution and a combination of many cylinders, cones, sphere and tori. As a result, we opt to identify the *critical points* on the profile curve first. A critical point represents discontinuity in the first derivative. In order to quantify this measurement, starting with the profile curve constructed by cross-sectional circle detection, we approximate its normal variation as follows. For each point on the profile curve, we fit two lines to its *left* and *right* neighbors, respectively and use the angle difference between them as the normal variation. As we can see from Fig. 12D (we normalize the normal variation to $[0, 1]$ interval), the points on smooth parts have low normal variation while the points at discontinuities in the first derivative have large normal variation. We can obtain the critical points by first detecting local maxima of the normal variation followed by thresholding (e.g. 0.4). After the critical points are identified, the profile curve can be broken into smaller parts where each part can be fitted by a line segment, circular arc or cubic splines (Fig. 12E and F). The segmentation result is shown in Fig. 12G and H. More examples involving surfaces of revolution can be found in Fig. 13.

4.1. Limitation and future work

Our method has some limitations. First, we currently use a uniform slicing along the major direction, which is not adaptive to the data's sampling rate or level of details. A too coarse slicing prevents us from detecting small details while a too fine one makes the projection image too sparse for circles to be detected (Fig. 14), both of which result in miss-detection.

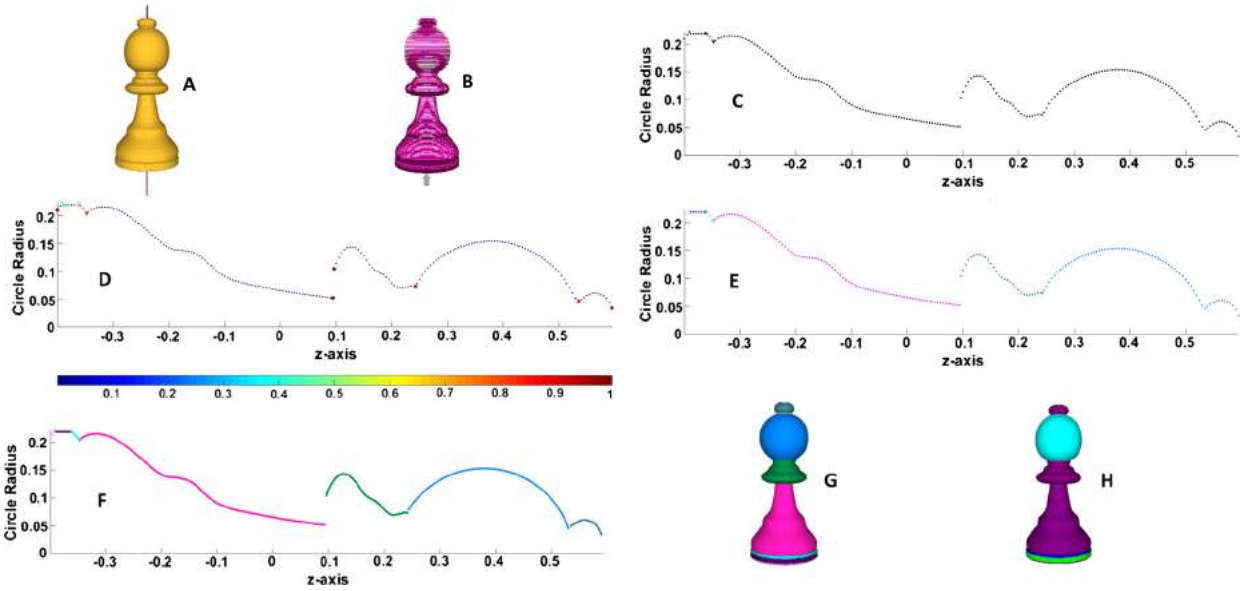

Fig. 12. Illustration of segmenting profile curve involving surface of revolution. **A**: A *bishop* model with detected axis; **B**: Cross-sectional slicing; **C**: Profile curve from detected circles; **D**: Color map of the (approximated) profile curve's normal variation where critical points are marked bigger; **E**: Segmentation of profile curve; **F**: Profile curve fitted by line segments, circular arcs and cubic splines; **G**: Segmentation result; **H**: Segmentation colored by primitive type. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Fig. 13. Our segmentation algorithm for some models with surface of revolution. **Top**: Random colors for different parts. **Bottom**: Segmentation colored by primitive type (surface of revolution is purple, others are colored similar to Fig. 3H). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

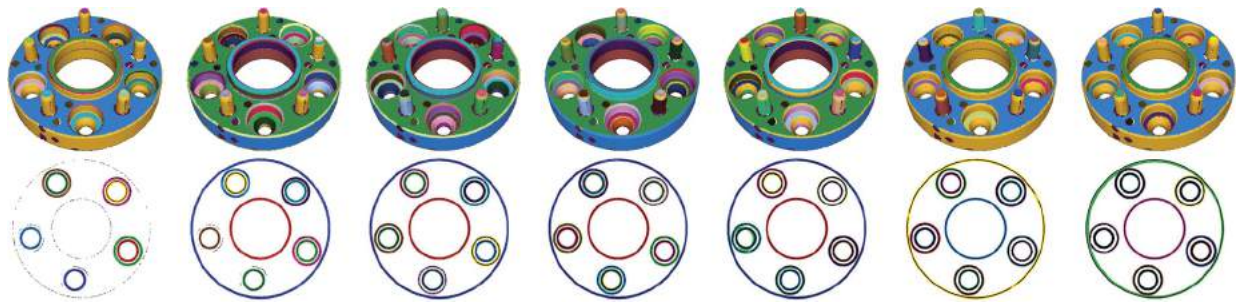

Fig. 14. Effect of various values of the slice's thickness τ. **Leftmost**: too fine slicing makes the projection image too sparse for circle detection; **Rightmost**: too coarse slicing makes the profile curve too sparse for circular primitive detection. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

However, we could generate multiple hypothetical primitives based on various thresholds and put all of them into the set-cover optimization (at the cost of extra computation). Similar strategy could be applied to the distance (ϵ) and normal

(α) thresholds. In the future work, we plan to explore the adaptive sampling techniques so that the slice's thickness could be adapted to the data density.

Second, our method in its current form supports plane, cylinder, cone, sphere, torus and extendable to surface of revolution. It, however, can be further extended to generalized cylinders with curvilinear axis. Indeed, instead of fitting a single straight line to circle centers within each connected component (Section 3.4.2), we could fit multiple line segments or even curvilinear axis which aims at generalized cylinders. Another potential future work is to fit ellipse or superellipse (Zhang and Rosin, 2003) instead of circles in the 2D cross-sectional image. With these extensions we believe the proposed framework can be extended to segment a much broader class of shapes.

5. Conclusion

We have presented our novel primitive-based segmentation algorithm for mechanical CAD models consisting of planes, cylinders, cones, spheres, tori and extendable to surfaces of revolution. Our approach first over-segments the input model, then estimates the model's major orientations and generates an over-complete set of planar and circular primitives which are then selected by the set cover optimization. For each orientation, we decompose the model into connected components of circular primitives of the same axis, then further divide each connected component into individual circular primitives by constructing a profile curve for each connected component and segmenting the profile curve into circular arcs and line segments. The final segmentation is obtained from the set cover optimization and finished by the boundary refinement. Experimental results on both synthetic and real scanned models show that our approach compares favorably with existing methods such as the RANSAC-based (Schnabel et al., 2007) and GlobFit (Li et al., 2011) approaches in terms of robustness and accuracy. In the future, we plan to extend our method to handle a broader class of primitives as discussed in Section 4.1.

Acknowledgments

We wish to thank the authors of the AIM@SHAPE Shape Repository and the Princeton Shape Benchmark (Shilane et al., 2004) for providing us the tested models. We also appreciate the authors of the RANSAC-based approach (Schnabel et al., 2007) and GlobFit (Li et al., 2011) for making their code publicly available.

References

Agathos, A., Pratikakis, I., Perantonis, S., Sapidis, N., Azariadis, P., 2007. 3D mesh segmentation methodologies for CAD applications. Comput-Aided Des. Appl. 4 (6), 827–841.
Akinlar, C., Topal, C., 2013. Edcircles: a real-time circle detector with a false detection control. Pattern Recognit. 46 (3), 725–740.
Attene, M., Falcidieno, B., Spagnuolo, M., 2006. Hierarchical mesh segmentation based on fitting primitives. Vis. Comput. 22, 181–193.
Beniere, R., Subsol, G., Gesquiere, G., Le Breton, F., Puech, W., 2011. Recovering primitives in 3D CAD meshes. In: Processing of the International Society for Optics and Photonics, vol. 7864, 78640R.
Beniere, R., Subsol, G., Gesquiere, G., Breton, F.L., Puech, W., 2013. A comprehensive process of reverse engineering from 3D meshes to CAD models. Comput. Aided Des. 45 (11), 1382–1393.
Benjamin, W., Polk, A.W., Vishwanathan, S., Ramani, K., 2011. Heat walk: robust salient segmentation of non-rigid shapes. Comput. Graph. Forum 30 (7), 2097–2106.
Boykov, Y., Veksler, O., Zabih, R., 2001. Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. 23 (11), 1222–1239.
Chen, T.-C., Chung, K.-L., 2001. An efficient randomized algorithm for detecting circles. Comput. Vis. Image Underst. 83 (2), 172–191.
Chen, X., Golovinskiy, A., Funkhouser, T., 2009. A benchmark for 3D mesh segmentation. ACM Trans. Graph. 28 (3), 73.
Cheng, Y., 1995. Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. 17 (8), 790–799.
Cohen-Steiner, D., Alliez, P., Desbrun, M., 2004. Variational shape approximation. ACM Trans. Graph. 23 (3), 905–914.
Cuevas, E., Oliva, D., Zaldivar, D., Pérez-Cisneros, M., Sossa, H., 2012. Circle detection using electro-magnetism optimization. Inf. Sci. 182 (1), 40–55.
Demir, I., Aliaga, D.G., Benes, B., 2015. Coupled segmentation and similarity detection for architectural models. ACM Trans. Graph. 34 (4), 104.
Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24 (6), 381–395.
Garland, M., Willmott, A., Heckbert, P.S., 2001. Hierarchical face clustering on polygonal surfaces. In: Processing of the Symposium on Interactive 3D Graphics, I3D '01. ACM, New York, NY, USA, pp. 49–58.
Golovinskiy, A., Kim, V.G., Funkhouser, T., 2009. Shape-based recognition of 3D point clouds in urban environments. In: International Conference on Computer Vision.
He, Y., Wang, H., Fu, C.-W., Qin, H., 2009. A divide-and-conquer approach for automatic polycube map construction, IEEE Int. Conf. on Shape Model. and App. Comput. Graph. 33 (3), 369–380.
Ho, C.-T., Chen, L.-H., 1995. A fast ellipse/circle detector using geometric symmetry. Pattern Recognit. 28 (1), 117–124.
Illingworth, J., Kittler, J., 1988. A survey of the hough transform. Comput. Vis. Graph. Image Process. 44 (1), 87–116.
Isack, H., Boykov, Y., 2012. Energy-based geometric multi-model fitting. Int. J. Comput. Vis. 97, 123–147.
Jagannathan, A., Miller, E., 2007. Three-dimensional surface mesh segmentation using curvedness-based region growing approach. IEEE Trans. Pattern Anal. Mach. Intell. 29 (12), 2195–2204.
Katz, S., Tal, A., 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans. Graph. 22 (3), 954–961.
Kim, H.-S., Kim, J.-H., 2001. A two-step circle detection algorithm from the intersecting chords. Pattern Recognit. Lett. 22 (6–7), 787–798.
Korte, B., Vygen, J., 2012. Combinatorial Optimization: Theory and Algorithms, 5th edition. Springer, Berlin, Heidelberg.
Lai, Y.-K., Hu, S.-M., Martin, R.R., Rosin, P.L., 2008. Fast mesh segmentation using random walks. In: Processing of the ACM Symposium on Solid and Physical Modeling, SPM '08. ACM, New York, NY, USA, pp. 183–191.
Lavoue, G., Dupont, F., Baskurt, A., 2005. A new CAD mesh segmentation method based on curvature tensor analysis. Comput. Aided Des. 37 (10), 975–987.
Le, T., Duan, Y., 2016. Circle detection on images by line segment and circle completeness. In: IEEE International Conference on Image Processing, pp. 3648–3652.

Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.-P., 2005. Mesh scissoring with minima rule and part salience. Comput. Aided Geom. Des. 22 (5), 444–465.
Leopardi, P., 2006. A partition of the unit sphere into regions of equal area and small diameter. Electron. Trans. Numer. Anal. 25.
Li, Y., Wu, X., Chrysanthou, Y., Sharf, A., Cohen-Or, D., Mitra, N.J., 2011. GlobFit: consistently fitting primitives by discovering global relations. ACM Trans. Graph. 30 (4), 52.
Lukacs, G., Martin, R., Marshall, D., 1998. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In: Processing of the European Conference on Computer Vision. Springer, pp. 671–686.
Mangan, A., Whitaker, R., 1999. Partitioning 3D surface meshes using watershed segmentation. IEEE Trans. Vis. Comput. Graph. 5 (4), 308–321.
Mitani, J., Suzuki, H., 2004. Making papercraft toys from meshes using strip-based approximate unfolding. ACM Trans. Graph., 259–263.
Monszpart, A., Mellado, N., Brostow, G., Mitra, N., 2015. Rapter: rebuilding man-made scenes with regular arrangements of planes. ACM Trans. Graph. 34 (4), 103.
Qiu, R., Zhou, Q.-Y., Neumann, U., 2014. Pipe-run extraction and reconstruction from point clouds. In: European Conference on Computer Vision, vol. 8691, pp. 17–30.
Rad, A.A., Faez, K., Qaragozlou, N., 2003. Fast circle detection using gradient pair vectors. In: Digital Image Computing: Techniques and Applications, pp. 879–887.
Reisner-Kollmann, I., Luksch, C., Schwarzler, M., 2011. Reconstructing buildings as textured low poly meshes from point clouds and images. In: Eurographics, pp. 17–20.
Schnabel, R., Wahl, R., Klein, R., 2007. Efficient RANSAC for point-cloud shape detection. Comput. Graph. Forum 26 (2), 214–226.
Shamir, A., 2008. A survey on mesh segmentation techniques. Comput. Graph. Forum 27 (6), 1539–1556.
Shapira, L., Shamir, A., Cohen-Or, D., 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis. Comput. 24 (4), 249–259.
Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 22 (8), 888–905.
Shilane, P., Min, P., Kazhdan, M., Funkhouser, T., 2004. The Princeton shape benchmark. In: Shape Modeling International.
Tenorth, M., Profanter, S., Balint-Benczedi, F., Beetz, M., 2013. Decomposing CAD models of objects of daily use and reasoning about their functional parts. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 5943–5949.
Theologou, P., Pratikakis, I., Theoharis, T., 2015. A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation. Comput. Vis. Image Underst. 135, 49–82.
Varady, T., Martin, R.R., Cox, J., 1997. Reverse engineering of geometric models – an introduction. Comput. Aided Des. 29 (4), 255–268.
Vazirani, V.V., 2001. Approximation Algorithms. Springer-Verlag New York, Inc., New York, NY, USA.
Vieira, M., Shimada, K., 2005. Surface mesh segmentation and smooth surface extraction through region growing. Comput. Aided Geom. Des. 22, 771–792.
von Gioi, R., Jakubowicz, J., Morel, J.-M., Randall, G., 2010. LSD: a fast line segment detector with a false detection control. IEEE Trans. Pattern Anal. Mach. Intell. 32 (4), 722–732.
Wu, J., Kobbelt, L., 2005. Structure recovery via hybrid variational surface approximation. Comput. Graph. Forum 24 (3), 277–284.
Xiao, D., Lin, H., Xian, C., Gao, S., 2011. CAD mesh model segmentation by clustering, Shape Modeling International (SMI) Conference 2011. Comput. Graph. 35 (3), 685–691.
Yamauchi, H., Lee, S., Lee, Y., Ohtake, Y., Belyaev, A.G., Seidel, H.-P., 2005. Feature sensitive mesh segmentation with mean shift. In: Processing of the International Conference on Shape Modeling and Applications. IEEE, pp. 238–245.
Yan, D.-M., Liu, Y., Wang, W., 2006. Quadric surface extraction by variational shape approximation. In: Kim, M.-S., Shimada, K. (Eds.), GMP. In: Lecture Notes in Computer Science, vol. 4077. Springer, pp. 73–86.
Yan, D.-M., Wang, W., Liu, Y., Yang, Z., 2012. Variational mesh segmentation via quadric surface fitting. Comput. Aided Des. 44 (11), 1072–1082.
Yi, B., Liu, Z., Tan, J., Cheng, F., Duan, G., Liu, L., 2014. Shape recognition of CAD models via iterative slippage analysis. Comput. Aided Des. 55, 13–25.
Yoshizawa, S., Belyaev, A., Seidel, H.-P., 2005. Fast and robust detection of crest lines on meshes. In: Proceedings of the ACM Symposium on Solid and Physical Modeling. ACM, New York, NY, USA, pp. 227–232.
Yuan, B., Liu, M., 2015. Power histogram for circle detection on images. Pattern Recognit. 48 (10), 3268–3280. Discriminative Feature Learning from Big Data for Visual Recognition.
Zhang, X., Rosin, P.L., 2003. Superellipse fitting to partial data. Pattern Recognit. 36 (3), 743–752.
Zhang, Y., Paik, J., Koschan, A., Abidi, M., Gorsich, D., 2002. Simple and efficient algorithm for part decomposition of 3-D triangulated models based on curvature analysis. In: International Conference on Image Processing, vol. 3, pp. III-273–III-276.
Zhou, Y., Yin, K., Huang, H., Zhang, H., Gong, M., Cohen-Or, D., 2015. Generalized cylinder decomposition. ACM Trans. Graph. 34 (6), 171.
Zuckerberger, E., Tal, A., Shlafman, S., 2002. Polyhedral surface decomposition with applications. Comput. Graph. 26 (5), 733–743.