Automatic method for sharp feature extraction from 3D data of man-made objects

Trung-Thien Tran¹, Van-Toan Cao¹, Van Tung Nguyen¹, Sarah Ali¹ and Denis Laurendeau¹

¹Computer Vision and Systems Laboratory, Department of Electrical and Computer Engineering, Laval University, 1065, Avenue de la Medecine, Quebec (QC), G1V 0A6, Canada
{trung-thien.tran.1, van-toan.cao.1,van-tung.nguyen.1,sarah.ali.1}@ulaval.ca, denis.laurendeau@gel.ulaval.ca

Keywords: Sharp feature; Feature extraction; Projected distance; Point cloud and mesh processing.

Abstract: A novel algorithm is proposed for extracting sharp features automatically from scanned 3D data of man-made CAD-like objects. The input of our method consists of a mesh or an unstructured point cloud that is captured on the object surface. First, the vector between a given point and the centroid of its neighborhood at a given scale is projected on the normal vector and called the ‘projected distance’ at this point. This projected distance is calculated for every data point. In a second stage, Otsu’s method is applied to the histogram of the projected distances in order to select the optimal threshold value, which is used to detect potential sharp features at a single scale. These two stages are applied iteratively with the other incremental scales. Finally, points recorded as potential features at every scale are marked as valid sharp features. The method has many advantages over existing methods such as intrinsic simplicity, automatic selection of threshold value, accurate and robust detection of sharp features on various objects. To demonstrate the robustness of the method, it is applied on both synthetic and real 3D data of point clouds and meshes with different noise levels.

1 INTRODUCTION

Digital scanning devices have been used for various applications. Due to the rapid development of scanning technologies, large sets of accurate 3D points can be collected with such devices. Therefore, more and more applications use these sensors, especially in industrial manufacturing.

Among emerging problems, sharp feature extraction from scanned data of CAD model has recently received much attention from the research community. Sharp features help to understand the structure of underlying geometry of a surface. Futhermore, it is very important on many issues such as segmentation (Lavoué et al., 2005), surface reconstruction (Weber et al., 2012) and resampling (Huang et al., 2013). Especially, most manufactured objects consist of the combination of common geometric primitives such as planes, cylinders, spheres, cones or toruses and the intersection between these primitives can be considered as sharp features. The input 3D data consists of meshes or point clouds. Therefore, existing methods are classified into two categories: mesh-based and point-based methods. Most existing methods focus only on point clouds or meshes which limits their flexibility and generality.

Several techniques (Hubeli and Gross, 2001; Ohtake et al., 2004; Hildebrandt et al., 2005) use polygonal meshes as input. In (Hubeli and Gross, 2001), a framework to extract mesh features from surfaces is presented. However, this approach is semi-automatic, that is the user is required to input a few control parameters before a solution is found. Hildebrandt et al. (Hildebrandt et al., 2005) have also proposed a new scheme based on discrete differential geometry, avoiding costly computations of higher order approximating surfaces. This scheme is augmented by a filtering method for higher order surface derivatives to improve both the stability and the smoothness of feature lines. Ohtake et al. (Ohtake et al., 2004) propose to use an implicit surface fitting procedure for detecting view and scale independent ridge-valley structures on triangle meshes. Nevertheless, the methods still require thresholding value for scale-independent parameter in order to keep the most visually important features. Moreover, the methods only show the results of the feature lines for graphics models.

Point cloud preserves the structure of the underlying surface, so it recently has been used in processing and modelling. Sharp features are normally found at the points that have large variation in curvature or
discontinuities in normal orientation of the surface. Consequently, several approaches (Demarsin et al., 2007) detect sharp features using differential geometry principles exploiting surface normal and curvature information. Nevertheless, the first step of this method is a normal-based segmentation that does not guarantee the quality of results because of noisy point data. Mérigot et al. (Mérigot et al., 2011) compute principal curvatures and normal directions of the underlying surface through Voronoi covariance measure (VCM). Then they estimate the location and direction of sharp features using Eigen decomposition. However, this method requires significant computational time for calculating the VCM.

Recently, Gumhold et al. (Gumhold et al., 2001) use PCA to calculate the penalty function that takes into consideration curvature, normal and correlation. Pauly et al. (Pauly et al., 2003) calculate surface variation using the same mathematical technique. Weber et al. (Weber et al., 2010) have proposed to use statistical methods to infer local surface structure based on Gaussian map clustering. Park et al. (Park et al., 2012) used tensor voting to infer the structure near a point. However, all of them have same problem in requiring many parameters, which is set manually.

Therefore, this paper proposes a novel algorithm for extracting sharp features automatically from 3D data, which is developed from our previous work (Tran et al., 2013). We first preprocess the scanned data by finding the neighborhood and estimating the surface normal. The keypoint of this paper is the use of Otsu’s method to detect potential sharp features at a given scale. Then an automatic threshold value is selected by Otsu’s method to detect potential sharp features at a given scale. The process is iteratively applied for incremental scales. In the end, valid sharp features are determined by multi-scale analysis as a refinement. Generally, our method focuses on two main computation steps: (i) calculating the projected distance at a single scale and automatically extracting potential sharp features. (ii) applying these calculations at multiple scales. Therefore, the strong points of the proposed method are summarized in the following:

- The proposed method is very simple, fast. Furthermore, the method accepts both meshes and unstructured point clouds as input. This allows the method to work without conversion between types of data and adds flexibility to the technique.
- No parameter needs to be set by the user since the entire process is completely automatic. This is a significant advantage over existing methods.
- Our approach explores multiple scales in order to extract sharp features accurately with certain levels of noise. In most cases, the detected features are one-point wide.
- Accurate and robust results are achieved by the method even for noisy data and complex objects.

The rest of this paper is organized as follows. Neighborhood and normal estimation for meshes and point clouds are described in Section 2. The main proposed algorithm is described in Section 3. Results and discussion are presented in Section 4, while Section 5 draws some conclusions on the proposed method.

### 2 POINT NEIGHBORHOOD AND SURFACE NORMAL ESTIMATION

The input 3D data can consist of meshes or point clouds that have different structures to represent the underlying surface of a given object. The neighborhood around a point is used to estimate the normal at that point. Therefore, the problem of finding the neighborhood of each point is addressed first and the problem of estimating the normal vector for each point is then described according to the type of data.

#### 2.1 Point clouds

A point cloud is defined as a set of points within a three-dimensional coordinate system. This type of data acquired from the scanners represents noisy samples of the object surface. The information about connectivity and surface normal of underlying surfaces is lost completely in the sampling process. Moreover, it resides implicitly in the relationship between a sampled point and its neighborhood. Hence, the first step in surface normal estimation at a given point is to find the number of points around this point. In this paper, the data is structured by using k-d trees (Weiss, 1994) to search the k nearest neighborhood.

Classical principal component analysis PCA in (Hoppe et al., 1992) is usually selected for normal estimation because of its simplicity and efficiency. However, a point cloud contains a set of non-uniform points, so a weight that is derived from its distance to the centroid should be assigned to each point in the neighborhood in order to improve the robustness of the method. A covariance matrix $CV$ of point $p_j$ is established in Equation (1) and analyzed, the eigenvector corresponding to the smallest eigenvalue is considered as the normal vector $n_i$ at point $p_i$.

$$CV_i = \sum_{j \in N(i)} \mu_j (p_j - \overline{p_i})^T (p_j - \overline{p_i}) \in R^{3x3} \quad (1)$$
Figure 1: Overall procedure of the proposed method.

where $p_i = \frac{1}{k} \sum_{j \in N(i)} p_j$ is $p_i$’s local data centroid, $N(i)$ is the set of $k$ points in the neighborhood around point $p_i$. $\mu_j$ is calculated by the Gaussian function $\mu_j = \exp(-\frac{d_j^2}{2k})$ of the distance $d_j = \|p_j - p_i\|$. The scale $k$ is the number of neighboring points.

2.2 Meshes

A polygonal mesh is a collection of vertices, edges and faces that represents the shape of an object. This type of data is different from point clouds because it contains edges connecting the vertices. Therefore, an approach is needed to find the neighborhood of a point based on this connection. A $k$-ring neighborhood of a vertex $p_i$ is defined as a set of vertices that are connected to $p_i$ by at most $k$ edges.

A simple and effective method for normal vector estimation is to calculate it as the weighted average of the normal vectors of the triangles formed by $p_i$ and pairs of its neighbors inside 1-ring. The basic averaging method is according to Equation (2):

$$n_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} \omega_j \frac{\left[ p_j - p_i \right] \times \left[ p_{j+1} - p_i \right]}{\| \left[ p_j - p_i \right] \times \left[ p_{j+1} - p_i \right] \|}$$

(2)

where $p_j$ is one of the points inside 1-ring. $\omega_j$ is a normalized weighting factor that can be calculated by angle-weighted, area-weighted and centroid-weighted methods. A good summary of these methods can be found in (Jin et al., 2005). In this paper, we use a simple method proposed by Gouraud (Gouraud, 1971) ($\omega_j = 1$).

3 PROPOSED METHOD FOR EXTRACTING SHARP FEATURES

In this section, we describe our algorithm for extracting sharp features at multiple scales. A block diagram of the method is shown in Figure 1. First, the projected distance definition and the calculation procedure for each point are introduced in Section 3.1. Then a threshold value is selected automatically by Otsu’s algorithm in Section 3.2. After this step, potential sharp features are detected at a given scale. These steps are applied iteratively for multiple scales. The approach for detecting valid and reliable sharp features is described in Section 3.3.

3.1 Projected distance calculation

Projected distance is a value calculated by projecting the vector between a given point and the centroid of its neighborhood at a given scale on the normal as given in Equation (3). The procedure is shown in Figure 2.

$$pdis(i) = \text{abs}(\langle p_i - \bar{p}_i \rangle \cdot \vec{n}_i) = \text{abs}(\| p_i - \bar{p}_i \| \cdot \cos \theta_i)$$

(3)

where $\theta_i$ is the angle between vector $(p_i - \bar{p}_i)$ and vector $\vec{n}_i$, which is the surface normal at point $p_i$ calculated in Section 2.1 or Section 2.2 according to the type of input data. $\bar{p}_i$ is the centroid of neighborhood points determined with the k-d tree for point clouds Section 2.1 or k-ring for meshes Section 2.2. Angle $\theta_i$ between vector $(p_i - \bar{p}_i)$ and vector $\vec{n}_i$ can be greater or smaller than 90 degrees, but the projected distance is always a positive number. That is why the direction of the normal vector is not considered here as it is in (Hoppe et al., 1992), only the orientation is used in our work.

Figure 2: Procedure for calculating the projected distance at each point. The green star is the centroid of the neighborhood.
The projected distance value expresses the structure of the underlying surface supported by the neighborhood. Therefore, the central idea of our method is that the projected distance is almost zero for a point lying on a smooth surface area. However, the projected distance has a large value if a point is located on or near a sharp feature, as shown in Figure 3. Our method uses this property of the projected distance to assess whether a point is located on a sharp feature or not.

### 3.2 Automatic selection of a threshold value for the detection of sharp features

The projected distances for every data point (point cloud or mesh) was computed at the same scale in the previous section. This section help extract potential sharp features at a scale. First, every projected distance is normalized between [0,1] as displayed in Figure 4(a). Then a threshold value for the projected distance needs to be calculated to evaluate whether or not a given point is located on a sharp feature, as shown in Figure 3. Our method uses this property of the projected distance to assess whether a point is located on a sharp feature or not.

![Figure 3: Projected distances calculated at smooth (blue points) and sharp areas (red points). Green points are the centroid of data.](image)

### 3.3 Multiscale processing and refinement

Potential sharp features are extracted from the steps described in the two previous sections at a single scale. A real feature can be recognized at multiple scales by a human. Besides, the result at a single scale may contain several false features because of noise. Therefore, the proposed method calculates the projected distance and extracts sharp features automatically at multiple scales to eliminate false sharp features.

Valid sharp features are those recorded at all scales. Hence, Equation (5) explains the means of deciding whether or not a point is valid sharp feature:

$$F(i) = \begin{cases} 
1 & \text{if } \sum_{s=1}^{ns} f(i) = ns \\
0 & \text{if } \sum_{s=1}^{ns} f(i) < ns 
\end{cases}$$

where $F(i)$ is the final sharp feature map, $ns$ is the number of scales that are investigated for point clouds or meshes in Section 4.2 and $f(i)$ is calculated from Equation (4) in Section 3.2. The results of the multiscale sharp feature extraction method are shown in Figure 9 and Figure 10 for point clouds and meshes, respectively.

### 4 RESULTS AND DISCUSSION

We have tested the proposed method on various complex models corrupted by different noise levels. Furthermore, the results provided by our method are compared with some other methods using mean curvature (Yang and Lee, 1999; Watanabe and Belyaev, 2001), normal vector (Hubeli and Gross, 2001) and
surface variation in Pauly’s method (Pauly et al., 2003) for point-based methods and Ohtake’s method (Ohtake et al., 2004) for mesh-based methods. Computation time and limitations of our method are also reported.

4.1 Comparison with other methods

4.1.1 Point-based method

As mentioned above, a sharp feature is a point at which the surface normal vector (Demarsin et al., 2007) or the curvature (Yang and Lee, 1999) is discontinuous in value, so the Mean curvature and the normal difference between adjacent triangles are used as the factors for determining sharp feature locations. It assigns a value \( w_i \) defined by normal difference between given point and its neighborhood (Hubeli and Gross, 2001).

\[
w_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} \cos \left( \frac{n(i) \cdot n(j)}{||n(i)||} \right)^{-1}
\]

where \( n(i), n(j) \) and \( N(i) \) are defined in Section 2.1.

In (Pauly et al., 2003), Pauly et al. use surface variation \( \sigma_i \), which is calculated as the ratio between the smallest eigenvalue and sum of the eigenvalues of the covariance matrix \( CV_i \) in Equation 1 established by the neighborhood around a given point.

\[
\sigma_i = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}
\]

Therefore, we will compare our projected distance with these three parameters. We investigate these methods by applying them to the fandisk point cloud at a single scale \( k=16 \), that was mentioned as the optimal one in (Weber et al., 2010). Moreover, the other three methods are not automatic and require some threshold values to be set, so our automatic threshold estimation is applied to them. The results of the four methods are shown in Figure 5 and Figure 6.

Observing Figure 5 and Figure 6 indicates that our method is able to detect sharp features that are one point wide. Mean curvature, normal and Pauly’s methods detect rather thick features including many potential feature points. Because the mean curvature and the surface normal do not reveal shape information of the underlying surface as the projected distance does. The projected distance is small at smooth and near-edge points and much smaller than the value at sharp features. Therefore, the feature lines detected by the proposed method are thinner and more accurately located. The results prove that ‘projected distance’ is a better factor that can be used for detecting sharp features.

Figure 7 shows that even when tuning the threshold value manually to improve its performance, Pauly’s method still fails to produce thin and accurate features. For instance on threshold values for surface variation set at \( T=0.1 \), 0.15 and 0.2, the results contain many spurious sharp features. With threshold \( T=0.25 \), some good features are missing and the result still contains spurious sharp features (black ellipse in Figure 7).

Furthermore, the absolute accuracy of our method is demonstrated in Figure 6 and Table 1. A synthetic cube point cloud was generated with 91 sharp features composed of 386 data points. Mean curvature, normal and Pauly’s method detect many spurious sharp features while our method extracts all 91 sharp features correctly and at the right location.

4.1.2 Mesh-based method

In this section, we compare the results of the mesh-based method to Ohtake’s method (Ohtake et al., 2004). The method is applied to synthetic cube models with different scales and threshold values. The results show that our method is able to detect sharp features more accurately and with fewer spurious features compared to the mesh-based method.
His method uses an implicit surface fitting procedure for detecting ridge-valley structure on surface approximated by dense triangulation. His implemented code is available from the author’s website and the selected threshold settings are the default values for scale-independent parameter, connecting and iterations. The results are shown in Figure 8. In his results, ridge and valley points are displayed in red and blue, respectively. We find that valley points are located on smooth areas while ridge points are missing at sharp features such as in the edge of block model. In the other hand, our method detects sharp features accurately and does not include valley points (blue) as in his method. Moreover, his method detects many spurious sharp features at the small and thin holes as shown in Figure 8.

4.2 Results for various models

The method is applied on other meshes and point clouds. Most models are taken from the AIM@SHAPE Shape Repository and the Octa-flower model from Ohtake. With point cloud models, we investigated the behaviour of the proposed method at multiple scales \(k=(12,16,\ldots,40)\) and Figure 9 shows the results for several point cloud models. Although these models contain many different primitives including planes, spheres, cylinders and even free-form primitives, our method can detect sharp features accurately in each model. For meshes, 1,2,3 and 4-ring neighborhoods are used to calculate the projected distances for every vertex. Figure 10 shows the results for some mesh models and demonstrates the efficiency of the method to detect thin sharp features.

4.3 Computation cost

The proposed method are executed on Matlab on a 3.2 GHz Intel Core i7 platform. Table 2 for point clouds and Table 3 for meshes show computation time of the method. This process is implemented without parallel processing. We believe that a C++ implementation would speed up the process significantly and parallel processing could be applied to process multiple scales simultaneously.

4.4 Robustness to noise

To illustrate the robustness of the proposed method to a certain level of noise, some results for noisy fan-disk point cloud models and real data are presented in this section. First, Gaussian noise with zero mean and standard deviations of 1-5 % of the average distance

<table>
<thead>
<tr>
<th>Model name</th>
<th>Points</th>
<th>Total time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fandisk</td>
<td>6475</td>
<td>1.05</td>
</tr>
<tr>
<td>Smooth-feature</td>
<td>6177</td>
<td>1.01</td>
</tr>
<tr>
<td>Double-torus2</td>
<td>2668</td>
<td>0.44</td>
</tr>
<tr>
<td>Block</td>
<td>12909</td>
<td>2.01</td>
</tr>
<tr>
<td>Octa-flower</td>
<td>12868</td>
<td>2.11</td>
</tr>
<tr>
<td>Cylinder</td>
<td>50000</td>
<td>8.34</td>
</tr>
<tr>
<td>Box</td>
<td>50000</td>
<td>8.21</td>
</tr>
<tr>
<td>BEVEL2</td>
<td>64250</td>
<td>4.92</td>
</tr>
<tr>
<td>Real data1</td>
<td>108349</td>
<td>17.4</td>
</tr>
<tr>
<td>Real data2</td>
<td>30751</td>
<td>4.99</td>
</tr>
</tbody>
</table>
Figure 11: Sharp features extracted from fandisk model with different levels of noise.

Figure 12: Results on noisy data with outliers and holes: (a) Original object and markers. (b) Scanned data with outliers and holes. (c) Pauly’s method (Pauly et al., 2003). (d) Ohtake’s method (Ohtake et al., 2004). (e) Our Method.

Table 3: Timing of our proposed method for mesh models.

<table>
<thead>
<tr>
<th>Model name</th>
<th>Points</th>
<th>Faces</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fandisk</td>
<td>6475</td>
<td>12946</td>
<td>1.48</td>
</tr>
<tr>
<td>Smooth-feature</td>
<td>6177</td>
<td>12350</td>
<td>1.40</td>
</tr>
<tr>
<td>Double-torus2</td>
<td>2668</td>
<td>5340</td>
<td>0.61</td>
</tr>
<tr>
<td>Block</td>
<td>2132</td>
<td>4272</td>
<td>0.52</td>
</tr>
<tr>
<td>Octa-flower</td>
<td>7919</td>
<td>15834</td>
<td>1.79</td>
</tr>
<tr>
<td>Trim-star</td>
<td>5192</td>
<td>10384</td>
<td>1.18</td>
</tr>
<tr>
<td>Sharp-sphere</td>
<td>8271</td>
<td>16538</td>
<td>1.87</td>
</tr>
<tr>
<td>Casting</td>
<td>5086</td>
<td>10204</td>
<td>0.42</td>
</tr>
<tr>
<td>Real data1</td>
<td>108349</td>
<td>214189</td>
<td>21.4</td>
</tr>
<tr>
<td>Real data2</td>
<td>30751</td>
<td>60783</td>
<td>6.84</td>
</tr>
</tbody>
</table>

between points was added to the point cloud data. Then our method is applied to this data corrupted by noise. Figure 11 shows the results for different noise levels. For a noise level of (1-3)% acceptable results are still achieved. For a noise level of (4-5)% a small part of the model is contaminated by spurious feature points. Therefore, applying a prior denoising algorithm could help the proposed method to obtain better results.

Figure 12 shows the results on real scanned data, which contains many outliers and holes. Figure 12(b). The data is captured by the GO! scan sensor (http://www.goscan3d.com). Other methods such as Pauly’s (Figure 12(c)) and Ohtake’s (Figure 12(d)) methods are also tested. Observing the results indicates that Pauly’s method misses several sharp features, while Ohtake’s method fails totally for noisy data.

4.5 Limitations

Although the proposed method can extract sharp features accurately, it also depends on the level of noise of the captured data and also on the distribution of the data. For instance for areas with low density, the improvement brought by the weighting factor is minor. Therefore, preprocessing stages such as resampling...
and pre-filtering would be necessary in these cases to improve the results. Figure 13 shows the results after removing outliers and filling holes in real scanned data, sharp features are detected accurately.

5 CONCLUSIONS

We have introduced an automatic method for extracting sharp features from 3D data. This method accepts both meshes and point clouds as input. The projected distance is calculated at multiple scales, which is supported by the k nearest neighborhood in point clouds or the k-ring neighborhood in meshes. Then realizable sharp features are extracted automatically by using Otsu’s method. In addition to its simplicity, the proposed method outperforms other methods presented in the literature.

In the future, the algorithm could be improved by connecting discrete sharp features into parametrized curves for obtaining high-level descriptions. Furthermore, we plan to use the results of our algorithm for practical problems such as remeshing or mesh generation.

ACKNOWLEDGEMENTS

The authors thank the AIM@SHAPE Shape Repository and Ohtake for making their codes and models available. Moreover, the authors would like to thank their colleagues for fruitful discussions. This research project was supported by the NSERC/Creaform Industrial Research Chair on 3-D Scanning.

REFERENCES


