A Fast and Accurate Denoising Algorithm for Two-Dimensional Curves

Jie Shen and David Yoon

University of Michigan-Dearborn, {shen | dhyoon}@umich.edu

ABSTRACT

In this paper we propose a new concept, sparing least-squares denoising, which is used to significantly reduce the time cost of denoising algorithms that utilize least-squares fitting techniques. This idea is implemented by a fast feature-preserving denoising method, partition of feature and non-feature regions, and introduction of a new index, vertex sparsity. We use this index to decide the location on curves where an expensive least-squares fitting is really needed. Our numerical experiments indicate that our new approach yields an average 2.93 times saving in the computation time without remarkable sacrifice on denoising accuracy, compared to existing least-squares fitting methods.

Keywords: Metrology, Quality Control, Laser Sensor, Noise.

1. INTRODUCTION

Measurement noise is an intrinsic problem with various existing laser and white light sensors such that denoising is a crucial step for accurate metrology and quality control. The path/curve noise is also a current problem in tool path generation from CAD/CAM design surfaces. In the past, different surface smoothing algorithms have been proposed [1-5]. Among them, least-squares fitting methods [4, 6] yield a very high denoising accuracy, and therefore are quite suited for metrology and quality control. However, a least-squares fitting method is normally five to ten times more expensive in the computational time cost, which hinders its wide applications, especially when many boundary curves need to be processed or near real-time requirement is imposed.

The objective of this paper is to propose an approach that is significantly faster than the traditional least-squares fitting methods, and in the meantime maintains its high denoising accuracy. A piecewise algorithm is used to speed up least squares whenever possible. Its unique contribution is the introduction of a new concept, *sparing least-squares fitting*, and a new index, *vertex sparsity*. In this paper, we limit our effort in two-dimensional curve cases, especially in the format of polylines. In other words, the output format of our algorithm is a polyline. However, the concept can be easily extended to three-dimensional cases.

The remaining of this paper is organized as follows. In Section 2, a new method of sparing least-squares denoising is introduced. Then, numerical experiments are reported and discussed in Section 3. Finally, some concluding remarks are given in Section 4.

2. SPARING LEAST-SQUARES DENOISING

In [6], we proposed the concept of hybrid smoothing or denoising to achieve an accurate denoising. However, the main shortcoming of that approach is high computational cost associated with least-squares fitting technique. In this paper, we follow the overall approach used in [6] with a significant improvement in reducing computational time cost.

2.1 Feature-Preserving Pre-smoothing

The pre-smoothing is an important pre-processing step in handling a denoising task. It is possible that a signal/noise ratio is so low that sharp features could not be correctly detected without a pre-smoothing. If a filter is randomly selected for pre-smoothing, even though the noise could be suppressed, the sharp features would be also smoothed out. This is not what we expect and creates the dilemma of a chicken-and-egg problem.

The authors proposed to use a median filter as a first step to solve this chicken-and-egg problem. The unique feature of the median filter is its theoretical guarantee to maintain global sharp features, which are the features that should not disappear in a denoising process. In this way, it is possible to correctly detect all the feature regions on the pre-smoothed curves, and then to use this partition information to guide a real denoising process on the initially-noised curve. As a result, our approach works properly even when signal/noise ratios are very low.

2.2 Partition of Curves by a Feature-Preserving Pre-smoothing Procedure

There are many existing approaches for surface or region partition [7-9]. We don't plan to create a new one. Instead, a commonly-used region grow method is adopted in this study. In order to make a partition successful, the signal/noise ratio should be high enough. Otherwise, a pre-smoothing is necessary, and global sharp features must be maintained during the pre-smoothing process.

In this paper, a region grow method is applied on curves that are pre-denoised. A curve is divided into non-feature and feature regions. The global sharp features are located in the feature regions, while the non-feature regions are smooth and/or contain just local noises.

2.3 Hybrid Denoising

After the partition of a curve, the result of pre-smoothing is discarded except the partition information of feature and non-feature regions. A hybrid scheme is applied on the input noised curve discriminately over feature and non-feature regions. In feature regions, a median filter is used to maintain the sharp global features, while in non-feature regions a sparing least-squares fitting treatment is proposed for the sake of high denoising accuracy and high computational efficiency.

The basic idea of the sparing least-squares fitting treatment is to avoid the least-squares fitting as much as possible because it is computationally time consuming. If the vertex density is very high in a local region, then the local least-squares fitting of a quadratic or cubic curve segment in this region becomes unnecessary, because an approximation error becomes smaller even with a linear approximation. On the contrary, if the vertex density is very low in a local region, a local least-squares fitting becomes crucial to achieve a low approximation error, which in turns leads to a high denoising accuracy.

In order to implement the above idea, the first thing that needs to be done is the introduction of a vertex sparsity index (VSI), which can be used to identify if vertices are distributed sparsely or densely on a curve. We define VSI as a ratio of the distance between the first and fourth vertices in a two-ring vertex neighborhood, as shown in Figure 1, to the feature size of a curve. In Figure 1, the current vertex is **P**, and the first and fourth vertices in its two-ring neighborhood are $\mathbf{Q}_1$ and $\mathbf{Q}_4$, respectively. The feature size of a curve refers to the maximum length of its bounding box.

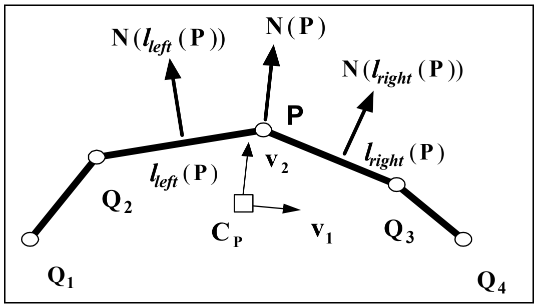


Fig. 1: A two-ring vertex neighborhood of a vertex **P**.

The second task that we need to accomplish is to find out a suitable threshold for VSI, t_{VSI}. We consider that the vertices on a curve are sparsely distributed if VSI > t_{VSI}. Otherwise, the vertices are distributed densely. In order to determine this threshold, we consider a typical case as shown in Figure 2, in which a circular sector of a circle is investigated. We define an approximation error of chord $\overline{\mathbf{AB}}$ as a ratio of the area of circular segment ABC to the area of circular sector AOBC. A distance ratio of line segment $\overline{\mathbf{AB}}$ is defined as a ratio of the distance between vertices **A** and **B** and the diameter of the circle. According to trigonometry, the approximation error, *ap*, and distance ratio, *dr*, can be expressed by

$$ap = \frac{\theta - 2\sin\frac{\theta}{2}\cos\frac{\theta}{2}}{\theta} \tag{1}$$

$$dr = \sin\frac{\theta}{2} \tag{2}$$

where θ is a central angle of chord $\overline{\mathbf{AB}}$. Table 1 gives the values of the approximation error and distance ratio for various central angles. We consider an approximation error, 0.00161, is sufficient for achieving a high denoising accuracy, as illustrated by boldface numbers in Table 1. Since VSI is a measure related to four line segments, we multiply 4 with the distance ratio, 0.0491, and let t_{VSI} equal 0.2 approximately. Figure 3 shows a vertex density on a circle, which corresponds to the approximation level dictated by t_{VSI}.

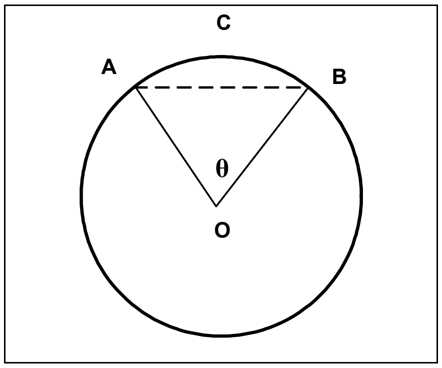


Fig. 2: A circular sector of a circle.

A circle has the minimum possible feature size for a given area among all C_1 continuous curves. On the other hand, a square is an extreme case for C_1 discontinuous curves, and the ratio of its minimum feature size to the size of the circle with the same amount of area is 0.89, which is close to unity. Therefore, it is reasonable to infer that t_{VSI}, obtained through the way described above, is an approximate bound that guarantees a dense vertex distribution, if VSI < t_{VSI}, for any arbitrary two-dimensional curve. Figure 3 shows a vertex distribution on a circle when VSI= t_{VSI}. Note that the vertex distribution in Figure 3 is scale variant, while the approximation error and distance ratio in Equation (2) are scale invariant.

θ (degree)	Approximation error	Distance ratio
180	1	1
90	0.363	0.707
45	0.0997	0.383
22.5	0.0255	0.195
11.25	0.00641	0.0980
5.625	**0.00161**	**0.0491**
2.813	0.000402	0.0245
1.406	0.000100	0.0123

Tab. 1: Approximation error and distance ratio of a chord at different central angles.

With a scale invariant threshold, t_{VSI}, available, we are ready to complete the third task in implementing the sparing least-squares denoising. For each vertex in a non-feature region, we calculate its VSI. If VSI is smaller than t_{VSI}, a mean filter is applied. Otherwise, a local least-squares fitting is used. If there are some outliers on an input curve, they must be removed first before the computation of VSI. There are some existing methods to remove outliers [10-12], and readers should choose one at their own disposal.

Our numerical experiments indicate that the mean filter is an accurate and fast denoising algorithm for non-feature regions with a dense vertex distribution, which is possible during a scanning process on a smooth curved surface. For all the non-feature regions, it is important to find a suitable denoising algorithm, because these regions could be any arbitrary smooth curved surfaces. It is not our purpose to claim the median and mean filters as our contribution. However, for the convenience of readers, we list our version of median and mean filters for polylines as Algorithms 1 and 2 below.

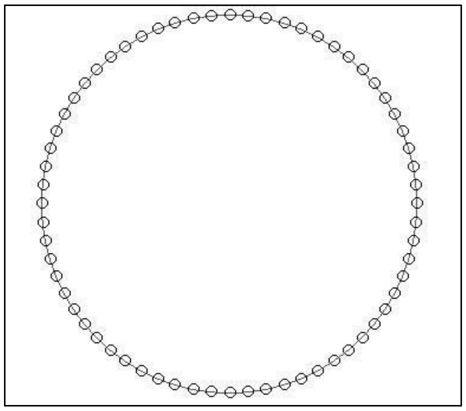


Fig. 3: Vertex distribution on a circle when VSI= t_{VSI} .

Algorithm 1: fast_median_filter(PL)
Data Structures: L = set of all line segments; V = set of all vertices.
Precondition: *PL* contains the information of a polyline, including *L* and *V*. Note that the first and last line segments of the closed polyline *PL*, l_1 and l_n , are adjacent to each other, i.e., $l_0 = l_n$, where *n* is the number of line segments in *PL*.
Postcondition: Smoothed polyline is stored in *PL*

```
(1) calculate the median normal, $\hat{\mathbf{N}}(l_i)$ , for each line segment, $l_i \in L$
    (1.1) initialize a real variable, $min\_sum_{angle}$ , to a big number
    (1.2) loop over each line segment: $l_j \in S_{neighbor} = \{l_k \mid k = i-2, i+2\}$
        (1.2.1) calculate the sum of angles, $sum_{angle}$ , between the normals of the current line segment, $l_j$ , and
                other line segments in $S_{neighbor}$
        (1.2.2) if $sum_{angle}$ is less than $min\_sum_{angle}$ , then $min\_sum_{angle} = sum_{angle}$
    (1.3) the median normal for $l_i$ , $\hat{\mathbf{N}}(l_i)$ , is equal to the normal of the line segment that corresponds to
          $min\_sum_{angle}$ in the loop of Step (1.2)
(2) calculate perturbation at each vertex $\upsilon \in V$
```

$$\mathbf{U}_\upsilon = 0.5((\mathbf{C}(l_{left}(\upsilon)) - \mathbf{P}(\upsilon)) \bullet \hat{\mathbf{N}}(l_{left}(\upsilon)))\, \hat{\mathbf{N}}(l_{left}(\upsilon) + (\mathbf{C}(l_{right}(\upsilon)) - \mathbf{P}(\upsilon)) \bullet \hat{\mathbf{N}}(l_{right}(\upsilon)))\, \hat{\mathbf{N}}(l_{right}(\upsilon)) , \qquad (3)$$

where $\mathbf{C}(l)$ and $\mathbf{P}(\upsilon)$ are the centroid of line segment $l \in L$ and the position of vertex $\upsilon \in V$, respectively. $l_{left}(\upsilon)$ and $l_{right}(\upsilon)$ are two line segments that are adjacent to vertex, υ .

Algorithm 2: mean_filter(PL)
Data Structures: L = set of all line segments; V = set of all vertices.
Precondition: *PL* contains the information of a polyline, including *L* and *V*.
Postcondition: Smoothed polyline is stored in *PL*

```
(1) calculate the mean normal, $\hat{\mathbf{N}}(l_i)$ , for each line segment, $l_i \in L$
    (1.1) let $\mathbf{N} = \sum_{j=i-2}^{j=i+2} \mathbf{N}(l_j)$
    (1.2)    let the normalized N be the mean normal for $l_i$, $\hat{\mathbf{N}}(l_i)$
(2) calculate perturbation at each vertex $\upsilon \in V$ by Equation (3)
```

The overall treatment for the sparing least-squares denoising is given in Algorithm 3, which reflects our philosophy of applying the expensive least-squares fitting only when it is absolutely necessary.

Algorithm 3: sparing_least-squares_denoising(PL)

Data Structures: L = set of all line segments; V = set of all vertices.
Precondition: *PL* contains the information of a polyline, including *L* and *V*.
Postcondition: Denoised polyline is stored in *PL*

```
(1) make a copy, $PL'$, of the input polyline $PL$
(2) execute fast_median_filter( $PL'$ )
(3) identification of sharp corners
    (3.1)    loop over each vertex $\upsilon \in V'$ on $PL'$
        (3.1.1)  if the angle between the normals of two line segments, which are incident to vertex $\upsilon$, is greater
                 than an angular threshold $\theta_t$, then mark the two neighboring line segments left to $\upsilon$ and the two
                 neighboring line segment right to $\upsilon$ as feature line segments.
(4) loop through each feature line segment and mark its two end nodes as feature vertices
(5) loop over each vertex $\upsilon \in V$ on $PL$ and determine VSI for each vertex
(6) main denoising loop with a specified number of denoising steps
    (6.1)    loop over each line segment $l \in L$
        (6.1.1)  if $l$ is not a feature line segment, calculate its mean normal by using Step (1) of Algorithm 2
        (6.1.2)  otherwise, calculate its median normal by using Step (1) of Algorithm 1
    (6.2)    loop over each vertex $\upsilon \in V$ on $PL$
        (6.2.1) if $\upsilon$ is not a feature vertex,
            (6.2.1.1) if VSI > $t_{VSI}$, use least-squares fitting to determine $\mathbf{U}_\upsilon$
            (6.2.1.2) otherwise, use mean normal to determine $\mathbf{U}_\upsilon$ as in Algorithm 2
        (6.2.2)  otherwise,
            (6.2.2.1)  if $l_{left}(\upsilon)$ is a feature line segment, $\hat{\mathbf{N}}(l_{left}(\upsilon))$ =median normal of $l_{left}(\upsilon)$; otherwise,
                       $\hat{\mathbf{N}}(l_{left}(\upsilon))$ = mean normal of $l_{left}(\upsilon)$.
            (6.2.2.2)  Similar operation is applied on $l_{right}(\upsilon)$ to calculate $\hat{\mathbf{N}}(l_{right}(\upsilon))$
            (6.2.2.3)  calculate the smoothing perturbation $\mathbf{U}_\upsilon$ by Equation (3)
        (6.2.3)  update the vertex coordinate by $\mathbf{P}_\upsilon = \mathbf{P}_\upsilon + \mathbf{U}_\upsilon$
```

Note that $\mathbf{P}_\upsilon$ in the above algorithm refers to the location vector of vertex υ. The number of denoising steps is usually specified by users. The conventional least-squares fitting treatment can be found in [6].

3. NUMERICAL EXPERIMENTS

Our sparing least-squares denoising was implemented in VC++ and tested on a HP Notebook PC with a 1.06 GHz Pentium III CPU and 504 MB of RAM. The time complexity of the approach is O(*n*), where n is the number of vertices of polylines. However, there is a large coefficient associated with the least-squares fitting, which is called in a frequency that is problem-dependent.

In order to quantify the denoising accuracy of two types of least-squares denoising algorithms, two error metrics are used in this paper. The first one is a vertex distance error metric, which measures the sum of distances between each vertex of a test polyline and a reference polyline. The second is an error metric of normal of line segments. It measures the sum of angles between each line segment of a test polyline and the corresponding line segment of a reference polyline. We represent the angles in radian in this paper.

In the first group of tests, we choose four different types of algebraic curves as a reference polyline, and then generate certain amount of random synthetic noises by using a random number generator in C++. In test case 1 of

Figure 4, the polyline consists of a part of $y = x^3$ on the right and a quarter of circle on the left, while test case 2 is a curve represented by $y^2 = x^3$. Rows 1 and 2 in Figure 4 represent these two curves and noised ones, respectively. The polylines denoised by the conventional and sparing least-squares fitting are given in rows 3 and 4 for a visual comparison. Table 2 shows a quantitative comparison between these two types of least-squares fitting in terms of execution time and error metrics in test cases 1 and 2. Here, the execution time is measured in milliseconds per denoising step.

A high-order drop shape, $\rho^5 + \left(\theta - \frac{\pi}{20}\right)^2 - 2\rho\left(\theta - \frac{\pi}{20}\right) = 0$, is used in test case 3, and the equation for the curve in test case 4 is $\left(x^2 + y^2\right)^2 = a^2\left(x^2 - y^2\right)$. Figure 4 provides a visual comparison of the denoising results between the two types of least-squares fitting, while Table 2 gives quantitative error metrics and execution time of two approaches in test cases 3 and 4. It can be easily seen from Figure 4 that the denoising results of two approaches are almost identical. This is also supported by the error metrics in Table 2.

Figure 5 is a more complex example of an airplane. To explore the effect of different noise levels on the performance of the two approaches, we designed two test cases, 5 and 6. As you can see from this figure and Table 3, the conventional and sparing least-squares approaches yield almost the same denoising result except a significant difference in computation time.

To investigate the effect of vertex density of a polyline on the execution time, we designed two test cases (7 and 8), as shown in Figure 6, on the basis of the drop shape in test case 3. From Table 3, an observable tendency is that the higher the vertex density is, the more execution time can be saved with the sparing least-squares treatment, compared to the conventional least-squares fitting.

One unexpected thing is that the normal error of our method is usually lower than that of least squares in Tables 2 and 3. One possible explanation is that the least squares fitting method is focused on the optimization of minimum distance deviation, while the mean filter is focused on the average surface normal, which may be beneficial to obtaining a lower normal error.

Figure 7 shows an application of the two approaches in denoising a boundary of a real point-cloud set. As you can see, all the major sharp corners were correctly preserved. In this case, we do not know the true underlying curve, and therefore it is impossible to calculate the error metrics.

We define time saving as a ratio of the execution time of conventional least-squares fitting to the execution time of sparing least-squares treatment. The average time saving for all the test cases in Figures 4 through 7 is 2.93 times, while the average differences between the two approaches in distance and normal error metrics are 12.7% and 9.3%, respectively. It is reasonable to infer that the sparing least-squares treatment gains a significant reduction in computation time and loses slightly in denoising accuracy, compared to the conventional least-squares denoising.

In the area of statistics, there are many studies related to bandwidth selection for locally weighted least-squares regression [13]. The selection problem could become a non-linear optimization, which is much more computationally expensive than a regular linear least-squares fitting. Since we have demonstrated the advantage of our new approach over the conventional linear least-squares smoothing, there is no point for us to conduct any further comparison with the bandwidth selection method.

4. CONCLUDING REMARKS

In this paper, we propose a new denoising algorithm in which the essential component is a sparing least-squares treatment. It significantly reduces the high computation cost associated with the conventional least-squares denoising approaches, while the high denoising accuracy is still maintained. The same concept should be easily extended to three-dimensional cases in the future.

5. ACKNOWLEDGEMENT

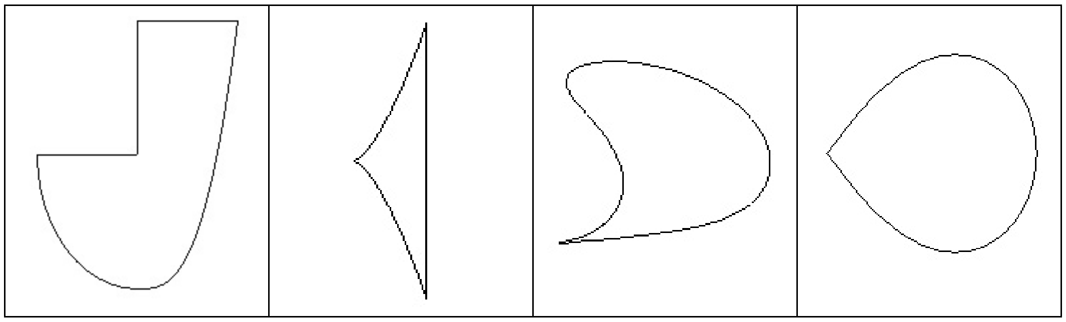
This paper was supported in part by USA NSF DMI 0514900. The comments from two anonymous reviewers are sincerely appreciated.

Test case	1	2	3	4
# of vertices	60	39	134	36
Denoising steps	100	200	200	200
Conventional LS				
Execution time (ms/step)	4.4	1.9	7.9	2.25
Distance error	2.49	0.86	42.24	1.23
Normal error (rad)	3.58	2.03	5.34	1.03
Sparing LS				
Execution time (ms/step)	1.6	1.85	1.35	1.8
Distance error	2.49	0.86	46.55	1.20
Normal error (rad)	3.15	2.03	5.21	1.28

Tab. 2: Comparison between conventional and sparing least-squares denoising.

Test case	5	6	7	8
# of vertices	192	192	67	34
Denoising steps	50	50	200	200
Conventional LS				
Execution time (ms/step)	13.6	10.8	3.05	1.75
Distance error	58.04	78.18	58.69	85.0
Normal error (rad)	15.97	18.12	2.75	2.85
Sparing LS				
Execution time (ms/step)	3.4	3.2	0.675	0.29
Distance error	71.84	81.57	82.09	102.7
Normal error (rad)	15.33	16.29	2.39	3.14

Tab. 3: Comparison between conventional and sparing least-squares denoising.



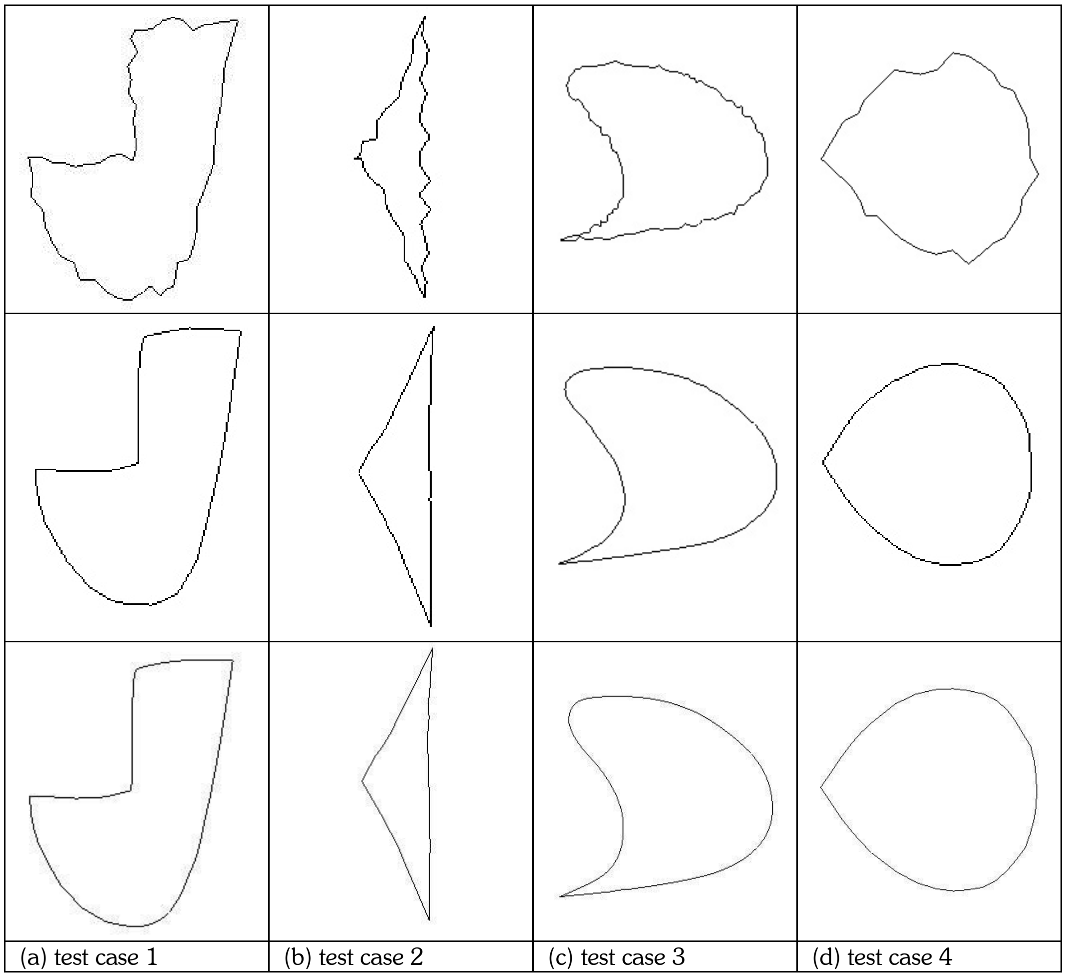
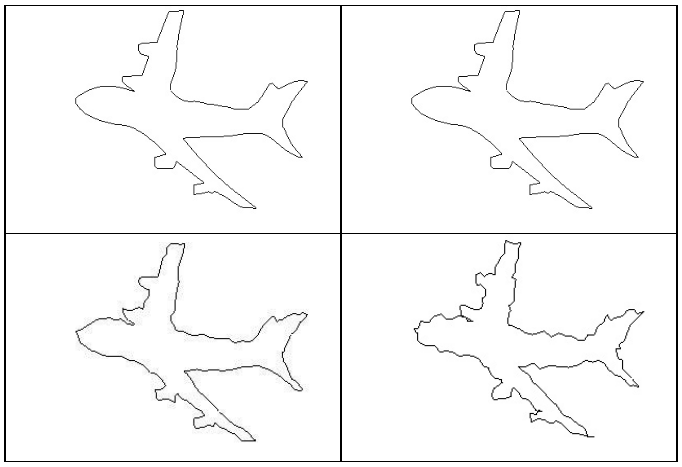


Fig. 4: Denoising of noised polylines whose underlying shape is an algebraic curve or its combination (row 1: original polyline; row 2: noised polyline; row 3: denoised polyline by conventional least-squares fitting; row 4: denoised polyline by sparing least-squares treatment).



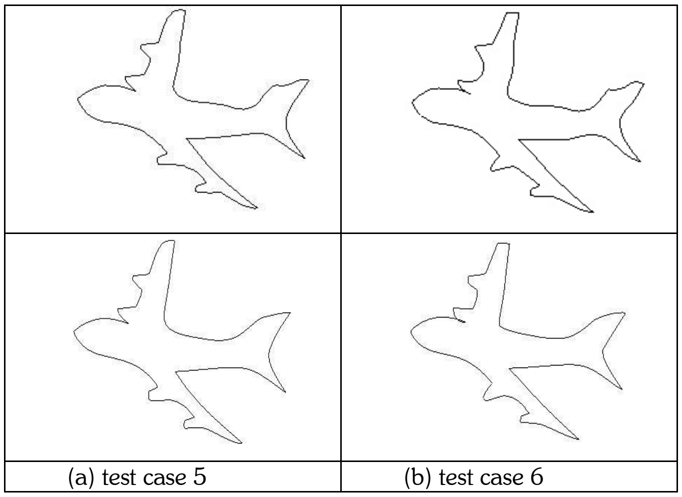


Fig. 5: Influence of noise levels on the denoising results (row 1: original polyline; row 2: noised polyline; row 3: denoised polyline by conventional least-squares fitting; row 4: denoised polyline by sparing least-squares treatment).

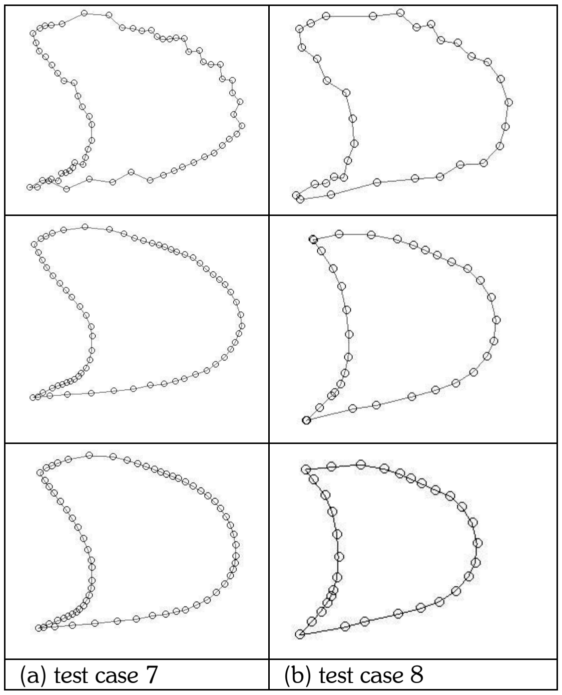


Fig. 6: Influence of vertex density on the denoising results (row 1: noised polyline; row 2: denoised polyline by conventional least-squares fitting; row 3: denoised polyline by sparing least-squares treatment).

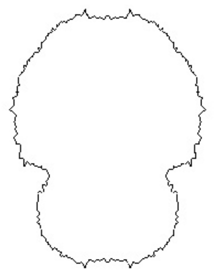
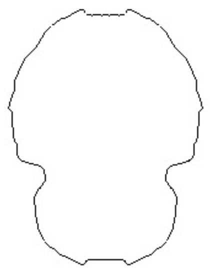
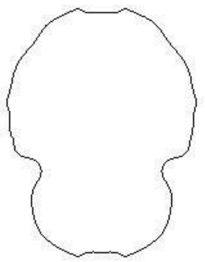
		
(a) input curve	(b) conventional least-squares denoising	(c) Sparing least-squares denoising

Fig. 7: Test case 9 -- smoothing of a noised boundary obtained from the edge detection of a point-cloud data set (# of vertices = 342; # of smoothing steps = 100; time saving = 4.96 times).

6. REFERENCES

[1] Belyaev, A.; Ohtake, Y.: A comparison of mesh smoothing methods, Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics, 2003, 83-87.

[2] Clarenz, U.; Diewald, U.; Rumpf, M.: Anisotropic geometric diffusion in surface processing, Proceedings of IEEE Visualization, 2000, 397-405.

[3] Desbrun, M.; Meyer, M.; Schroder, P.; Barr, A. H.: Anisotropic feature-preserving denoising of height fields and bivariate data, Graphics Interface, 2000, 145-152.

[4] Fleishman, S.; Cohen-Or, D.; Silva, C.: Robust moving least-squares fitting with sharp features, ACM Transactions on Graphics., 24(3), 2005, 544-552.

[5] Shen, Y.; Barner, K. E.: Fuzzy vector median-based surface smoothing, IEEE Transactions on Visualization and Computer Graphics, 10(3), 2004, 266-277.

[6] Shen, J.; Maxim, B.; Akingbehin, K.: Accurate Correction of Surface Noises of Polygonal Meshes, International Journal for Numerical Methods in Engineering, 64(12), 2005, 1678-1698.

[7] Mangan, A. P.; Whitaker, R. T.: Partitioning 3D surface meshes using watershed segmentation, IEEE Transactions on Visulization and Computer Graphics, 5(4), 1999, 308-321.

[8] Leonardis, A.; Gupta, A.; Bajcsy, R.: Segmentation of range images as the search for geometric parametric models, International Journal of Computer Vision, 14, 1995, 253-277.

[9] Varady, T.; Benko, P.; Kos, G.: Reverse engineering regular objects: simple segmentation and surface fitting procedures, International Journal of Shape Modeling, 4(3-4), 1998, 127-141.

[10] Schall, O.; Belyaev, A.; Seidel, H.: Robust filtering of noisy scattered point data, Eurographics Symposium on Point-based Graphics, 2005.

[11] Xie, H.; McDonnell, K. T.; Qin, H.: Surface reconstruction of noisy and defective data sets, IEEE Visualization 2004, 2004, 259-266.

[12] Kolluri, R.; Shewchuk, J. R.; O'Brien, J. F.: Spectral surface reconstruction from noisy point clouds, Symposium on Geometry Processing, 2004, 11-21.

[13] Fan, J.; Gijbels, I.: Data-driven bandwidth selection in local polynomial fitting: variable bandwidth and spatial adaptation, Journal of the Royal Statistical Society, Series B, 57(2), 1995, 371-394.