A Set of Denoising Algorithms for 2-D Closed Curves

Jie Shen[1], David Yoon[1], Huahao Shou[2] and Dong-Ming Zhao[1]

[1]University of Michigan - Dearborn, {shen|dhyoon|dmzhao}@umich.edu
[2]Zhejiang University of Technology, shh@zjut.edu.cn

ABSTRACT

Accurate reduction of two-dimensional curve noises is crucial to metrology and reverse engineering. In this paper we propose a set of schemes for two-dimensional curve denoising. Our approach consists of three main steps: 1) feature-based pre-smoothing of noised polylines; 2) curve partitioning of polylines into different regions; 3) two hybrid denoising schemes of arbitrary polylines with noises. Numerical experiments indicate the effectiveness of our approach against existing methods.

Keywords: Metrology, Denoise, Reverse Engineering, Curves.

1. INTRODUCTION

In the area of metrology and reverse engineering, the problem of measurement or object reconstruction is often transformed to an analysis of two-dimensional closed curves, which correspond to the silhouette or cross-section of objects. Due to the limitation of various sensors and edge detection algorithms, the resulting curves commonly contain certain amount of noises. How to eliminate these noises is an important issue with respect to improving the accuracy for measurement and object reconstruction.

Although a considerable number of studies have been conducted in the past [1-4,7], the denoising accuracy of existing methods is still not good enough for applications in metrology and reserve engineering. The objective of this study is to propose a new set of denoising schemes that have a better denoising accuracy and is suited to arbitrary two-dimensional curves with or without sharp corners.

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

2. A SET OF NEW CURVE DENOISING ALGORITHMS

Our new approach consists of three main components: 1) a polyline denoising scheme with sharp corners preserved; 2) a high-accuracy denoising scheme for smooth curved polylines; 3) a hybrid approach for arbitrary polylines with or without sharp corners.

2.1 A Polyline Denoising Scheme with Sharp Corners Preserved

Median filter is one of best classic filters in handling noises at sharp corners. When we apply it to polylines, the basic concept is to calculate a median normal for each line segment first, and then to calculate smoothing perturbation for each vertex on the basis of the following Equation (1), as illustrated in Figure 1. The median normal of a line segment is obtained by a voting among the normals of all nearby line segments with respect to the current line segment. The voting selects a normal that has a minimum sum of the angles with other normals, where a normal of a two-dimensional line segment is defined by a unique direction that is perpendicular to the line segment, points to the outside of the area enclosed by the polyline, and lies in the two-dimensional plane in which the polyline is located.

The median filtering procedure is as follows.

Algorithm 1: 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: $l_j \in S_{neighbor} = \{l_k \mid k = i-2,\ldots,i+2\}$

(1.2.1) calculate the sum of angles, sum_{angle}, between the normals of the current line segment 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 $\upsilon \in V$

$$\mathbf{U}_{\upsilon} = \frac{((\mathbf{C}(l_{left}(\upsilon)) - \mathbf{P}(\upsilon)) \bullet \hat{\mathbf{N}}(l_{left}(\upsilon)))\,\hat{\mathbf{N}}(l_{left}(\upsilon)) + ((\mathbf{C}(l_{right}(\upsilon)) - \mathbf{P}(\upsilon)) \bullet \hat{\mathbf{N}}(l_{right}(\upsilon)))\,\hat{\mathbf{N}}(l_{right}(\upsilon))}{2} \quad (1)$$

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 from the left and right, respectively.

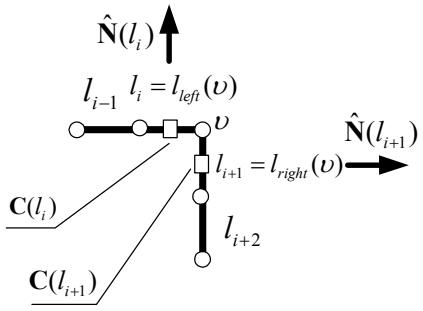


Fig. 1. Median filter for a polyline.

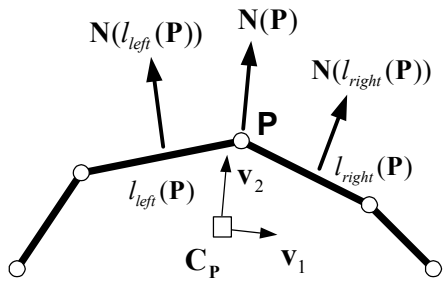


Fig. 2. Local coordinate of vertex **P**.

2.2 A High-accuracy Denoising Scheme for Curved Polylines without Sharp Corners

Mean filter is one of the commonly-used classic filters for smoothing polylines without sharp corners. Gaussian filter or its variants can be viewed as a weighted mean filter. In the context of polylines, a mean normal is calculated for each line segment, and then a smoothing perturbation for each vertex is determined. The mean normal refers to a normalized vector obtained from the vector summation of normals of all the nearby line segments.

We apply the mean filter to smooth a noised polyline by the following procedure.

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 $v \in V$ by using Equation (1) □

Most symbols in Algorithm 2 share the same meaning as those in Algorithm 1 except for $\hat{\mathbf{N}}(l_i)$. Essentially, the mean filter is a linear filter, which will cause noticeable linear approximation errors when a second-order or even higher-order curve with a sparse vertex density is processed. To overcome this problem, we propose the following high-accuracy algorithm for smoothing curved polylines.

First of all, the normal and tangential directions at each vertex $\mathbf{P}$ of a polyline is estimated by the principal component method [5]. The covariance matrix of the set of neighboring vertices is

$$\mathbf{CV} = \sum_{\mathbf{q} \in Nbhd(\mathbf{P})} (\mathbf{q} - \mathbf{C_P}) \otimes (\mathbf{q} - \mathbf{C_P}) \tag{2}$$

where $Nbhd(\mathbf{P}) = \{v_j \mid j = i-2, i+2\}$ is the set of neighboring vertices at vertex $\mathbf{P}$, and $\mathbf{P}$ corresponds to vertex v_i in this case. $\mathbf{C_P}$ is the centroid of $Nbhd(\mathbf{P})$, and $\otimes$ is outer product operator of vectors. A Jacobi transformation [6] can be used to determine eigenvectors $(\mathbf{v}_1, \mathbf{v}_2)$ and eigenvalues $(\lambda_1 \geq \lambda_2)$ of the CV. $\mathbf{V}_1$ and $\mathbf{V}_2$ represent the tangential and normal directions, respectively. A local coordinate system ($\mathbf{v}_1$, $\mathbf{v}_2$) can be established by letting $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively in the local *x* and *y* directions, with the centroid of $Nbhd(\mathbf{P})$ as its origin, as in Figure 2.

After knowing the local coordinate system at vertex $\mathbf{P}$, we are ready to conduct a least-squares fitting. A second-order local curve patch is used in Algorithm 3, and an accurate calculation of nearest point on the fitted local curve patch is carried out for each vertex on the polyline. Note that a so-called "curve patch" in this paper does not refer to a surface area, but means a curve region instead.

Algorithm 3: Robust Least-squares Fitting of a Second-order Local Curve Patch

(1) Loop over every vertex $\mathbf{P}$ for a second-order fitting

(1.1) Determine a local quadric curve patch.

In the local coordinate system $(\mathbf{v}_1, \mathbf{v}_2)$, a quadric curve patch:

$$y = f(x) = a_1 x^2 + a_2 x + a_3 \tag{3}$$

is used to approximate the curve in the neighborhood of vertex $\mathbf{P}$. Here, coordinate *x* is measured along $\mathbf{v}_1$ direction, and *y* coordinate is measured in the $\mathbf{v}_2$ direction. The linear least-squares estimation of three coefficients a_i is expressed as

$$\mathbf{BA} = \mathbf{Z} \tag{4a}$$

in which

$$\mathbf{B} = \begin{bmatrix} x_1^2 & x_1 & 1.0 \\ x_2^2 & x_2 & 1.0 \\ \cdots & & \\ x_n^2 & x_n & 1.0 \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \tag{4b}$$

where *n* is the number vertices of $Nbhd(\mathbf{P})$, which refers to the two-ring neighborhood of vertex $\mathbf{P}$. If we consider $\mathbf{P} \in Nbhd(\mathbf{P})$, *n* equals 5 in the case of a polyline. If we solve Equation (4a) by using $\mathbf{A} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{Z}$, $\mathbf{B}^T\mathbf{B}$ may be ill-conditioned. Thus, singular value decomposition [6] is used to solve $\mathbf{BA} = \mathbf{Z}$. □

Our high-accuracy denoising scheme for a smooth curved polyline is illustrated in Figure 3, in which $\mathbf{P}$ is the current vertex and $\mathbf{Q}_1$ through $\mathbf{Q}_4$ are its 2-ring neighboring vertices. $\mathbf{V}_0$ through $\mathbf{V}_4$ are correction vectors for the second-order smoothing, and are determined by a vector from each vertex to its closest point on the local quadric curve patch. The closest point can be located in a process for finding the distance from each vertex to the local quadric curve patch

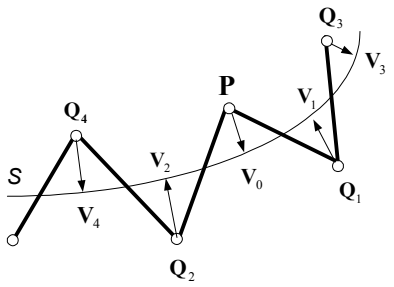


Fig. 3. Least-squares fitting of a quadric or cubic curve patch.

For the local quadric curve patch, the closest point is determined by a root finding as follows. We rewrite the local quadric curve patch in Equation (3) by a general quadratic equation

$$S(\mathbf{p}) = \mathbf{p}^T \mathbf{A}\,\mathbf{p} + \mathbf{b}^T \mathbf{p} + c = 0 \tag{5a}$$

where $\mathbf{p}^T = [x \quad y]$ represents points on the curve $S(\mathbf{p})$. $\mathbf{A}$, $\mathbf{b}$ and c are represented by

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} a_2 \\ -1 \end{bmatrix} \quad \text{and} \quad c = a_3 . \tag{5b}$$

For a given vertex $\mathbf{q} = [x_q \quad y_q]^T$ of a polyline, its closest point $\mathbf{p}$ on the line S can be determined by the following geometric relationship

$$\mathbf{q} - \mathbf{p} = t\,\nabla S(\mathbf{p}) = t(2\mathbf{A}\mathbf{p} + \mathbf{b}) \tag{6}$$

where t is a scalar. $\nabla S(\mathbf{p})$ is the gradient of the curve, which is in the direction of curve normal and should be the same as the direction of $\mathbf{q} - \mathbf{p}$. The rearrangement of Equation (6) yields

$$\mathbf{p} = (\mathbf{I} + 2t\,\mathbf{A})^{-1}(\mathbf{q} - t\mathbf{b}) = \begin{bmatrix} \dfrac{x_q - a_2 t}{2a_1 t + 1} \\ y_q + t \end{bmatrix} \tag{7}$$

where $\mathbf{I}$ is an identity matrix. The substitution of Equation (7) into Equation (5a) will get a polynomial equation of degree 3 with respect to t. Once a suitable root of t is found, the closest point $\mathbf{p}$ is then determined by Equation (7).

After the nearest points on the curve are known, our high-accuracy denoising scheme for a smooth curved polyline is determined by the following procedure.

Algorithm 4: Determination of Correction at Each Vertex of a Polyline
Precondition: An integer counter $count(\mathbf{Q}) = 0$ for each vertex $\mathbf{Q}$ of a polyline
Postcondition: a smoothing perturbation, $\mathbf{U_Q}$, at each vertex $\mathbf{Q}$ has been calculated
(1) loop over every vertex $\mathbf{Q}$ of a polyline, which is not in a sharp corner region
  (1.1) loop over 2-ring vertices $\mathbf{q} \in Nbhd(\mathbf{Q})$ (from v_{i-2} to v_{i+2})
    (1.1.1) calculate the nearest point $\mathbf{p}$ via Equation (7) for a quadric curve fitting
    (1.1.2) calculate the nodal normal at vertex $\mathbf{q}$, $\tilde{\mathbf{N}}_\mathbf{q}$
      (1.1.2.1) if $l_{left}(\mathbf{q})$ is a feature line segment, $\hat{\mathbf{N}}(l_{left}(\mathbf{q}))$ =median normal of $l_{left}(\mathbf{q})$; otherwise, $\hat{\mathbf{N}}(l_{left}(\mathbf{q}))$ =mean normal of $l_{left}(\mathbf{q})$.
      (1.1.2.2) Similar operation is applied on $l_{right}(\mathbf{q})$ to calculate $\hat{\mathbf{N}}(l_{right}(\mathbf{q}))$
      (1.1.2.3) $\tilde{\mathbf{N}}_\mathbf{q}$ = normalization of $\hat{\mathbf{N}}(l_{left}(\mathbf{q})) + \hat{\mathbf{N}}(l_{right}(\mathbf{q}))$
    (1.1.3) Accumulate a correction to the overall perturbation at vertex $\mathbf{q}$

$$\mathbf{U_q} \leftarrow \mathbf{U_q} + ((\mathbf{p} - \mathbf{q}) \bullet \tilde{\mathbf{N}}_q)\tilde{\mathbf{N}}_q, \quad count(\mathbf{q}) \leftarrow count(\mathbf{q}) + 1, \tag{8}$$

  where $\mathbf{U_q}$ is the overall correction at vertex $\mathbf{q}$, and $\bullet$ is a dot product.

(2) loop over every vertex $\mathbf{Q}$ that is not in a sharp corner region of the polyline

$$\mathbf{U_Q} \leftarrow \mathbf{U_Q} / count(\mathbf{Q}) \quad . \tag{9}$$

□

Note that the averaging operation in Equation (9) is a crucial step for the stability of Algorithm 4, and $count()$ ensures the partition of unity. $\mathbf{U_Q}$ is used to update the position of each vertex that is not in a sharp corner region at the current iteration.

2.3 A Hybrid Approach to Denoise Arbitrary Polylines

Geometric continuity and discontinuity are two extreme poles, which should not be handled by a same procedure from a philosophical viewpoint. To smooth arbitrary polylines that contain both geometric continuity and discontinuity, we designed two hybrid schemes: mean-median and quadric-median. The former is the combination of mean and median filters, while the latter is to blend our quadric fitting with the median filter.

Algorithm 5: mean_median(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) make a copy, PL', of the input polyline PL
(2) execute *median_filter*(PL')
(3) identification of sharp corners
  (3.1) loop over each vertex $v \in V'$ on PL'
    (3.1.1) if the angle between the normals of two line segments, which are incident to vertex v, is greater than an angular threshold θ_t, then mark the two neighboring line segments left to v and the two neighboring line segment right to v as feature line segments.
(4) main smoothing loop with a specified number of smoothing steps
  (4.1) loop over each line segment $l \in L$
    (4.1.1) if l is not a feature line segment, calculate its mean normal by using Step (1) of Algorithm 2
    (4.1.2) otherwise, calculate its median normal by using Step (1) of Algorithm 1
  (4.2) loop over each vertex $v \in V$ on PL
    (4.2.1) if $l_{left}(v)$ is a feature line segment, $\hat{\mathbf{N}}(l_{left}(v))$ =median normal of $l_{left}(v)$; otherwise, $\hat{\mathbf{N}}(l_{left}(v))$ = mean normal of $l_{left}(v)$.
    (4.2.2) Similar operation is applied on $l_{right}(v)$ to calculate $\hat{\mathbf{N}}(l_{right}(v))$
    (4.2.3) calculate the smoothing perturbation $\mathbf{U}_v$ by Equation (1)
    (4.2.4) update the vertex coordinate by $\mathbf{P}_v = \mathbf{P}_v + \mathbf{U}_v$ □

Algorithm 6: quadric_median(PL)
(1) through (3) are the same as in Algorithm 5
(4) loop through each feature line segment and mark its two end nodes as feature vertices
(5) main smoothing loop with a specified number of smoothing steps
  (5.1) loop over each line segment $l \in L$
    (5.1.1) if l is not a feature line segment, calculate its mean normal by using Step (1) of Algorithm 2
    (5.1.2) otherwise, calculate its median normal by using Step (1) of Algorithm 1
  (5.2) loop over each vertex $v \in V$ on PL
    (5.2.1) if v is not a feature vertex, use Algorithm 4 to determine $\mathbf{U}_v$

(5.2.2) otherwise, use Steps (4.2.1), (4.2.2) and (4.2.3) in Algorithm 5 to determine $\mathbf{U}_\upsilon$

(5.2.3) update the vertex coordinate by $\mathbf{P}_\upsilon = \mathbf{P}_\upsilon + \mathbf{U}_\upsilon$ □

Note that $\mathbf{P}_\upsilon$ in Algorithms 5 and 6 refers to the location vector of vertex υ. The number of smooth steps is usually specified by users.

3. NUMERICAL EXPERIMENTS

The proposed set of algorithms 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 all four different smoothing algorithms (1, 2, 5 and 6) is O(*n*), where *n* is the number of vertices of polylines, even though Algorithm 6 has a larger coefficient.

In order to quantify the denoising accuracy of different 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++. These two groups of polylines are shown in rows 1 and 2 of Figure 4, respectively. The noised polylines are processed by the proposed set of denoising algorithms and illustrated in rows 3 through 6 of Figure 4. The calculated error metrics and the number of smoothing steps are given in Table 1.

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. Test case 2 is a curve represented by $y^2 = x^3$, while 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. 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 the denoising results of five different algorithms in a visual format, while Table 1 gives quantitive error metrics of algorithms in each test case in Figure 4. Note that the anisotropic diffusion was implemented by a bilateral filter, which was not convergent and thus terminated at step 3 for a best possible denoising performance. It can be easily seen from Figure 4 that our two hybrid algorithms provided the best denoising results. This is also supported by the error metrics in Table 1. The smaller the error metrics, the better the denoising accuracy. The only noticable difference between Algorithms 5 and 6 is that the latter takes much longer time to complete.

Figure 5 is a more complex example of an airplane. To explore the effect of different noise levels on the performance of our two hybrid algorithms, we designed three test cases. Test case 5 of Figure 5 is an original polyline without any noise, while two different levels of noises were added in test cases 6 and 7. Row 1 of Figure 5 consists of the original polylines with or without noises added. If you compare the denoising results of our two hybrid algorihtms (rows 2 and 3, respectively), you will find out that the quadric-median scheme preserved the tail and body parts better. In Table 1, the error metrics of the mean-median scheme were slightly worse than those of the quadric-median scheme, but the former had a much shorter computation time. The problem with the method in [4] is that no feature preserving presmoothing was used so that the chicken-and-egg problem between feature recognition and presmoothing is not well solved.

To investigate the effect of vertex density of a polyline on the denoising accuracy, some tests were conducted. Due to the page limitation, we don't present detailed data here. An observed tendency is that the lower the vertex density is, the better denoising accuracy the quadric-median scheme has than the mean-median scheme, if the polyline is a high-order curve.

4. CONCLUDING REMARKS

In this paper, we propose a set of new schemes for denoising arbitrary two-dimensional closed polylines with or without sharp corners. If the vertex density of a polyline is high, it is recommended to use our mean-median scheme. On the other hand, if the vertex density is low, our quadric-median scheme should be adopted. Numerical experiments were conducted for a comparison with conventional mean and median filters as well as anisotropic diffusion, and the results indicate the effectiveness of our hybrid schemes in handling arbitrary polylines that contain both geometric continuity and discontinuity. This is particularly useful in handling real complex objects.

5. ACKNOWLEDGEMENTS

This paper was supported in part by USA NSF DMI 0514900, a REEDF grant from the State of Michigan, and a Rackham grant from the University of Michigan.

6. REFERENCES

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

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

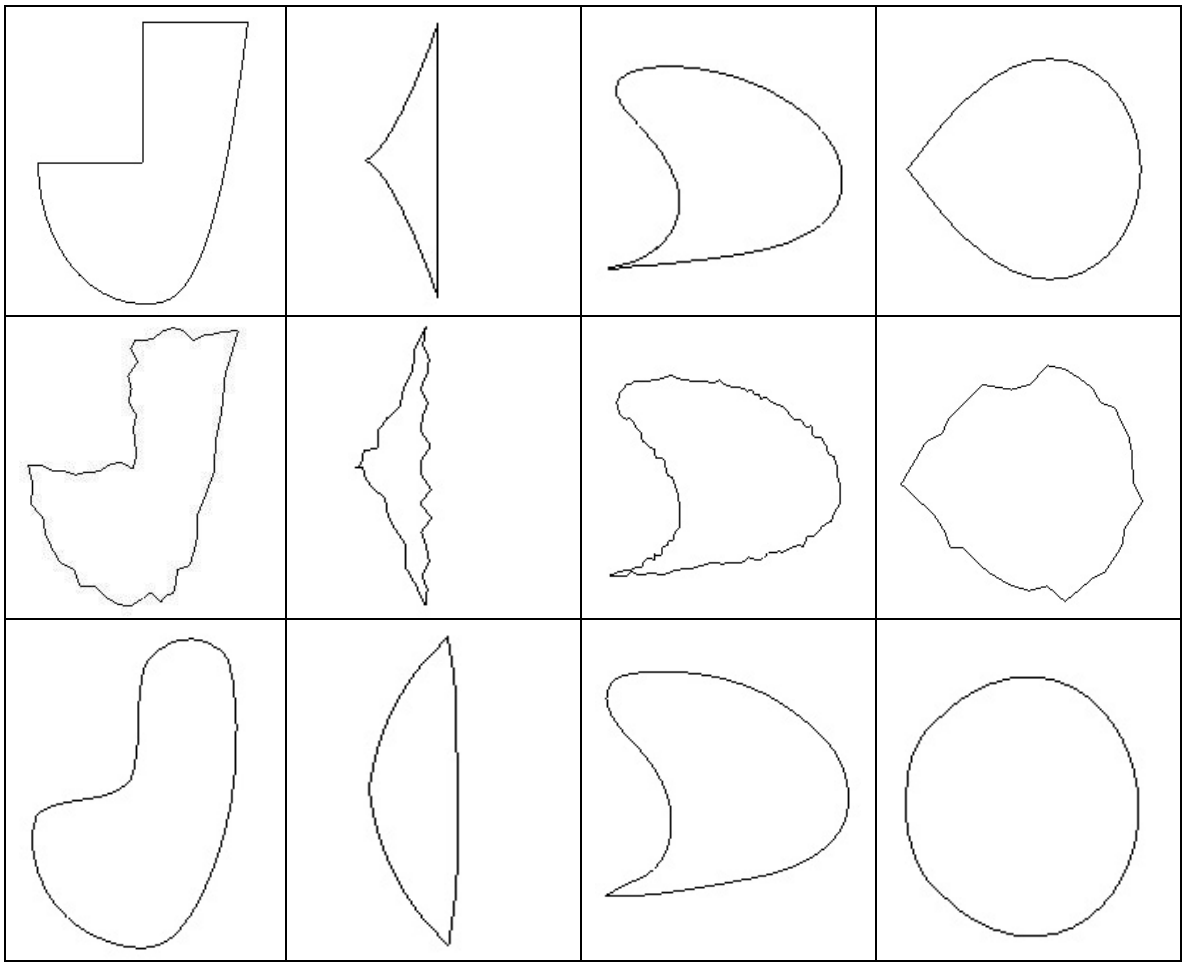
[3] Desbrun, M., Meyer, M., Schroder, P. and Barr, A. H., Anisotropic feature-preserving denoising of height fields and bivariate data, *Graphics Interface,* 2000, pp 145-152.

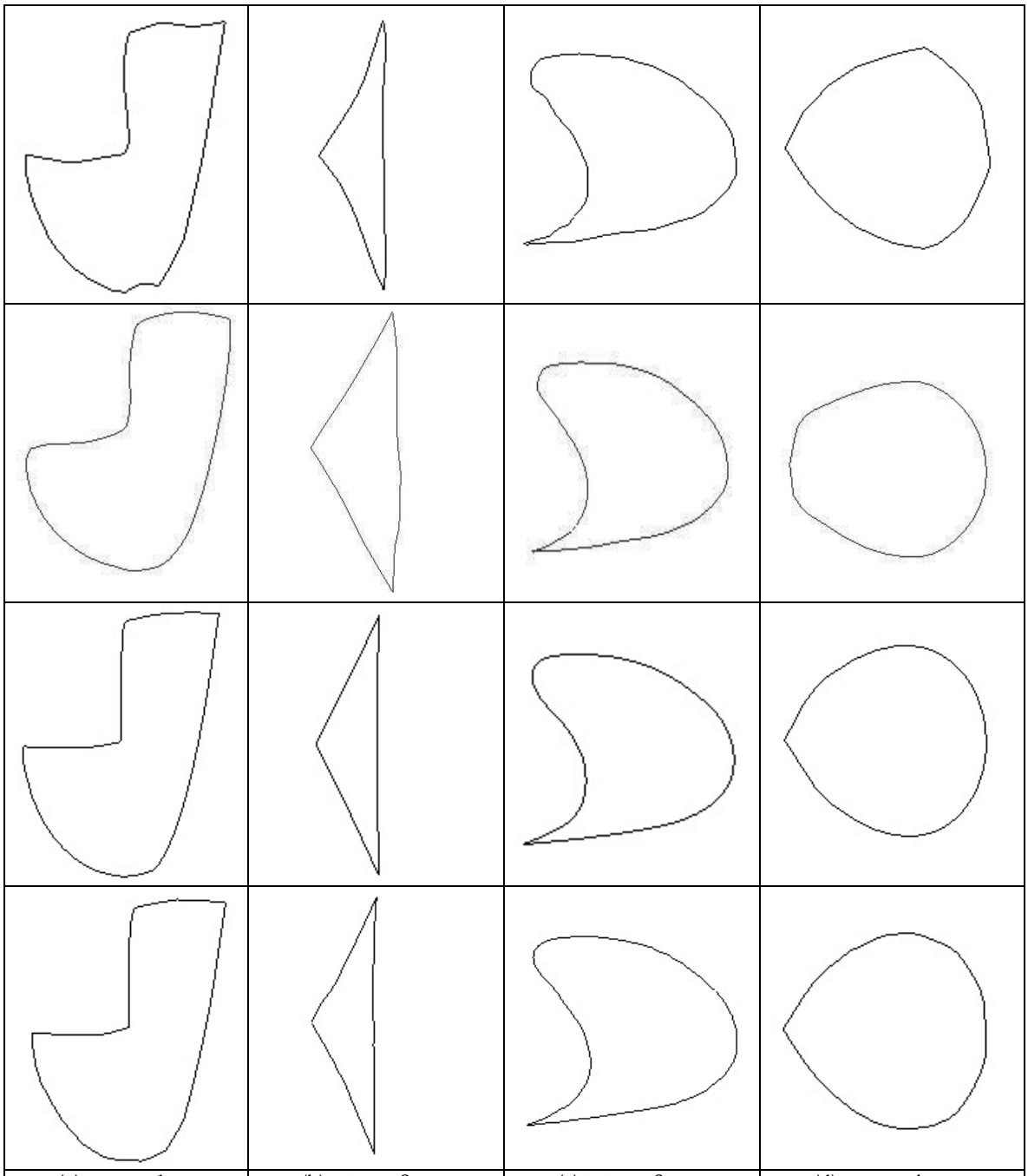
[4] Fleishman, S., Cohen-Or, D. and Silva, C., Robust moving least-squares fitting with sharp features, *ACM Transactions on Graphics,* Vol. 24, No. 3 2005, pp 544-552.

[5] Hoppe, H., De Rose, T., Duchamp, T., MaDonald, J. and Stuetzle, W., Mesh Optimization, *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, Vol. 27, 1993, pp 19-26.

[6] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P., *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press: Cambridge,1992, pp 994.

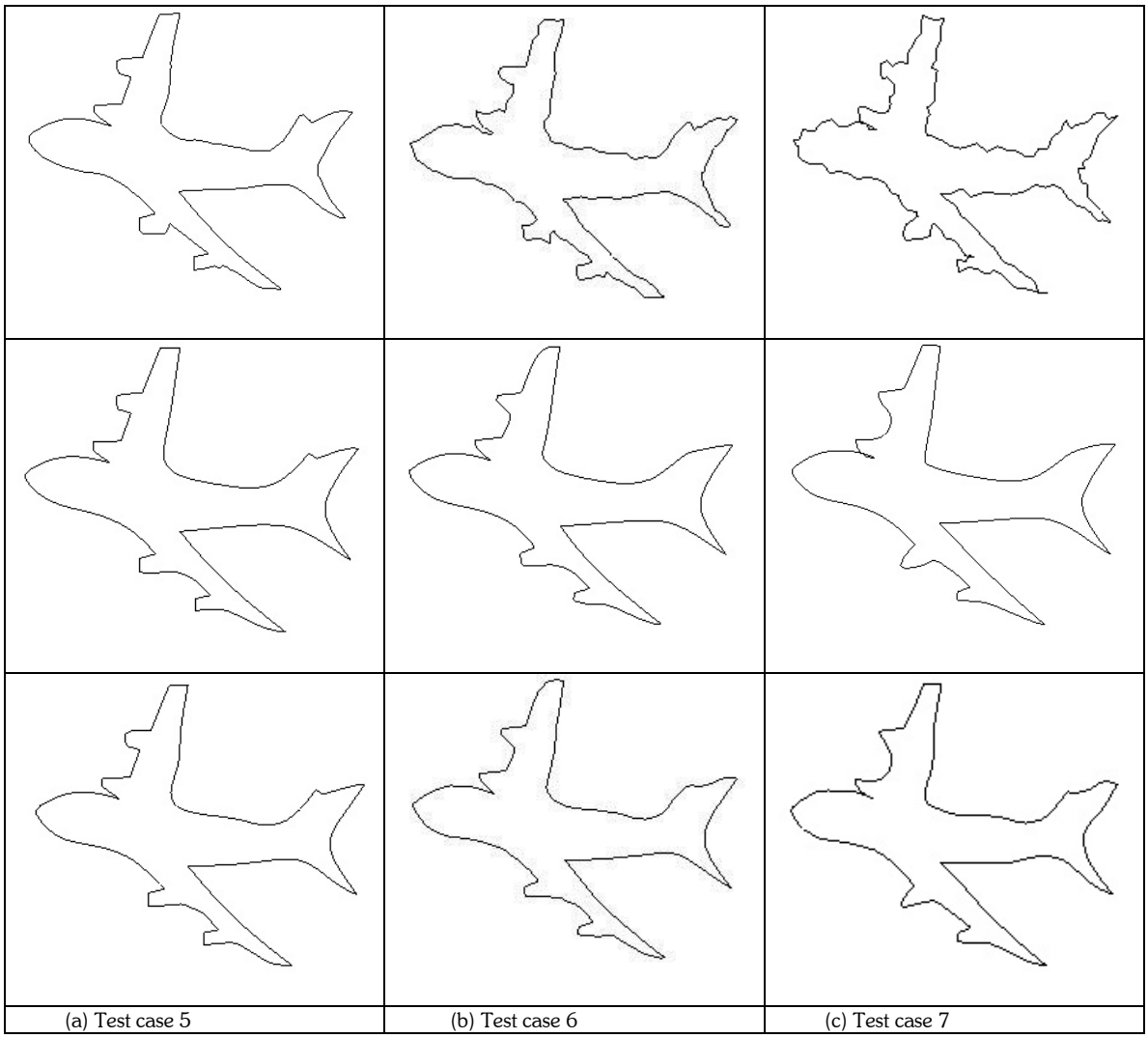
[7] Shen, Y., Barner, K. E., Fuzzy vector median-based surface smoothing, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 10, No. 3, 2004, pp 266-277.





(a) test case 1	(b) test case 2	(c) test case 3	(d) test case 4

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 mean filter; row 4: denoised polyline by median filter; row 5: denoised polyline by anisotropic diffusion; row 6: denoised polyline by mean-median; row 7: denoised polyline by quadric-median).



(a) Test case 5 (b) Test case 6 (c) Test case 7

Fig. 5. Influence of noise levels on the smoothing results of our schemes (row 1 – input polylines with no noise in column 1, level 1 of noise in column 2, and level 2 of noise in column 3; row 2 – smoothing result of mean-median; row 3 – smoothing result of quadric-median.)

Tab. I. Denoising result of different test polylines.

Test Case	# of vertices	Smoothing Step	Execution Time (sec)	Distance Error Metric	Normal Error Metric (rad)
1	60	100 100 3 100 100	0.02 (MN*) 0.09 (MD) 0.0006 (AD) 0.09 (NM) 0.44 (QM)	7.12 (MN) 3.04 (MD) 7.69 (AD) 1.96 (NM) 2.49 (QM)	13.21 (MN) 6.28 (MD) 7.98 (AD) 2.77 (NM) 3.58 (QM)
2	39	200 200 3 200 200	0.02 (MN) 0.11 (MD) 0.0006 (AD) 0.08 (NM) 0.38 (QM)	3.97 (MN) 0.82 (MD) 1.32 (AD) 1.10 (NM) 0.86 (QM)	9.78 (MN) 1.72 (MD) 4.43 (AD) 1.67 (NM) 1.56 (QM)

3	134	200 200 3 200 200	0.07 (MN) 0.39 (MD) 0.002 (AD) 0.18 (NM) 1.58 (QM)	48.92 (MN) 52.35 (MD) 111.43 (AD) 43.02 (NM) 42.24 (QM)	7.17 (MN) 7.08 (MD) 6.26 (AD) 5.11 (NM) 5.34 (QM)
4	36	200 200 3 200 200	0.02 (MN) 0.12 (MD) 0.0006 (AD) 0.06 (NM) 0.45 (QM)	2.32 (MN) 1.67 (MD) 8.63 (AD) 1.56 (NM) 1.23 (QM)	3.56 (MN) 1.41 (MD) 2.95 (AD) 1.04 (NM) 1.03 (QM)
5	192	50	0.21 (NM) 0.57 (QM)	49.81 (NM) 34.26 (QM)	8.02 (NM) 7.43 (QM)
6	192	50	0.22 (NM) 0.68 (QM)	70.44 (NM) 58.04 (QM)	15.63 (NM) 15.97 (QM)
7	192	50	0.21 (NM) 0.54 (QM)	79.91 (NM) 78.18 (QM)	17.30 (NM) 18.12 (QM)

* MN – mean; MD – median; AD – anisotropic diffusion; NM – mean-median; QM – quadric-median.