Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2010, Article ID 154891, 13 pages
doi:10.1155/2010/154891

Research Article

Two-Stage Outlier Elimination for Robust Curve and Surface Fitting

Jieqi Yu, Haipeng Zheng, Sanjeev R. Kulkarni, and H. Vincent Poor

Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA

Correspondence should be addressed to Jieqi Yu, jieqiyu@princeton.edu

Received 1 January 2010; Revised 25 April 2010; Accepted 7 June 2010

Academic Editor: Ling Shao

An outlier elimination algorithm for curve/surface fitting is proposed. This two-stage hybrid algorithm employs a proximity-based outlier detection algorithm, followed by a model-based one. First, a proximity graph is generated. Depending on the use of a hard/soft threshold of the connectivity of observations, two algorithms are developed, one graph-component-based and the other eigenspace-based. Second, a model-based algorithm, taking the classification of inliers/outliers of the first stage as its initial state, iteratively refits and retests the observations with respect to the curve/surface model until convergence. These two stages compensate for each other so that outliers of various types can be eliminated with a reasonable amount of computation. Compared to other algorithms, this hybrid algorithm considerably improves the robustness of ellipse/ellipsoid fitting for scenarios with large portions of outliers and high levels of inlier noise, as demonstrated by extensive simulations.

1. Introduction

Curve and surface fitting has a broad range of applications. For example, in computer vision, curves and surfaces are important geometric primitives and shape descriptors. As a result, curve and surface fitting is commonly carried out in applications such as object reconstruction and feature extraction. In related fields such as image processing, pattern recognition, and machine learning, it is consequently an essential tool.

As an extensively studied field, curve and surface fitting has a rich literature. Various algorithms have been proposed from several different perspectives. Most of the algorithms are based on minimizing an l_2 norm, which results in least-squares fitting methods. There are two main categories of least-squares fitting algorithms, algebraic fitting [1–3], and geometric fitting [4–6], depending on the definition of error distances.

Least-squares methods are efficient and accurate in situations in which observations are accurate or are contaminated by a moderate amount of noise. However, when the noise level is high or when the set of observations contains wildly erroneous observations, which is often the case when dealing with real environments, least-squares algorithms break down, and more robust algorithms are required to accomplish the task.

Under those circumstances, l_1 distance is often employed instead of Euclidean distance to tackle the outliers. There is a considerable literature focusing on this case as well. Al-Subaihi and Watson [7] proposed a series of curve fitting algorithms using the l_1 norm and a Gauss-Newton step, which includes examples of fitting lines in three-dimensional space, as well as fitting circles and ellipses in two-dimensional space. Al-Subaihi [8] further investigated an algorithm that fits circular arcs to observations using the l_1 norm. As shown by numerical examples, the algorithm can produce satisfying results with the presence of a small portion of extreme outliers. Although fitting curves and surfaces using the l_1 norm is an effective way of dealing with outliers, it still requires a concrete model for the curve and surface to be fitted. In particular, a specific algorithm has to be designed for each model based on its mathematical form, typically yielding high computational cost. As a result, it is still desirable to remove the majority of the outliers from the observations first without the knowledge of the specific model, and then to fit the observations to the model using

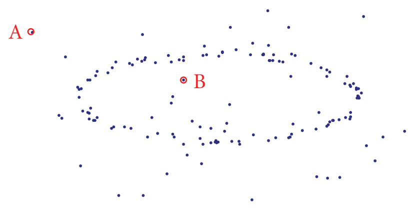


Figure 1: Two different types of outliers. Proximity-based methods are good at detecting outliers similar to point A, while model-based methods can better detect outliers similar to point B.

more commonly used and efficient algorithms, such as least-squares.

Being a well-developed field itself, outlier detection has been widely studied from very different perspectives, such as statistics, neural networks and machine learning. Hodge and Austin [9] provided a comprehensive survey of outlier detection algorithms. Among all these methods, the proximity-based outlier detection algorithms have the advantage that they do not require prior knowledge about the distribution of the observations. The most widely applied group of proximity-based algorithms is k-Nearest Neighbor (k-NN). For example, Byers and Raftery [10] proposed a method that determines the distance D_k from an observation to its kth nearest neighbor and compares it with a threshold d. If $D_k > d$, then the observation is classified as an outlier. In 2000, Ramaswamy et al. [11] proposed a similar algorithm that ranks the observations according to D_k in descending order and marks the top n observations in the ranking as outliers. In addition, Knorr and Ng [12] introduced an algorithm that classifies an observation as an outlier if less than m of its k nearest neighbors lie within a disk of radius d. Hautamäki et al. [13] proposed an algorithm based on a weighted directed graph (k-nearest neighbor graph). The graph is constructed in such a way that each vertex (observation) has k directed edges to its k nearest neighbors with edge weight being the distance between the observation and its neighbor. Any vertex that has indegree less than a threshold T is classified as an outlier. k-NN flavored algorithms can be applied to various outlier detection problems, for example, spatial outlier detection [14].

Despite a large number of proximity-based algorithms available, we still need to design a new outlier elimination algorithm in this work, due to the special properties of the curve and surface fitting problem. Proximity-based algorithms alone are not sufficient to eliminate all the outliers for the fitting problem. One of the difficulties is that the outliers and inliers usually intertwine in a complex manner, as shown in Figure 1. Some of the outliers are distant from the curve (like point A), yet others may fall closer to the curve (like point B). The outlier detection algorithms based on proximity may not be effective for detecting the B-type outliers. Moreover, even for A-type outliers, k-NN flavored algorithms fail in certain scenarios (see Section 3 for further details).

In order to better handle the B-type outliers, we may resort to model-based outlier detection algorithms like random sample consensus (RANSAC) [15]. RANSAC works nicely in various applications, such as [16], in which the percentage of outliers is relatively low and the inliers have little noise. However, with a high percentage of outliers, when the number of the parameters of the model is large, even if the inlier noise is negligible, the computational cost of RANSAC looms as a serious problem. On defining w as the portion of inliers, d as the minimum number of observations needed to fit a model, and p as the probability of successfully finding the correct model after running RANSAC k times, we have the following relationship

$$p = 1 - (1 - w^d)^k. \tag{1}$$

When w is small (the percentage of outliers is high), we need k to be large in order to have a sufficiently high p. Unfortunately, the increase in k with respect to d is exponential. If we assume $w = 0.5$, to guarantee $p = 0.99$, we need $k = 16$ for a simple two-dimensional straight line model; yet for ellipse fitting, where $d = 5$, we need $k = 146$. For an ellipsoid ($d = 9$), we need $k = 2356$. Moreover, when d is large, the fitting algorithm itself becomes rather expensive to run even once.

In one of the latest variations of RANSAC [17], the number of iterations needed can be dramatically decreased by replacing pure random sampling (which is the origin of the name RANSAC) by guided sampling, in which promising "inliers" are given more weight at resampling. Moreover, unlike the vanilla version of RANSAC, which uses the number of observations in accordance to the suggested model to evaluate its goodness, a more robust measure, (weighted) median absolute deviation (MAD/WMAD) is introduced to reduce the influence of outliers on model evaluation. In terms of total numbers of fittings needed, this algorithm performs very well in scenarios of low inlier noise. However, the algorithm is susceptible to falling into local minima and generating models completely different from the desired one. This phenomenon is aggravated when the level of noise on the inliers is high and the portion of outliers is large. The details of this phenomenon can be found in the simulation results of Section 5.

Another approach to robustly fit curves and surfaces is by enumerating all possible solutions. For example, in the case of ellipse fitting, Five Point Fit Ellipse Fitting (FPFEF) [18] is an algorithm in which all five-tuples of observations are selected and fit by ellipses, the median of the parameters of which being the final fitting result. In terms of robustness, this algorithm performs competitively; however, the computational complexity is essentially combinatorial, which is unacceptable when the number of parameters is high. Specifically, for a model with d parameters, suppose the total number of observations (including both inliers and outliers) is K; then the total number of iterations needed for FPFEF is

$$\binom{K}{n} = O(K^n). \tag{2}$$

For a reasonable ellipse fitting problem with 100 observations, the total number of parameters is $n = 5$, and thus we need approximately 75 million iterations to enumerate all 5-tuples in order to fulfill the requirement of the FPFEF algorithm.

To counter these problems, in this paper, we develop a two-stage hybrid outlier detection algorithm for curve and surface fitting that combines the proximity-based and model-based outlier detection algorithms. First, with the help of algebraic graph theory [19], we employ an outlier detection algorithm based on the graph formed by the observations, using their proximity information. Second, after reducing the portion of outliers within the remaining set of observations, we employ a model-based outlier detection algorithm to efficiently refine the results. In the first stage, a large portion of the most distant, isolated outliers, which can be a great hazard to model-based outlier detection methods, are detected and eliminated. In the second stage, the subtler, closer outliers that go against the curve/surface model are detected, and some of the misclassified inliers are retrieved. These two stages compensate for each other to form a more efficient and accurate outlier detection algorithm. To our knowledge, there are no such hybrid outlier elimination algorithms discussed in the literature for curve/surface fitting on point cloud data.

The structure of the paper is as follows: in Section 2, the model of our outlier detection problem is specified with several important assumptions. Section 3 describes our proximity-based outlier detection algorithm, the first stage of our hybrid outlier detection algorithm. By employing hard and soft thresholds, the graph-component-based and eigenspace-based algorithms are described, respectively. As a necessary supplement, the estimation of the length of a curve based on proximity information is also introduced in Section 3. Section 4 summarizes the model-based outlier detection algorithm. The simulation results, taking ellipse and ellipsoid fitting as examples, are described in Section 5. Section 6 concludes our work.

2. Model and Assumptions

First, we need to specify the model by stipulating a few assumptions about the data, both for inliers and outliers. Let $\mathcal{M}$ be a d-dimensional manifold with finite volume V in a d^*-dimensional space $\mathbb{R}^{d^*}$ ($d \leq d^*$). Specifically, an ellipse is a 1-dimensional manifold in 2-dimensional space. We define the inliers as observations that are drawn from $\mathcal{M}$ independently with a certain distribution, and then possibly corrupted by a small amount of noise. Due to the noise, the inliers might fall outside of $\mathcal{M}$. Outliers are different in the sense that they are at a larger distance from $\mathcal{M}$. We define δ-outliers as outliers whose distance to $\mathcal{M}$ is at least δ in the space $\mathbb{R}^{d^*}$.

In addition, we need to assume that the inliers are the majority of the observations, more than 50% of all the observations.

The basic idea of our proximity-based outlier detection is that if the inliers are all distributed over a manifold $\mathcal{M}$ of finite volume, then when the number of the inliers N is sufficiently large, with high probability, the distances between inliers become sufficiently small such that if we connect all the observations within distance δ, the inliers will form a strongly connected component in a proximity graph. Therefore, by selecting the largest strongly connected component formed by connecting observations within radius δ, we can effectively detect δ-outliers, that is, observations at least δ-distant from $\mathcal{M}$.

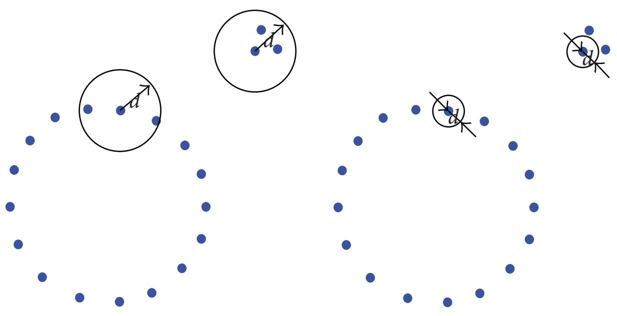


Figure 2: k-NN algorithm failure example.

To tackle the outliers that lie close enough to $\mathcal{M}$ (distance less than δ), we use a model-based algorithm so that only the observations that agree with the model of $\mathcal{M}$ are classified as inliers. Since the proximity-based outlier detection algorithm can guarantee that all remaining outliers are at most δ distant from $\mathcal{M}$, it is rather easy to use implementation-friendly l_2 loss functions for our model-based algorithm, instead of resorting to other more computationally costly algorithms.

3. Proximity-Based Algorithm

Although there are many proximity-based algorithms available, they may not be appropriate in the context of curve/surface fitting. Take the vanilla version of the k-NN algorithm [10] as an example.

In Figure 2, a collection of observations (inliers) on a circle together with a small cluster of outliers are fed to the k-NN outlier elimination algorithm. Suppose we set $k = 2$. Then as shown in the left hand side of the figure, when the radius of d is set to be large enough so that all observations on the circle are classified as inliers, the three outliers are also classified as inliers.

On the other hand, as illustrated in the right-hand side of Figure 2, when d is sufficiently small so that the observations in the small cluster disqualify as inliers, so do all the true inliers. In this scenario given $k = 2$, no matter how we select d, there is no possible way to eliminate the outliers.

However, if we take into consideration the connectivity of graph components formed by links within radius d in the case shown in the left hand side of Figure 2, then the entire set of observations can be easily separated into two connected components, one much larger than the other, which can be used as an argument for classifying the observations in the small cluster as outliers. Therefore, we need a proximity-based algorithm that takes into consideration the connectivity information among observations in a more holistic way beyond the k-NN flavored algorithms, which only use information of local neighborhoods of observations.

```
Data: $K$ observations $\{(x_i, y_i)\}_{i=1}^{K}$
Result: $\mathcal{I}$, indices of inliers in the dataset
begin
    forall the $i, j \in 1, 2, \dots, K$ do
        $Q_{ij} \leftarrow$ Euclidean distance between $(x_i, y_i)$ and $(x_j, y_j)$
    end
    Estimate the curve length $L$ based on $Q$ according to Algorithm 2
    forall the $i, j \in 1, 2, \dots, K$ do
        if $Q_{ij} - \alpha L > 0$ then
            $W_{ij} \leftarrow 0$, no edge between vertexes $i$ and $j$
        else
            $W_{ij} \leftarrow 1$, edge added between vertexes $i$ and $j$
        end
    end
    Get the set of connected components $\{\mathcal{C}_k = (\mathcal{V}_k, \mathcal{E}_k)\}_{k=1}^{S}$ of the graph corresponds to matrix $W$
    $\mathcal{I} \leftarrow \{i : i \in \mathcal{V}_k, |\mathcal{V}_k| > \beta K\}$
end
```

Algorithm 1: Graph-component-based algorithm.

Consequently, it is desirable to develop a graph-based algorithm to fully exploit the proximity information for outlier detection.

3.1. Graph-Component-Based Outlier Detection

3.1.1. Detecting Outliers from Graph Structure. As described in Section 2, when the number of inliers is sufficiently large, with high probability, the inliers will form a strongly connected graph component. On the other hand, the δ-outliers, distant from the inliers as well as other outliers, will not form connected components with many vertices. As a result, we can effectively detect δ-outliers.

For the special case when $\mathcal{M}$ is a closed curve, we have been able to derive a bound on the probability of detecting δ-outliers. It shows that on a closed curve of circumference L, if the N independent inliers are uniformly distributed on the curve, then the probability that they fail to form a strongly connected graph component (with all points within distance $\delta = \alpha L$ connected) decays asymptotically as $O(Ne^{-\alpha N})$. The details of the bound is described as follows.

On a closed curve of length L, on which a direction is defined, if there are N independent uniformly distributed observations, then with probability no less than $1 - \epsilon$, each observation has a neighbor within on-curve distance (curve length) $\delta = \alpha L$ $(0 < \alpha < 1)$ along the curve direction, where

$$\epsilon = \frac{Ne^{-\alpha N}}{1 - \alpha}. \tag{3}$$

The derivation of the bound can be found in Appendix A.

According to this bound, with probability $1 - \epsilon$, each observation on the curve has a neighbor within on-curve distance $\delta = \alpha L$ along the curve direction, and thus its Euclidean distance to its neighbor is less than δ. So, each observation is connected to its neighbor. Therefore, in the proximity graph formed by connecting all neighbors within radius δ, there exists a single loop that connects all the observations. As a result, the generated graph is strongly connected. Thus, for a given ϵ, by properly selecting δ according to the bound, we can effectively eliminate δ-outliers by simply classifying large connected components as inliers and others as outliers.

The graph-component-based outlier detection algorithm for curves is based on the reasoning above. After selecting an appropriate value of α, and taking $\delta = \alpha L$, we form a proximity graph by connecting observations if the Euclidean distance between them is less than δ. Then, we classify all the connected components that have less than βK vertices as outliers, with K being the total number of observations. Practically, β ranging from 0.05 to 0.1 is an appropriate choice. Algorithm 1 summarizes the details of detecting αL-outliers for a closed curve of circumference L in 2-dimensional space.

It is worth pointing out that the length L of the curve, or more generally, the volume of $\mathcal{M}$ needs to be estimated from the observations. This is described in Section 3.1.2.

3.1.2. Estimating the Volume of $\mathcal{M}$. The efficacy of our outlier detection algorithm is directly related to the choice of δ. A small value of δ leads to small probability of forming a strongly connected component of all the inliers, yet a large value of δ results in tolerance to wilder outliers, which is rather disastrous to the second stage model-based outlier detection algorithm.

The choice of δ is directly related to the volume of the manifold $\mathcal{M}$ in which the inliers reside. In order to select a proper δ, one needs to have an estimate of the volume V of $\mathcal{M}$, without even having an accurate model fitted to it. Specially, in the one-dimensional case, the length of the curve L is essential to outlier detection. As a result, we provide an estimate ofthe volume of $\mathcal{M}$ as follows.

Suppose $\mathcal{M}$ is a d-dimensional manifold, and that N observations are drawn independently from $\mathcal{M}$. If the probability density of the observations is given by $\rho(\mathbf{x})$, $\mathbf{x} \in \mathcal{M}$, then

$$\begin{aligned}\mathbb{P}(&\text{Nearest neighbor distance} \leq y) \\ &\approx (N-1)\mathcal{V}\left(B_y^{(d)}\right) \int_{\mathcal{M}} \rho^2(\mathbf{x})\mathrm{d}\mathbf{x},\end{aligned} \tag{4}$$

where $\mathcal{V}(B_y^{(d)})$ stands for the volume of a d-dimensional ball with radius y. Specially, if the observations are uniformly distributed, then

$$\mathbb{P}(\text{nearest neighbor distance} \leq y) \approx \frac{(N-1)\mathcal{V}\left(B_y^{(d)}\right)}{V}. \tag{5}$$

In the special case in which a curve of length L is considered, and in which N observations are independent and uniformly distributed on the curve, then

$$L \approx \frac{2(N-1)y}{\mathbb{P}(\text{nearest neighbor distance} \leq y)}. \tag{6}$$

The details of the estimates (5) and (6) can be found in Appendix B.

In practice, the distribution $\mathbb{P}$(nearest neighbor distance $\leq y$) can be estimated from the histogram of the nearest-neighbor distances of each observation (or resampled observations, if performed in a bootstrapping manner). To ensure that all the nearest-neighbor distances are between the inliers, it is desirable to truncate the tail of the histogram. In the worst case, we can cut off 50% of the nearest-neighbor distances to roughly ensure that no outliers mix in. Then the value of N is simply the remaining number of observations after cutting off the tail. With properly selected quantile y, the curve length L can be readily estimated.

The reason that we only use nearest-neighbor distances in our estimate is because nearest-neighbor distances are fairly good approximations (from above) to the true distances on the manifold. Other distances among the inliers could be informative, yet not compatible with our measure on the manifold.

If the inliers are contaminated by noise, then even the nearest-neighbor distances are not very accurate estimates of the distances on the manifold. Estimates based on (6) could deteriorate. Estimates given by (6) in the noisy case are larger than the true volume of the manifold.

The algorithm for estimating the length of a curve is summarized in Algorithm 2. The basic idea is to first take a vector $\mathbf{m}$ of nearest-neighbor distances, sorted in ascending order. Then, cut off 50% of the tail of the vector, and use the histogram of the remaining nearest-neighbor distances to estimate the curve length L according to (6).

3.2. Eigenspace-Based Algorithm. In practice, graph-component-based outlier detection works very well when there is no noise or a very small amount of noise on the inliers, since in this case, our estimate of the volume of $\mathcal{M}$ is very accurate and we are able to choose δ properly. However,

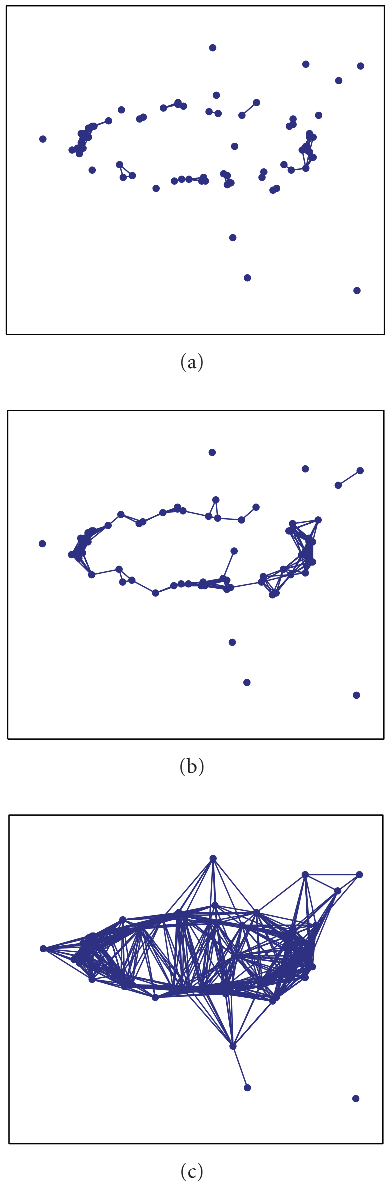


FIGURE 3: Different choices of connection radius lead to different results. The figure on the left depicts the situation in which the radius is so small that even the inliers do not connect sufficiently with each other. The figure on the right illustrates the case in which the radius is so large that almost all the observations are connected with each other, and thus makes it impossible to distinguish inliers from outliers. The figure in the middle shows that with a properly selected radius of connection, the inliers form a large connected component, leaving most of the outliers unconnected.

due to the noise on the inliers, we usually cannot determine a very accurate estimate of the size of $\mathcal{M}$. In this case, if a hard connection rule is applied to generate the graph, it might lead to an over-connected graph in which too many outliers are connected to inliers, or an under-connected graph, in which the inliers are isolated in several pieces, as shown in Figure 3.

To solve this problem, we choose to use a soft threshold (heat kernel) to determine the connectivity among observations, and then analyze the eigenspace (ideally, the null space)

```
Data: $Q$, with $Q_{ij}$ being Euclidean distance between $(x_i, y_i)$ and $(x_j, y_j)$
Result: Curve length estimate $L$
begin
    for $i \in 1, 2, \ldots, K$ do
        $\mathbf{m}_i \leftarrow \min\{Q_{ij}, j \in \{1, 2, \ldots, K\} \setminus \{i\}\}$
    end
    $\mathbf{m}' \leftarrow$ sort $\mathbf{m}$ in ascending order
    $\mathbf{m}^* \leftarrow$ take the first $\lfloor K/2 \rfloor$ elements of $\mathbf{m}'$
    $q \leftarrow p$-quantile of $\mathbf{m}^*$
    $L \leftarrow 2q(\lfloor K/2 \rfloor - 1)/p$
end
```

Algorithm 2: Circumference estimation of a curve (one dimensional manifold).

of the graph Laplacian of the resulting weighted undirected graph to detect outliers.

3.2.1. Constructing Proximity Graph and Its Laplacian. According to algebraic graph theory, the connectivity information about a graph is reflected in the graph Laplacian and its corresponding eigenvalues and eigenvectors. Instead of using an unweighted graph with a hard connection rule as in the case of the previous subsection, the graph Laplacian is constructed using heat kernels in the following manner:

(1) form a matrix Q with Q_{ij} being the Euclidean distance between the observations (x_i, y_i) and (x_j, y_j);

(2) form a fully connected proximity graph with edge weight $W_{ij} = e^{-Q_{ij}^2/t}$ (heat kernel);

(3) construct the graph Laplacian matrix: $\mathcal{L} = D - W$, where D is a diagonal matrix with its diagonal entries as column sums (or row sums) of W;

(4) compute eigenvalues and eigenvectors for the generalized eigenvector problem, $\mathcal{L}\mathbf{f} = \lambda D\mathbf{f}$.

Then, the set $\{\lambda \mid \lambda \approx 0, \lambda \in \{\lambda_i\}_{i=1}^K\}$ and its corresponding eigenvectors form the cornerstone of our proximity-based outlier detection algorithm. The algorithm for constructing the graph Laplacian is summarized in Algorithm 3.

The success of proximity-based outlier detection depends heavily on the choice of the "radius of connection" (i.e., $\sqrt{t}$ in the heat kernel). Thus, the value of t is of essential importance. Yet since the heat kernel is a soft threshold, it is more tolerant to this choice compared to the choice of δ in the graph-component-based algorithm. Still, the choice of t is a rather delicate problem. According to algebraic graph theory, t is closely related to the algebraic connectivity of the graph (the smallest nonzero eigenvector of the graph Laplacian). Here, we present an empirical rule for determining a good value of t. Sort all the elements of Q in ascending order and denote the resulting vector as $\mathbf{q}$. Then take the pKth element of $\mathbf{q}$ as $\sqrt{t}$; that is, assign $\sqrt{t}$ as the p-quantile of the distribution of all the elements of Q. This bears some similarity to the way we estimate L and select δ.

Intuitively, most of the small elements of Q are composed of the distances between inliers, according to our assumptions. By taking the pKth element of $\mathbf{q}$ as $\sqrt{t}$, we approximately guarantee a connected subgraph among the inliers of average degree $p - 1$, if we assume e^{-1} is the threshold for a "connection". Empirically, $p = 4$ works well.

```
Data: $K$ observations $\{(x_i, y_i)\}_{i=1}^K$
Result: The eigenspace of the Graph Laplacian $\mathcal{L}$
begin
    forall the $i, j \in 1, 2, \ldots, K$ do
        $Q_{ij} \leftarrow$ Euclidean distance between $(x_i, y_i)$ and $(x_j, y_j)$
    end
    Choose radius of connection $t$ based on $Q$
    forall the $i, j \in 1, 2, \ldots, K$ do
        $W_{ij} \leftarrow e^{-Q_{ij}^2/t}$
    end
    $D \leftarrow \mathbf{diag}(\mathbf{1}^T W)$
    $\mathcal{L} \leftarrow D - W$
    Solve $\mathcal{L}\mathbf{f} = \lambda D\mathbf{f}$
end
```

Algorithm 3: Construct the graph Laplacian.

3.2.2. Selecting Outliers in Eigenvectors. Given an appropriate value of t, the inliers form a strongly connected component, leaving most of the outliers loosely connected to it and each other. As a result, with a proper interchange of rows and columns, the matrix Q, and thus the Laplacian matrix $\mathcal{L}$, is close to a block diagonal matrix. Each block corresponds to a strongly connected component, with the largest one corresponding to the inliers.

For each block, there is an eigenvalue approximately equal to zero, and the corresponding eigenvector is almost a binary vector composed of 0's and 1's, with only the elements aligned with the block being 1's. Here, we assume that the l^∞ norms of all the eigenvectors are normalized to 1. Among all the approximate binary eigenvectors that correspond to the close-to-zero eigenvalues, some contain more 1's than others. The key step of outlier elimination is to eliminate those observations that correspond to the nonzero elements of binary eigenvectors with very few 1's, because the components composed of these observations are very weakly connected to other components, and thus are more likely to be outliers.

```
Data: Eligible eigenvectors $\{\mathbf{f}_i\}_{i=1}^{l}$, $\mathbf{f}_i \in \mathbb{R}^k$
Result: $\mathcal{O}$, indices of outliers in the set of observations
begin
    foreach $\mathbf{f}_i$, $i \in 1, 2, \ldots, l$ do
        Initialize:
            $\mathcal{I}'_i \leftarrow$ indices of randomly chosen $\left\lfloor \frac{K}{2} \right\rfloor$ elements of $\mathbf{f}_i$
            $\mathcal{I}_i \leftarrow null$
        while $\mathcal{I}_i \neq \mathcal{I}'_i$ do
            $\mathcal{I}_i \leftarrow \mathcal{I}'_i$
            $\alpha_{1/4} \leftarrow$ 25% quantile of $\{\mathbf{f}_{i,j\in\mathcal{I}_i}\}$
            $\mu \leftarrow$ 50% quantile of $\{\mathbf{f}_{i,j\in\mathcal{I}_i}\}$
            $\alpha_{3/4} \leftarrow$ 75% quantile of $\{\mathbf{f}_{i,j\in\mathcal{I}_i}\}$
            $I \leftarrow [\mu - \gamma(\mu - \alpha_{1/4}), \mu + \gamma(\alpha_{3/4} - \mu)]$
            $\mathcal{I}'_i \leftarrow \{j : \mathbf{f}_{i,j} \in I\}$
        end
        $\mathcal{O}_i \leftarrow \{j : j = 1, 2, \ldots, K, j \notin \mathcal{I}_i\}$
    end
    Output $\mathcal{O} \leftarrow \bigcup_{i=1}^{l} \mathcal{O}_i$
end
```

Algorithm 4: Select outliers in the eigenvectors.

Therefore, finding outliers is approximately equivalent to finding protruding 1's in the eigenvectors with close-to-zero eigenvalues. Thus, we have reduced a problem of high-dimensional outlier detection with a complex hypothesis into a one-dimensional outlier-detection problem. However, since the eigenvectors that correspond to close-to-zero eigenvalues are not perfectly binary, more elaborate methods need to be employed to detect outliers.

First, eligible eigenvectors (approximate binary eigenvectors) are selected from the collection of eigenvectors. Specifically, we take the eigenvectors that correspond to eigenvalues less than 0.1 as the candidates. Then, keeping only the binary eigenvectors, the "high frequency" eigenvectors are excluded. Note that $\mathbf{f}$ is a "high frequency" eigenvector if $(\sum_j |\mathbf{f}_j| - |\sum_j \mathbf{f}_j|)/\sum_j |\mathbf{f}_j|$ is sufficiently large; that is, those eigenvectors with both large; positive and negative elements are excluded. After that, a one-dimensional outlier detection algorithm is employed, as shown in Algorithm 4.

Algorithm 4 does not guarantee the "correct" result. It can be helpful to run it several times and choose the result with minimum intraclass deviation (as in the case of RANSAC). However, in our simulations, we ran the routine only once, which was good enough for our data.

4. Model-Based Algorithm

The algorithm described in the previous section uses only the proximity information of the observations. However, it is rather incapable of detecting type-B outliers in Figure 1, and more specifically, outliers that lie within the range of less than δ from $\mathcal{M}$, even if they obviously deviate from the true model.

To eliminate type-B outliers, we need to use the additional prior knowledge that the inliers are located on a manifold with certain shape and size; for example, an ellipse. Since we have greatly reduced the portion of outliers in the first step, we can of course achieve this goal by running a model-based outlier detection algorithm.

Moreover, since our proximity-based algorithm also guarantees, with high probability, that the remaining outliers are close enough to the true model, we can even be more efficient by initializing our model using all the observations classified as inliers in the first step, and iteratively eliminating the ones whose distances to the fitted model exceeds a threshold, until convergence. We can even retrieve misclassified inliers, if any, given that they lie right on the model fitted in the second step, the model-based algorithm.

We summarize the model-based outlier detection algorithm as follows:

(1) fit a model h to the "inliers" selected by the proximity-based outlier detection algorithm;

(2) test all the observations with respect to h, classify the observations that saliently deviate from h (above a threshold) as outliers and then classify other observations as inliers;

(3) refit model h based on the updated inliers;

(4) repeat step (2) and step (3) until the classification does not change any further.

Specifically, for algebraic fitting of curves, the second step of the algorithm is shown in Algorithm 5.

With this model-based outlier detection algorithm, we are able to detect most of the outliers missed in the first stage (usually harder ones very close to the inliers), and also clear the labels for misclassified inliers.

In addition, due to the fact that the number of outliers is rather small for the second step, and they are all close enough to the true model, the convergence of the second step is usually very fast.

```
Data: Inliers selected by a proximity-based algorithm, ${(x_i, y_i)}_{i\in\mathcal{I}}$
Result: $\mathcal{I}$, indices of inliers in the data set
begin
    Initialize: $\mathcal{I}' \leftarrow \mathcal{I}$, $\mathcal{I} \leftarrow null$
    while $\mathcal{I} \neq \mathcal{I}'$ do
        $\mathcal{I} \leftarrow \mathcal{I}'$
        fit a model $h(x_i, y_i) = 0$ to $\{(x_i, y_i)\}_{i\in\mathcal{I}}$
        compute average fitting deviation $\sigma = \sqrt{\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} h(x_i, y_i)^2}$
        $\mathcal{I}' \leftarrow \{i : h(x_i, y_i) < \alpha\sigma\}$
    end
    Output $\mathcal{I}$
end
```

Algorithm 5: Model-based algorithm.

5. Simulation Results

Among many curve/surface fitting problems of possible interest, we choose ellipse and ellipsoid fitting as our subject for simulation, because these two fitting problems are widely useful in various application scenarios, yet are sufficiently difficult, in terms of the number of parameters and the computational cost for each individual fitting, so that RANSAC is rather costly when the fraction of outliers is large.

In the outlier detection problem for ellipse and ellipsoid fitting, we assume that there are a total number of $K = M+N$ observations $\{(x_i, y_i)\}_{i=1}^{K}$ with N inliers and M outliers. The inliers are sampled on an ellipse and then contaminated by Gaussian noise with standard deviation σ_0; and we simply model the outliers as points on the ellipse added by a significantly intense Gaussian noise, with standard deviation $\sigma_1 > \sigma_0$.

Since we are interested in the case in which inliers are contaminated by nonnegligible noise, the eigenspace-based algorithm (which employs soft links among observations) is used as the algorithm in the proximity-based stage. The value of $\sqrt{t}$ is selected as the $4K$th element of $\mathbf{q}$ as in the explanation of Algorithm 3. Moreover, for the second stage of our algorithm, we choose $\alpha = 3$ in Algorithm 5 throughout our simulation.

5.1. A Typical Ellipse Fitting with Outlier Elimination. It is of interest to inspect a typical simulation for outlier detection on an ordinary ellipse to see the performance of our two-stage algorithm. In the simulation, we have $N = 100$ inliers and $M = 50$ outliers. The standard deviation of the independent Gaussian additive noise of inliers in both x and y directions is $\sigma_0 = 0.1$, and that of the outliers is $\sigma_1 = 2$. The true ellipse has an eccentricity $\epsilon = 0.95$ with semimajor length $a = 5$, and takes the standard position (centered at the origin, with semimajor axis aligned with the x axis).

The outlier detection results of our algorithm on an ellipse are shown in Figure 4. Note that the first stage of our algorithm makes several mistakes, with several outliers missed (the ones inside and closely outside the ellipse) and a few inliers misclassified (several circled observations on the perimeter of the ellipse). These mistakes are corrected by the second-stage model-based algorithm. And the final result leaves us a group of purged, low-noise observations from an ellipse.

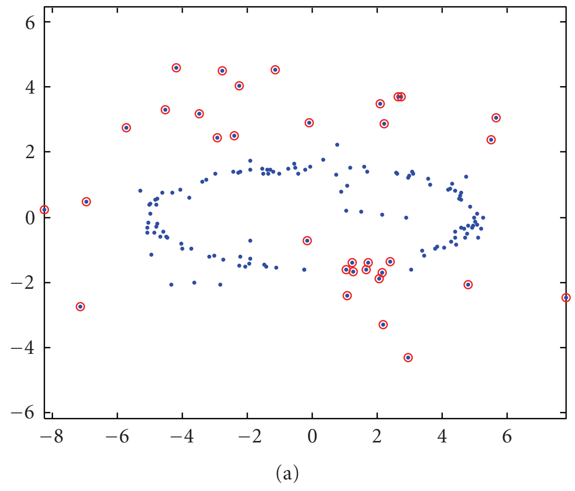
5.2. Improvement in Performance. It is worthwhile to compare the performance of fitting procedures with and without outlier elimination for different numbers of outliers. In this simulation, we choose $N = 100$, M ranging from 1 to 55, $\sigma_0 = 0.1$, and $\sigma_1 = 3$, keeping the true model for the ellipse the same as in the previous simulation.

To measure the performance of ellipse fitting algorithms, we employ the nonoverlapped area of the fitted and the true ellipses (normalized by the area of the true ellipse). To be more specific, the fitting error is defined as the normalized area difference between the true ellipse E_t and the fitted ellipse E_f:

$$\text{error rate} = \frac{S_{E_t\cup E_f} - S_{E_t\cap E_f}}{2S_{E_t}}, \tag{7}$$

where $S_{E_f\cup E_f} - S_{E_f\cap E_f}$ is the area difference and S_{E_t} denotes the area of the true ellipse. This measure is henceforth called relative area difference. This simulation result is shown in Figure 5. Each point in the figure is the average performance of 200 independent trials. It is obvious that the outlier detection algorithm effectively eliminates most of the outliers, even when the fraction of outliers is close to 50%.

5.3. Comparison with Vanilla RANSAC. Not only does the hybrid algorithm perform well in terms of small fitting error, it is also attractive computationally. Given that the model is complex enough, fitting the model to the observations can be deemed to be the most costly step. Therefore, the total number of times that the model fitting step is implemented can be a good measure of computational complexity of the outlier eliminating algorithms. A desirable algorithm should be able to achieve a reasonably good result in a small number of iterations. Since in the hybrid algorithm, the first-stage proximity-based algorithm eliminates a large portion of the wild outliers, the second stage empirically



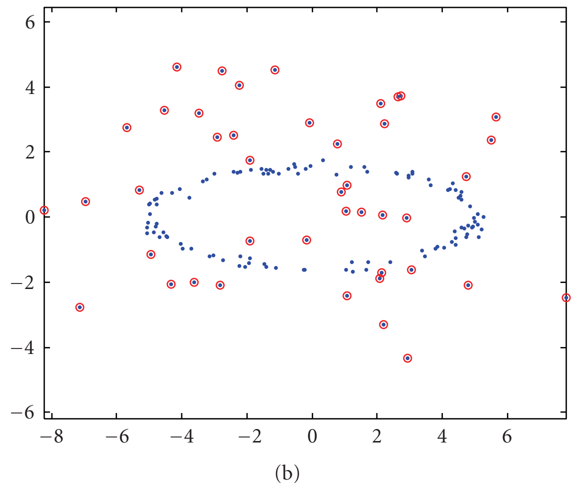


Figure 4: Outlier detection results for ellipse fitting. The circled observations are classified as outliers by the algorithm. The figure on the left demonstrates the detected outliers after the proximity-based algorithm, in which there are several missed outliers close to the ellipse, while a few inliers are misclassified as outliers. The figure on the right shows that these mistakes are corrected by the model-based algorithm.

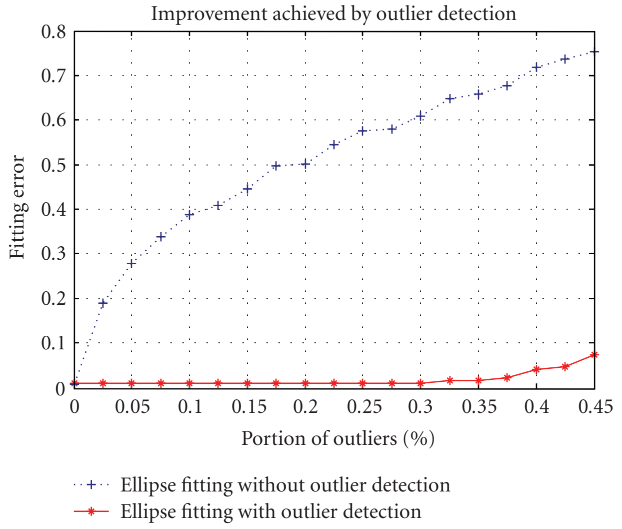


Figure 5: Comparison between ellipse fitting algorithms with and without outlier detection using the hybrid detection algorithm.

converges very fast, and is to some extent irrelevant to the percentage of outliers. On the contrary, to guarantee a similar performance, the number of iterations needed for the vanilla version of RANSAC increases rapidly as the percentage of inliers w decreases. According to (1), given that w^d is sufficiently small, the total number of needed iterations k satisfies

$$k \approx C \cdot \left(\frac{1}{w}\right)^d. \tag{8}$$

Obviously when d is large ($d = 5$ for ellipse fitting) and w is small, the number of iterations k can be enormous.

Therefore, we demonstrate the comparison of our hybrid algorithm and vanilla RANSAC in two ways. First, given a maximum number of iterations for vanilla RANSAC, which almost ensures the same computational complexity as our hybrid algorithm (both are almost constant-time), we compare the average fitting errors (in relative area difference) of the two methods. Second, given that our algorithm and the vanilla version of RANSAC have the same performance measured in terms of relative area difference, we compare the average numbers of iterations needed for the hybrid algorithm (second stage) and the vanilla RANSAC.

In this simulation, we choose inlier noise level as $\sigma_0 = 0.1$ and outlier noise level as $\sigma_1 = 3$, keeping the geometry of the ellipse unchanged. Moreover, we fix the total number of inliers as $N = 100$, and change the number of outliers to achieve different outlier percentage of the total points, $M/(M + N)$. For the first purpose, we fix the total number of iterations of vanilla RANSAC to 1000, and vary the outlier percentage from 0% to 45%. For the second purpose, we enforce the constraint that the RANSAC algorithm reach the same performance (in terms of relative area difference) as our hybrid algorithm, and record the averages of the numbers of iterations of both algorithms, with the outlier percentage ranging from 0% to 25%. For computational reasons, we set an upper limit of 20000 iterations for both algorithms.

For each percentage of outliers, we run the two algorithms 200 times, and record the averaged results in Figure 6 and Table 1.

In the simulation, our hybrid algorithm has a considerable advantage in terms of the total number of fittings. To achieve the same performance, the second stage of the hybrid algorithm needs only a few iterations, as compared to thousands of iterations for vanilla RANSAC. If we force RANSAC to terminate in 1000 iterations, its average performance deteriorates rapidly with respect to the percentage of outliers.

Moreover, close inspection indicates that the RANSAC algorithm, when the percentage of the outliers is high, very frequently reaches the maximum number (20000) of iterations, which gives the hybrid algorithm an advantage in terms of the number of iterations.

Table 1: Average number of iterations of RANSAC and the hybrid algorithm for different percentages of outliers.

Percentage of ouliers (%)	5	7.5	10	12.5	15	17.5	20	22.5	25
Average relative area difference (%)	1.0	1.0	1.0	1.0	1.0	1.0	1.1	1.1	1.1
Average iterations of RANSAC	174	221	288	386	1246	1534	1739	2351	3364
Average iterations of hybrid (2nd stage)	2.1	2.2	2.4	2.5	2.8	3.0	3.2	3.7	4

Table 2: Failure rates of RANSAC-EIS-Metropolis and the hybrid algorithm.

Inlier noise level σ_0	0	.01	.02	.04	.08	.16	.32
RANSAC-EIS-Metropolis failure rate (%)	0	1.5	2.5	7.5	9.5	27.5	36.5
Hybrid algorithm failure rate (%)	0	0	0	0	0	0	.5

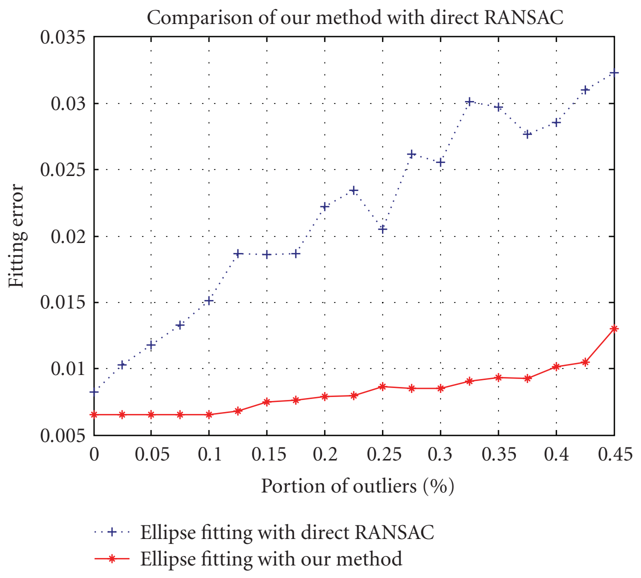


Figure 6: Comparison of performance (measured in relative area difference) between RANSAC and the hybrid algorithm, with the maximum of number of iterations set to 1000.

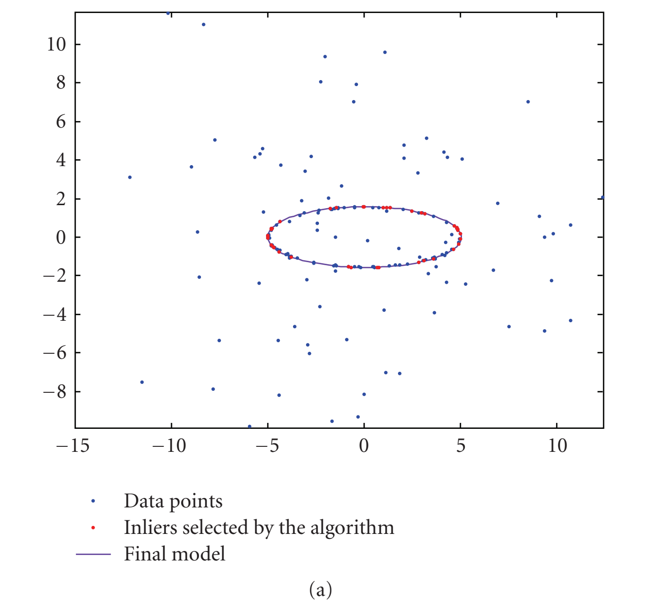
5.4. Comparison with RANSAC of Guided Sampling. To further demonstrate the efficacy of our algorithm, especially its robustness to inlier noise, we compare it with one of the variations of the RANSAC algorithm: RANSAC with efficient sampling using Ensemble Inlier Sets [17] (RANSAC-EIS-Metropolis).

RANSAC-EIS-Metropolis converges very rapidly compared to its predecessors. It works very effectively when the noise level of inliers is low, as shown in the left hand side of Figure 7, in which case the inlier noise level is $\sigma_0 = 0.02$. However, when the inlier noise level is high and the fraction of outliers is large, the criterion for evaluating the goodness of the model (MAD/WMAD) becomes less effective and there is a high probability that the algorithm will fail to find the correct model, as shown in the left hand side of Figure 7, in which case the inlier noise level is $\sigma_0 = 0.08$ (lower than all our previous simulations). To show the influence of the inlier noise level on the effectiveness of the RANSAC-EIS-Metropolis algorithm, and to demonstrate the robustness of our algorithm with respect to the inlier noise, we compare our algorithm to the RANSAC-EIS-Metropolis algorithm in terms of failure rate under different inlier noise levels. We use 100 inliers and 80 outliers, with inlier noise level σ_0 increasing from 0 to 0.32 and outlier noise level $\sigma_1 = 10$ fixed. The failure rate is measured as the percentage of fitting results having relative area difference beyond 0.3 in 200 trials. For the RANSAC-EIS-Metropolis algorithm, we set the maximum number of iterations as 500, much higher than the average number of iterations needed for a similar ellipse fitting problem described in [17]. Keeping everything else the same, the following results are obtained, as displayed in Table 2.

For low inlier noise levels, the two algorithms perform competitively. However, as the inlier noise level increases, the RANSAC-EIS-Metropolis algorithm fails to find the true model at a considerable rate, while the hybrid algorithm is not affected until the noise level is as high as $\sigma_0 = 0.32$. Although RANSAC-EIS-Metropolis is superior for low levels of inlier noise (which is usually the case for computer vision applications), as demonstrated in [17], it is not very resistant to the noise on inliers. Our hybrid algorithm, on the other hand, shows greater robustness to inlier noise level, which makes it more appropriate for application scenarios like fitting noisy data generated by radar and sonar.

This simulation results have another implication: if the model of the inliers is not identical to the true model, then even if there are no observational errors, there still exist systematic errors, which can also be a hazard to the RANSAC-EIS-Metropolis algorithm. Therefore, for applications in which the inlier data cannot be accurately captured by the model, it might be safer to use a hybrid algorithm, which is resistant to inaccuracy in the fitting model.

5.5. Robustness to Different Types of Outliers. Since our outlier detection algorithm is a hybrid, it is able to tackle a broad variety of outliers and the fitting error can be bounded. Here, we run our algorithm for different types of outliers by adjusting the noise level of outliers $\sigma_1 \in [0.1, 1.2]$, with



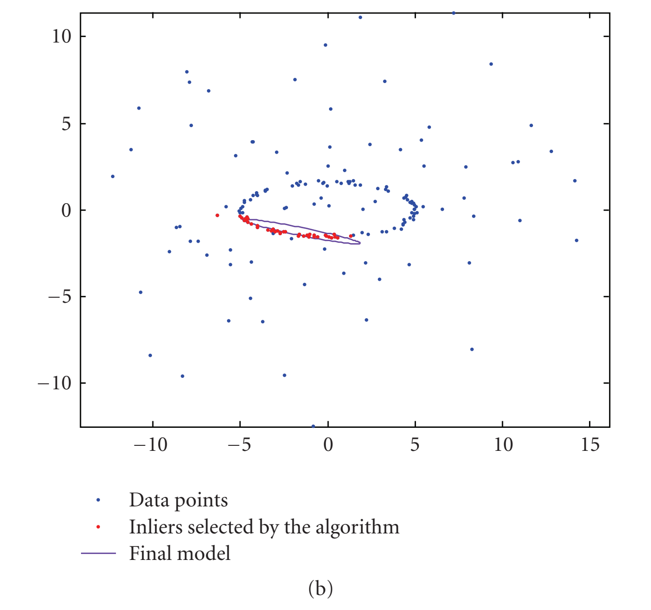


FIGURE 7: Typical fitting results for the RANSAC-EIS-Metropolis Algorithm. (a): a successful fit after 500 iterations, with inlier noise level $\sigma_0 = 0.02$. (b): an unsuccessful fit after 500 iterations, with inlier noise level $\sigma_0 = 0.08$.

$N = 120$, $M = 90$, $\sigma_0 = 0.1$, and the true model for the ellipse unchanged. The interesting results are shown in Figure 8, where for outliers closer to the inliers as well as outliers distant from the inliers, our algorithm performs consistently well, with fitting error tightly bounded below a low level. This shows the strong robustness of our scheme.

5.6. Generalization to 3-D: Ellipsoid Fitting. The basic set-up for the ellipsoid fitting simulation is as follows: for inliers, $N = 300$ and $\sigma_0 = 0.1$; for outliers, $M = 50$ and $\sigma_1 = 5$; the ellipsoid takes the standard position, with semiaxis lengths

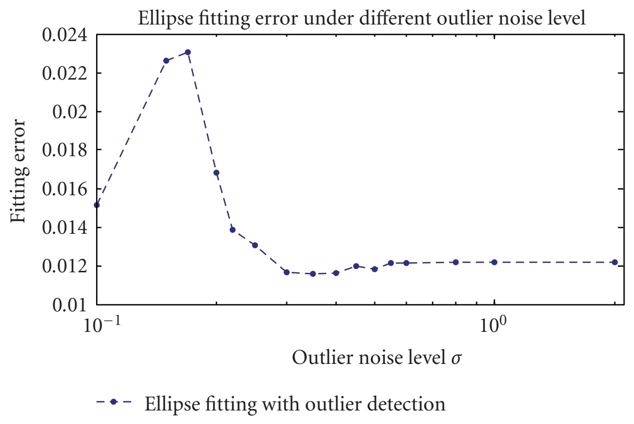


FIGURE 8: The performance of our two-stage algorithm under a variety of outlier noise levels. Our algorithm performs consistently well for outliers with very different magnitudes.

$a = 5$, $b = 4$, and $c = 3$. The outlier detection results are as shown in Figure 9.

6. Conclusion

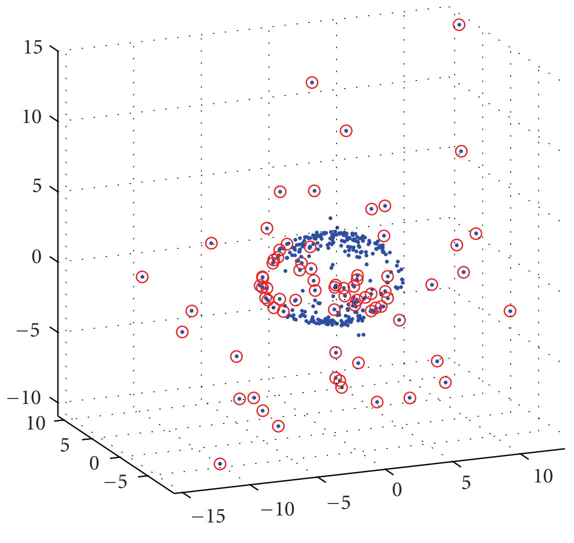
Proximity-based outlier detection algorithms are good for situations in which outliers are wildly erroneous and large in number. However, such algorithms work poorly for outliers that are close to the inliers, even though these outliers are obviously not consistent with the model; on the contrary, model-based algorithms are very good at detecting a small portion of outliers that are not consistent with the model. Yet, if the percentage of outliers is high and the number of parameters for the model is large, the implementation of model-based algorithms can be costly. In the problem of curve and surface fitting with many outliers, by combining these two types of algorithms, we have found a promising hybrid method that performs robustly with high accuracy for a variety of types and numbers of outliers. The hybrid algorithm can effectively reduce the total number of iterations required for fitting the model to the data (selected inliers). Consequently, although the proximity based step requires computation of all pairwise distances among all the data points, when the model is so complicated that fitting the model to the data becomes the critical step, the hybrid algorithm can be more efficient, compared to other outlier elimination algorithms that require large numbers of model refittings.

Appendices

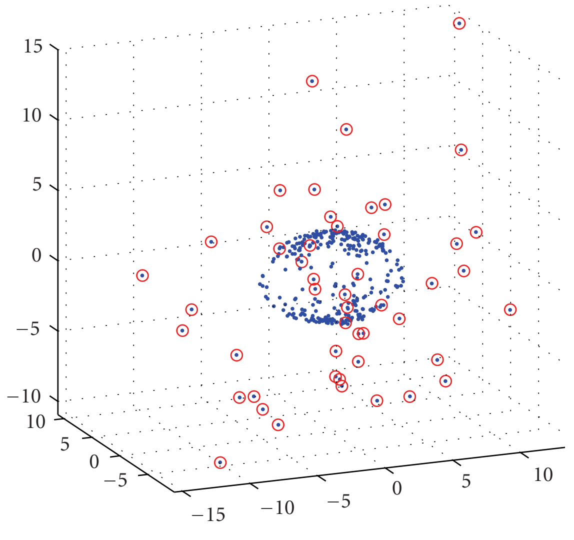
A. Derivation of the Bound

Proof. Define an event A as

$$A = \{\forall \text{ observation}, \exists \text{ a neighbor within distance } \delta \text{ along the curve direction}\}. \tag{A.1}$$



(a)



(b)

FIGURE 9: Outlier detection results for ellipsoid fitting. Similarly to the ellipse fitting case, the proximity-based algorithm eliminates most of the distant outliers, yet misclassifies some inliers as outliers, with several nearby outliers missed, as illustrated in (a). These errors are corrected by the second stage model-based algorithm, as shown in (b).

Then by the union bound, and the inequality $(1-x)^n \le e^{-nx}$, we have

$$\begin{aligned}\mathbb{P}(A) &= 1-\mathbb{P}(A^c)\\ &\ge 1-N\left(1-\frac{\delta}{L}\right)^{N-1}\\ &= 1-N(1-\alpha)^{N-1}\\ &\ge 1-\frac{Ne^{-\alpha N}}{1-\alpha}.\end{aligned}\tag{A.2}$$

Therefore, the probability of A is lower bounded by $1-\epsilon$, where $\epsilon = Ne^{-\alpha N}/(1-\alpha)$. □

B. Derivation of the Estimates (5) and (6)

Suppose the inliers are independently sampled from a one-dimensional manifold of length L with probability density $\rho(x)$. If N observations are sampled, then for any observation P located at x_0, the probability that its nearest neighbor on the manifold falls outside of the ball $B_y(x_0)$ of radius y is

$$\begin{aligned}&\mathbb{P}(\text{Distance to nearest neighbor of P} > y)\\ &\quad = \left(1-\int_{x_0-y}^{x_0+y}\rho(x)\mathrm{d}x\right)^{N-1}.\end{aligned}\tag{B.3}$$

Then, the probability that a randomly chosen data point has a nearest neighbor within the radius of y, defined as $F(y)$, is given by

$$F(y) = \int_0^L \rho(x_0)\left(1-\left(1-\int_{x_0-y}^{x_0+y}\rho(x)\mathrm{d}x\right)^{N-1}\right)\mathrm{d}x_0. \tag{B.4}$$

Applying the fact that the integral of the density on the entire manifold is one, (B.4) can be simplified into

$$F(y) = 1-\int_0^L \rho(x_0)\left(1-\int_{x_0-y}^{x_0+y}\rho(x)\mathrm{d}x\right)^{N-1}\mathrm{d}x_0. \tag{B.5}$$

If we assume that y is sufficiently small, then we may make a series of approximations

$$\begin{aligned}F(y) &\approx 1-\int_0^L \rho(x_0)(1-2y\rho(x_0))^{N-1}\mathrm{d}x_0\\ &\approx 1-\int_0^L \rho(x_0)(1-2(N-1)y\rho(x_0))\mathrm{d}x_0\\ &= 2(N-1)y\int_0^L \rho^2(x_0)\mathrm{d}x_0.\end{aligned}\tag{B.6}$$

Notice that $\int_0^L \rho(x_0)\mathrm{d}x_0$ is directly related to the volume of the manifold. If we assume that the observations are uniformly distributed, then $\rho(x_0) = 1/L$, and hence $\int_0^L \rho(x_0)\mathrm{d}x_0 = 1/L$. In this case we have

$$L \approx \frac{2(N-1)y}{F(y)}. \tag{B.7}$$

The method above can be directly generalized to the case of a d-dimensional manifold $\mathcal{M}$, as follows:

$$\begin{aligned}&\mathbb{P}(\text{Nearest neighbor distance} \le y)\\ &\quad \approx (N-1)\mathcal{V}\left(B_y^{(d)}\right)\int_{\mathcal{M}}\rho^2(\mathbf{x})\mathrm{d}\mathbf{x},\end{aligned}\tag{B.8}$$

where $\mathcal{V}(B_y^{(d)})$ denotes the volume of a d-dimensional ball with radius y.

Acknowledgments

This research was supported in part by the Office of Naval Research Grants N00014-09-1-0342 and N00014-07-1-0555, in part by the US Army Research Office under Grant W911NF-07-1-0185, and in part by the National Science Foundation under Grant CNS-09-05398 and Science & Technology Center Grant CCF-09-39370.

References

[1] F. L. Bookstein, "Fitting conic sections to scattered data," *Computer Graphics and Image Processing*, vol. 9, no. 1, pp. 56–71, 1979.

[2] G. Taubin, "Improved algorithm for algebraic curve and surface fitting," in *Proceedings of the IEEE 4th International Conference on Computer Vision*, pp. 658–665, Berlin, Germany, 1993.

[3] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 476–480, 1999.

[4] S. J. Ahn, W. Rauh, and H.-J. Warnecke, "Least-squares orthogonal distances fitting of circle, sphere, elipse, hyperbola, and parabola," *Pattern Recognition*, vol. 34, no. 12, pp. 2283–2303, 2001.

[5] S. J. Ahn, W. Rauh, H. S. Cho, and H.-J. Warnecke, "Orthogonal distance fitting of implicit curves and surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 620–638, 2002.

[6] A. Atieg and G. A. Watson, "A class of methods for fitting a curve or surface to data by minimizing the sum of squares of orthogonal distances," *Journal of Computational and Applied Mathematics*, vol. 158, no. 2, pp. 277–296, 2003.

[7] I. Al-Subaihi and G. A. Watson, "The use of the l_1 and l_∞ norms in fitting parametric curves and surfaces to data," *Applied Numerical Analysis & Computational Mathematics*, vol. 1, no. 2, pp. 363–376, 2004.

[8] I. A. Al-Subaihi, "An algorithm for fitting circular arcs to data using the l_1 norm," *Numerical Algorithms*, vol. 47, no. 1, pp. 1–14, 2008.

[9] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[10] S. Byers and A. E. Raftery, "Nearest-neighbor clutter removal for estimating features in spatial point processes," *Journal of the American Statistical Association*, vol. 93, no. 442, pp. 577–584, 1998.

[11] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, vol. 29, no. 2, pp. 427–438, 2000.

[12] E. M. Knorr and R. T. Ng, "Algorithms for mining distance-based outliers in large datasets," in *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB '98)*, pp. 392–403, New York, NY, USA, 1998.

[13] V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Outlier detection using k-nearest neighbour graph," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, pp. 430–433, Cambridge, UK, August 2004.

[14] Y. Kou, C.-T. Lu, and R. F. Dos Santos Jr., "Spatial outlier detection: a graph-based approach," in *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI '07)*, vol. 1, pp. 281–288, IEEE Computer Society, Washington, DC, USA, 2007.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[16] D. Li, D. Winfield, and D. J. Parkhurst, "Starburst: a hybrid algorithm for video-based eye tracking combining feature-based and modelbased approaches," in *Proceedings of the Computer Vision and Pattern Recognition Workshops*, p. 79, IEEE Computer Society, San Diego, Calif, USA, 2005.

[17] L. Fan and T. Pylvänäinen, "Robust scale estimation from ensemble inlier sets for random sample consensus methods," in *Proceedings of the 10th European Conference on Computer Vision (ECCV '08)*, vol. 5304 of *Lecture Notes in Computer Science*, pp. 182–195, Springer, Berlin, Germany, 2008.

[18] P. L. Rosin, "Ellipse fitting by accumulating five-point fits," *Pattern Recognition Letters*, vol. 14, no. 8, pp. 661–669, 1993.

[19] B. Mohar, "The Laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*, vol. 2, pp. 871–898, Wiley, New York, NY, USA, 1991.