Chapter 3

Finding Breaking Curves on 3D Range Data

In this chapter, we present a recursive least squares technique for extracting the *breaking curve* of $3D$ range open surfaces. Unlike differential operators-based methods, the algorithm we propose is robust to noise and is applied to unorganized point sets. No assumptions such as smoothness and/or continuity on the boundary's shape are performed. The method we present deals with large amount of data under a low computational cost, since no local computation is performed. A global approach is given to the technique in order to make it more robust, faster and simpler than individual point plus neighbors approaches.

3.1 Introduction

Boundary extraction is associated to many problems related with noisy $3D$ range data surfaces: segmentation [61, 40], object recognition, perceptual organization [20], applications to archaeology pottery reconstruction [26, 29], sherds classification, the $3D$ puzzle problem, etc. These applications deal usually with broken pieces and patches which can be characterized by their *breaking curves* (see fig. 3.1). This type of curves can characterize the $3D$ spatial organization of a set of broken sherds in order to reconstruct the original object. Breaking curves are a particular case of 3D edges; they are the boundaries corresponding to a 2D surface embedded in a 3D space. Actually, the points belonging to a 2D surface are considered to be *edge points* since there is a discontinuity in at least one direction of the first derivatives ($3D$ gradient). Much work has been done on 3D edge detection through differential operators [48, 118, 11]. The main drawback is that neighboring-based operations are highly expensive in terms of computational cost. Other local approaches are either based on computations that require a previous data ordering -triangular meshes- [61, 49, 36] or *boundary following*-like algorithms [74]. Techniques based on local computations -such as partial differential equations- suffer from extreme sensitiveness to noise.

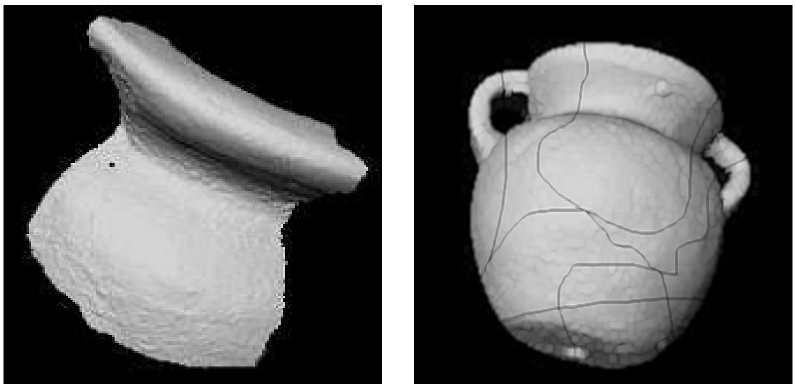


Figure 3.1: Broken sherd and its original pot.

3.1.1 Contribution

We propose a new method that is based on global computations, and, which : *i)* is fast in very large data sets, *ii)* is robust to noise, and *iii)* does not need data to be organized.

This method is based on a recursive algorithm that starts fitting a plane to the surface and in each iteration increases the number of planes by splitting the previous estimated ones. In each estimation, the boundaries for each plane are computed. The boundaries of each sub-portion fitted by a plane must be consistent with the whole data set. This means that only the plane boundaries points consistent with the surface boundaries are taken as "true" boundary points. The algorithm provides a measure, which is based on the spatial distribution of data, for checking the boundary condition. This procedure corresponds to a tree-like structure, where each level contains information on the sherd's boundary at a certain scale. Once, the algorithm is at the lowest level of the tree-structure, all the *consistent* boundary points are collected from all the branches and leaves. These points are the sherd's *boundary points.* As it is shown in the experiments, the more levels are investigated, much more accuracy is obtained for describing the breaking curve. Issues such as: "how many levels are needed?", and "when it is necessary to stop the splitting?" are answered as well. Moreover, from a computational point of view, we show that the algorithm is initialization independent. The manner it is formulated avoids implicitly ill-conditioning problems without being forced to add ad hoc numerical treatments.

3.1.2 Outline

The chapter, first, introduces the mathematical framework where the algorithm is based on. The main idea, based on a description for approximating surfaces through planar patches, is presented. In this framework, we describe which conditions must be satisfied by the boundary points of a planar surface. Moreover, the criteria for deciding how many planes are sufficient when approximating a surface is developed in terms of the noise of the observed data. Another important matter is the manner data is split up. This is also studied by using the obtained estimates for plane fitting, which implies computation saving. In section 3, the algorithm is presented. A function written in pseudo-code is described in order to show the algorithm's recursive nature. In section 4, some experiments are shown. The purpose is to show the behavior of this technique when dealing with high curvature regions, where local techniques yield misleading results. The computational cost is studied from both empirical and theoretical points of view. Finally, in section 5, we present the conclusions.

3.2 Background

In this section, we introduce the mathematical framework and the main geometrical idea behind the algorithm. Before beginning with the formulation, an issue to be considered is that a $2D$ surface can be approximated by $2D$ planar patches. Ideally, an infinite number of these planar patches would reconstruct the surface exactly.

3.2.1 Fitting a Plane to a Distribution of Points

The first and easiest sort of surface to start with is a planar surface described by a few number of points. This type of surfaces can be represented by a specific set of coordinate orthogonal axes adapted to the points spatial distribution. These are obtained by means of a linear regression that fits a $2D$ plane minimizing the orthogonal distance to the mentioned plane. A plane, actually, can be described by just two degrees of freedom that locate any point belonging to it. These two degrees of freedom are scalar values that measure the distance of a point along each of the axes.

The estimation of the principal axes is performing through Principal Component Analysis (hereafter PCA) [7]. The result of applying PCA on set of $3D$ points is a set of 3 unitary vectors and 3 scalar values. The unitary vectors are known as *eigenvectors*, and the scalar values as *eigenvalues*. The eigenvalues give a notion of the importance of a specific axis with respect to the others. This importance measure is actually the variance -in statistical terms- of the data along each axis (see fig.3.2). This means that an axis with large variance associated has the data distributed in a larger portion of space, than another axis with lower variance associated. In other words, when it comes to fit a plane to a planar distribution of points, there is an axis with negligible variance (fig.3.2(b)), which determines the noise of the planar distribution. The larger is the amount of variance in that direction, the lower is the

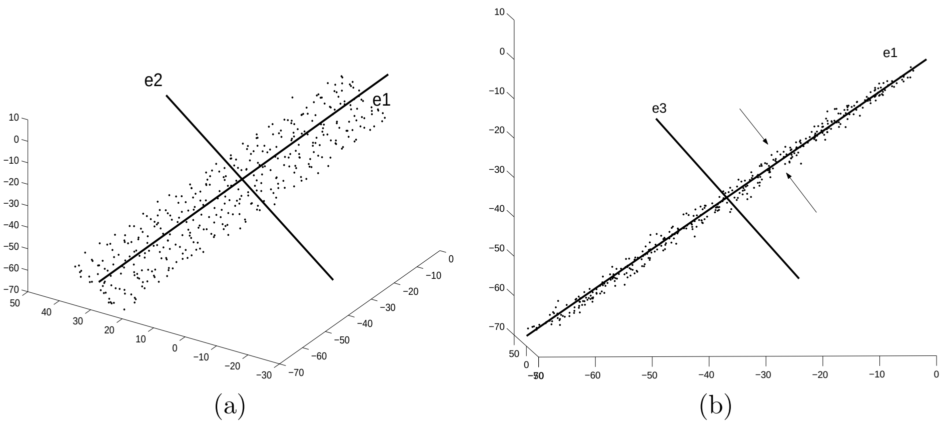


Figure 3.2: Two different views of a planar surface and its principal axes: View (a) shows the two principal axes of bigger variance $e1$ and $e2$, and, view (b) shows the variance with respect to the lowest variance axis $e3$.

likelihood of the point distribution to be a plane. PCA algorithm is coordinate free, which means that results are adapted to the nature of the data distribution but to the frame of reference where data is represented. This makes the technique independent of the absolute position, which is quite useful when dealing with pottery (pieces in general) scans.

3.2.2 Bounding Box Approach

Let us analyze some useful consequences of computing the principal axes of a noisy planar distribution of points. Firstly, the study of the projections along each of the two principal axes gives a manner of start investigating the surface dimensions. We can consider as a rough approximation the surface's shape to be a rectangle. Under this coarse way of studying the surface's dimensions, we can notice that the limits are defined in each of the two principal directions, i.e., the orthogonal projections onto each of the eigenaxis (fig. 3.3). This approximation permits finding a bounding box to the planar surface. There are at least four interesting points for a general contour shape of a planar distribution of points. Each of these four points are the contact points with the bounding box, and each one corresponds to be the limit point projection in each eigenaxis direction. It is interesting to note that the fewer number of points are in the planar distribution, the simpler is the object's shape and therefore, better approximated by the mentioned bounding box. This reasoning plays a crucial role when attempting to describe a complex surface by means of assembling simpler structures.

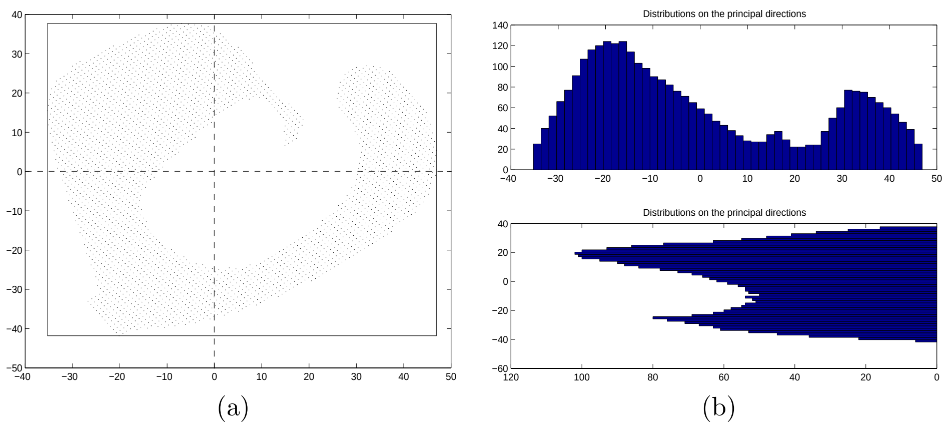


Figure 3.3: (a)Planar surface approximation and principal directions. This distribution can be bounded by a rectangle defined by (b) the orthogonal projections onto each of the two principal axis.

3.2.3 Approximation through Planar Surfaces

The role of planar surfaces fitting is emphasized since it is the basis of our approach. Representing a surface through small planes can assist a manner of extracting the main features, which can be used in a posterior process for recognition and perceptual organization. Given a certain measure of tolerance for the error between the point distribution surface and its approximation through planes, one can notice that regions with higher curvature need more planes to be adjusted, and regions with very low curvature may just need few planes to describe its complexity. Locally, the criteria for deciding how many planes are sufficient for approximating a surface can be the error measure provided by the mentioned negligible third eigenaxis (the one with lowest variance). In other words, if the error obtained from fitting a surface through just one plane is bigger than a certain tolerance, then a planar patch is not sufficient for describing the surface. Therefore, splitting into more than one planar patch will be necessary.

In order to see how the surface is approximated through planar patches, we introduce the formulation where the algorithm is based on. Consider a $2D$ noisy surface defined by a distribution of $3D$ points $\mathcal{P} = \{\vec{p}_1, \ldots, \vec{p}_N\}$. As a first approximation, we like to find a plane defined by 2 orthogonal vectors $W = \{\vec{e}_1, \vec{e}_2\}$ and a point $\vec{\mu}$ that defines the plane's location. The optimal plane parameters (W, μ) in a least squares sense, implies finding (W, μ) that minimize the orthogonal distance from the $3D$ points $\mathcal{P}$ to the plane,i.e., the equation to be minimized w.r.t. (W, μ) is:

$$\mathcal{E} = \sum_{n=1}^{N} \left[|\vec{p}_n - \mu|^2 - |W'(\vec{p}_n - \mu)|^2 \right] \tag{3.1}$$

There is a constraint connected to the unitary condition on W. It can be shown

[7] that finding W under these constraints, is equivalent to compute the eigenvector associated to the larger eigenvalues of the covariance matrix:

$$\Sigma = \frac{1}{N-1}\sum_{n=1}^{N}(\vec{p}_n - \mu)(\vec{p}_n - \mu)' \tag{3.2}$$

The point μ is, actually, the sample mean of $\mathcal{P}$:

$$\mu = \frac{1}{N}\sum_{n=1}^{N}\vec{p}_n \tag{3.3}$$

As a matter of fact, the lowest variance associated eigenvector is related with the error (3.1) by a N factor. Therefore, the computation of the covariance matrix gives us all the necessary information about the plane fitting: the plane parameters (W, μ) and even the error performing the fitting.

3.2.4 Recursive Splitting

When approximating a $2D$ surface through a plane, the error might be higher than a certain degree of tolerance. In that case, it is necessary performing a more complex modelling. After a first approximation through one plane, the second step is to consider two planar surfaces to fit the original distribution of points. Typically, when a simple model does not fit well data is because the data has to be explained through more complex models, or, through a combination of simple sub-models, which is more practical. In the case we consider, given the geometrical nature of the observed data, that the point distribution can be approximated by higher number of planar patches (a combination of simple models).

Given the principal directions $W = \{\vec{e}_1, \vec{e}_2\}$ and the sample mean μ there are some ways of dividing data into sub-portions. A manner of dividing data into two sets is along on of the principal direction, for instance, the one with the largest variance $\vec{e}_1$. The splitting in this case would be selecting all those points that are left to the sample mean in the $\vec{e}_1$ and the other set all those which are right to μ. More explicitly, one sub-set $\mathcal{P}_1^1$, will consist of points whose projection onto $\vec{e}_1$ is :

$$\vec{e}_1'(\vec{p}_{n_1} - \mu) \leq 0$$

and another set $\mathcal{P}_2^1$, will be formed by those points whose projection is:

$$\vec{e}_1'(\vec{p}_{n_2} - \mu) > 0$$

The same reasoning can be applied to the $\vec{e}_2$ direction. However for the sake of notation, we now just consider a division into two sub-sets. The notation that has been used for the two sub-sets is described as follows: for $\mathcal{P}_1^1$, the upper-index means that is, in this case, the first division applied to the data set $\mathcal{P}$. The sub-index is sub-portion index, in this specific case we have two sub-sets : $\mathcal{P}_1^1$ and $\mathcal{P}_2^1$.

The point here is that the same technique can be applied to each of the sub-sets. Therefore, what we obtain finally is a tree structure of sub-sets decompositions:

$$\begin{tikzcd}
 & & & \mathcal{P} \arrow[dll] \arrow[drr] & & & \\
 & \mathcal{P}^1_1 \arrow[dl] \arrow[dr] & & & & \mathcal{P}^1_2 \arrow[dl] \arrow[dr] & \\
\vdots & & \vdots & & \vdots & & \vdots
\end{tikzcd}$$

Branch Pruning: Bayesian approach

Regarding the idea of the certain degree of tolerance for splitting, there will be nodes that do not need to be split up. The purpose of this is to avoid over-fitting and the confusion noise and geometrical properties of a surface such as curvature. There are two important issues to be considered: one one hand, the *noise* in data originated from observations (scanning, etc...) and, on the other, the *error* due to model assumptions. In the framework we are dealing with, the error due to the model comes from fitting a plane to a region, and regions with high curvature yield a high error measure in the estimates. In order to deal with the entanglement noise/error, we apply to our technique the *Minimum Description Length* [83], where the aim is to evaluate the plausibility of different alternative models explaining the same observations (distribution of points). This evaluation is performed on the Occam's Razor Principle which states that one should not make more assumptions than the minimum needed. In other words, when a specific region is "sufficiently" flat, the algorithm decides not to split on that area.

In this Bayesian framework, "sufficiently" means a trade off between the error measure in the estimates and the complexity of the model. It is considered more complex a model that describes data with many planes, than just one plane, since many more degrees of freedom (number of parameters) are involved. This model selection criterion translates into the introduction of a penalty term in equation (3.1). This penalty term comes from approximating the posterior distribution for a d-dimensional parameter set $\hat{\theta}$ by a Gaussian [33], so that the evidence for a data set D under a set of hypotheses $\{h_i\}$ is written as follows:

$$P(D|h_i) \approx P(D|\hat{\theta}, h_i)P(\hat{\theta}|h_i)(2\pi)^{d/2}|H|^{-1/2}$$

where $H = \nabla\nabla \log P(D|\hat{\theta}, h_i)$ is a Hessian matrix which measures how peaked the posterior is around the Maximum a Posteriori value. In other words, this measure tells us about the uncertainty of the estimates $\hat{\theta}$. Moreover, Rissanen [83], assuming uncorrelated parameters and identically distributed over N observations, approximates the Hessian $H = NH_0$-where H_0 is the Hessian for a single observation. Under this approximation the evidence for a data set D is:

$$\begin{aligned}
P(D|h_i) &\approx P(D|\hat{\theta}, h_i)P(\hat{\theta}|h_i)(2\pi)^{d/2}|NH_0|^{-1/2} \\
&= P(D|\hat{\theta}, h_i)P(\hat{\theta}|h_i)(2\pi)^{d/2}N^{-d/2}|H_0|^{-1/2}
\end{aligned}$$

Therefore, taking the negative logarithms of the previous expression, equation (3.1) has now an additional term:

$$\mathcal{E} = \frac{1}{2\sigma^2}\sum_{n=1}^{N}\left[|\vec{p}_n - \mu|^2 - |W'(\vec{p}_n - \mu)|^2\right] + \frac{d}{2}\log N \tag{3.4}$$

where σ^2 models the noise variance in the observed distribution of points. It is assumed, that data is affected by identically independent distributed Gaussian noise. Note that embedding this problem into a Bayesian framework allows us to both model noise and distinguishing it from error in the estimates at the same time.

Splitting Condition: Lower Bounds approximation

Consider a distribution of points fitted by a plane described by the parameters θ_{single}. The same distribution can be split up into many subsets fitted by a more complex model based on many planes. Let θ_{many} be the collection of parameters that describe this second model. The idea of introducing a Bayesian formulation into this problem has the purpose of dealing with a tool that allows making a decision between the model θ_{single} and θ_{many}. However, the fact of deciding between one of these models implies a previous estimation of both models as well as computing the error in the estimates through (3.4). From a computational point of view a relevant question arises: Is there any way of performing such a decision without a previous computation of the second model parameters θ_{many}?

The way the algorithm is presented allows an incremental model selection performance at each level of the tree. However, adding an extra computational measurement for model comparison would lead to an excessive computational cost. This can be overcome through some approximations that will automatically determine a decision from the data itself. Let $\mathcal{E}_1$ be the negative log-evidence approximated around θ_{single}:

$$\mathcal{E}_1 = -\log P(D|\theta_{\text{single}}) + \frac{d}{2}\log N \tag{3.5}$$

Following the same idea, let $\mathcal{E}_2$ be:

$$\mathcal{E}_2 = -\sum_{m=1}^{M}\log P(D|\theta^{\mathbf{m}}_{\text{many}}) + \sum_{m=1}^{M}\frac{d}{2}\log\left[\frac{N}{M}\right] \tag{3.6}$$

where it has been assumed that the M sub-models are independent one from each other (hard clustering) and they contain approximately the same number of points N/M. The number of dimensions for each sub-model is d as well, since we are fitting planes. The worst case scenario happens when fitting through the second model θ_{many} yields a noiseless estimation, i.e.,

$$-\sum_{m=1}^{M}\log P(D|\theta^{\mathbf{m}}_{\text{many}}) \approx 0$$

which might correspond to an over-fitting situation, for instance, when each subset is formed by triangles.

Under this approximation, the splitting condition can be written as follows:

$$\begin{aligned}
\mathcal{E}_0 + \frac{d}{2}\log N &> \frac{dM}{2}\left[\log N - \log M\right] \approx \frac{dM}{2}\log N, \\
\text{with } N >> M & \\
\mathcal{E}_0 &> \frac{d(M-1)}{2}\log N \qquad (3.7)
\end{aligned}$$

Taking into account that a plane is defined by one point μ and one unit vector $\vec{n}$, the value for the number of degrees of freedom for a plane is $d = 3 + 2 = 5$ in case of 2-D planes. The number of planes resulting from splitting will be assigned to M. On the other hand, the noise is assumed to be gaussian with zero mean and σ^2 variance. Therefore, the error $\mathcal{E}_0$ is obtained from the negative logarithm of a gaussian distribution, i.e.,

$$\mathcal{E}_0 = \frac{1}{2\sigma^2}\sum_{n=1}^{N}\left[|\vec{p}_n - \mu|^2 - |W'(\vec{p}_n - \mu)|^2\right] + \frac{3N}{2}\log(2\pi\sigma^2) \qquad (3.8)$$

Granularity

The splitting condition has been derived from the geometrical specific properties of a given distribution of points that represent a 2-dimensional surface. It has been shown that these geometrical features are entangled with noise. The assumptions made for modelling noise lead to a manner of treating the fitting either with one or many planes on a specific area. Once the splitting condition (equation (3.7)) is not satisfied for a specific region, the algorithm is said to be at one of the leaves of the tree. This means that the applied plane model for that region is sufficiently *flat* in order to explain the geometrical distribution of points. In this case, it is time for computing the boundary points of that specific area.

As introduced in section 3.2.1, from the computation of the principal directions of a plane fitted to a distribution of points yields 4 contact points that define a bounding-box where the distribution is contained. Even though there are many ways of computing the contours of a planar distribution of points in 2 dimensions, there are two important computational issues to be considered:

1. Take advantage of computations already performed. The plane's estimated parameters can be used for estimating the points distribution density.

2. Avoid extra-checking. Given that many sub-regions are analyzed through splitting and plane fitting, the contour detection at one of these sub-regions has to be simple and fast.

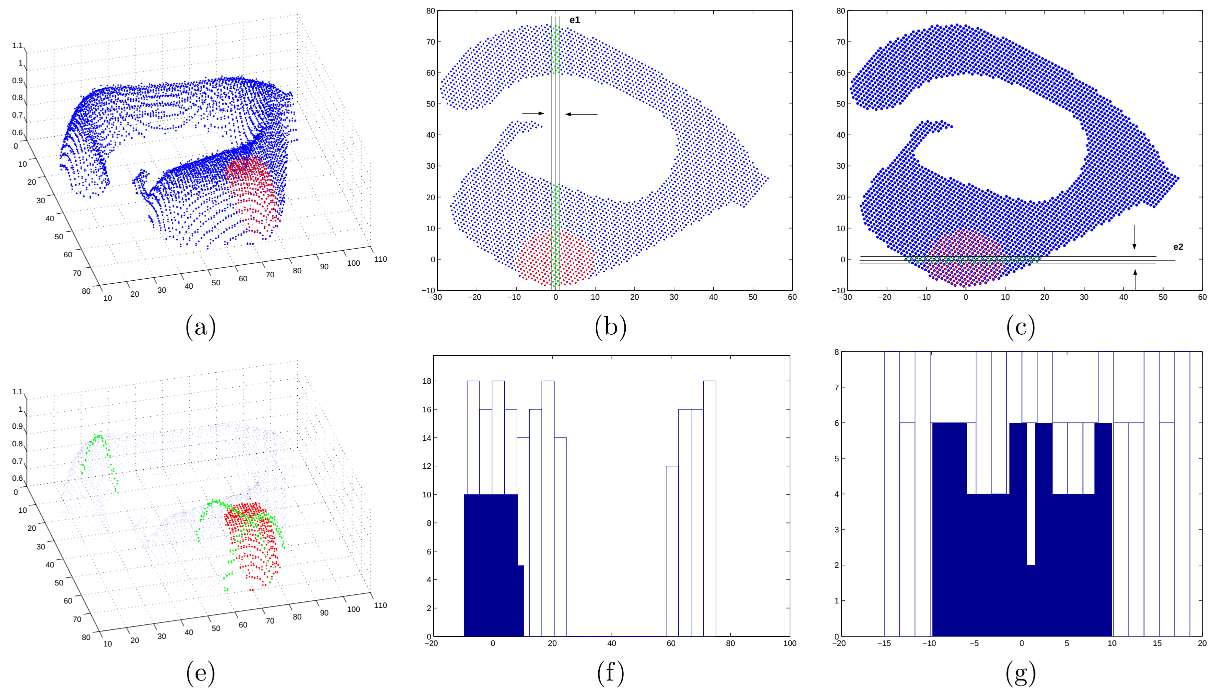


Figure 3.4: Boundary analysis of a sub-set. Projections onto the two principal eigenvectors in (b) and (c). The histograms (f) and (g) of the projections along one of the two main directions define the representation for finding boundary points. Histogram (f) in blue detects a boundary point to the left. Histogram (g) in blue cannot determine any boundary point since it is completely surrounded by the whole data set $\mathcal{P}$ histogram.

Studying a few number of points in each sub-region leads to a sufficiently large resulting number of contour points after putting together all the estimates. The analysis of contour checking is based on one direction only. In this case, we consider points to be checked out as boundary points, all those within a specific orthogonal tolerance ϵ to the principal axis of a certain sub-region. Since the distribution of points is not continuous, the algorithm has to accept a certain degree of tolerance when it comes to study projections along one selected direction. The idea of *granularity* gives us a manner of introducing a measure of amplitude ϵ. Moreover, for the sake of computational cost saving, it is preferable to project just a few number of points, rather than the whole data set.

In this framework, we define *density* as the number of points N_0 in a certain area $A = \pi R_0$:

$$\rho = \frac{N_0}{\pi R_0^2} = \frac{2}{\pi \epsilon^2} \tag{3.9}$$

where R_0 is the radius of a $2D$-sphere that approximates the point distribution. Assuming homogeneous isotropic density, we take ϵ as the average distance between

2 points.

$$\epsilon = R_0 \sqrt{\frac{2}{N_0}} \tag{3.10}$$

The radius R_0 can be obtained from computing the geometrical mean of the bounding box dimensions, i.e., in each of the two principal directions compute the distance between the extremal point projections (fig. 3.3(a)). This defines some width λ_1 and some height λ_2 for a given bounding-box. Therefore, we can write $R_0 = \sqrt{\lambda_1 \lambda_2}$.

Figure 3.4 shows the way the boundary points are determined. First, select one of the two principal directions, for instance $\vec{e}_1$. Subsequently, project onto $\vec{e}_1$ only the points that satisfy:

$$\vec{p}_n \text{ , such that, } |\vec{e}_2'(\vec{p}_n - \mu)| < \epsilon \tag{3.11}$$

This ϵ parameter makes possible to deal with complex contour shapes. Figure 3.4 shows geometrically the idea of the condition expressed in eq. (3.11). From these projections, the only points we can select are those which belong to the data sub-set (the points inside the red patch in figure 3.4 (a)), the rest of points belonging to the whole data set are just to check if the *boundary* point belonging to the subset is a "true" *boundary* point. Formally, the condition to be satisfied by a boundary point of a sub-set $\mathcal{P}_n^L$ is:

Definition 3.2.1 *$\vec{p}_k \in \mathcal{P}_n^L$ is a boundary point if*

1. *$\vec{e}_1'(\vec{p}_k - \mu) \geq \vec{e}_1'(\vec{p}_i - \mu) \forall i$ with $\vec{p}_i \in \mathcal{P}$ or*
2. *$\vec{e}_1'(\vec{p}_k - \mu) \leq \vec{e}_1'(\vec{p}_i - \mu) \forall i$ with $\vec{p}_i \in \mathcal{P}$*

where $\mathcal{P}$ is the whole set of points.

Note that the symbols "$\geq$" and "$\leq$" are taking into account the fact that a point belonging to the sub-set also belongs to $\mathcal{P}$. Also notice that no point outside the subset $\mathcal{P}_n^L$ is considered, at this step, to be a boundary point since we are just dealing with projections, there is no security of finding a "true" boundary point which does not belong to $\mathcal{P}_n^L$ from this projection.

3.3 The Algorithm

This section summarizes the procedure for finding *breaking curves.* The recursive function that computes the boundary points has for arguments: the set of $3D$ points $\mathcal{P}$ and the subset of points corresponding to a specific surface patch $\mathcal{P}_{\text{subset}}$. For a given data set $\mathcal{P} = \{\vec{p}_1, \ldots, \vec{p}_N\}$, the initial considered sub-set is the same data set $\mathcal{P}$. The function is described in table 3.1.

The function calls itself twice, according to the split along the first principal direction $\vec{e}_1$. However, there are many possible combinations, such as splitting according

Table 3.1: Algorithm's pseudo-code

```
function b = computeBoundaryPoints($\mathcal{P}$,$\mathcal{P}_{\text{subset}}$)
  • Find the principal components ($\vec{e}_1, \vec{e}_2$) and compute the sample mean $\mu$ for the
    set of $N$ points $\mathcal{P}_{\text{subset}}$.
  • Compute $\epsilon$ amplitude from the plane parameters:
        $\epsilon = \sqrt{\frac{2\lambda_1\lambda_2}{N}}$
  • Compute the boundary points $b_0$ for $\mathcal{P}_{\text{subset}}$ the using $\mathcal{P}$, $\epsilon$ and the conditions
    eq. (3.11) and definition 3.2.1.
  • Stop-splitting condition:
    – If the error $\mathcal{E}_0$ (eq. 3.8) from fitting the sub-set $\mathcal{P}_{\text{subset}}$ is:
          $\mathcal{E}_0 > \frac{5}{2}\log N$
      then split
```

$$\begin{tikzcd}
 & \mathcal{P}_{\text{subset}} \arrow[dl] \arrow[dr] & \\
\mathcal{P}_{\text{subset}_1} & & \mathcal{P}_{\text{subset}_2}
\end{tikzcd}$$

```
        * $b_1$ = computeBoundaryPoints($\mathcal{P}$,$\mathcal{P}_{\text{subset}_1}$)
        * $b_2$ = computeBoundaryPoints($\mathcal{P}$,$\mathcal{P}_{\text{subset}_2}$)
      and concatenate the resulting boundary points into $b = \text{cat}(b_0, b_1, b_2)$.
      Therefore, return $b$.
    – else return $b = b_0$.
```

to the first and the second principal direction $(\vec{e}_1, \vec{e}_2)$, which implies calling four times the function computeBoundaryPoints, and therefore, spanning a tree of four branches each time. We have described only a partition into two subsets in order to follow the previous section's description, and for sake of notation. This leads to the stoping condition: $\mathcal{E}_0 > \frac{5}{2}\log N$. Note that this condition and the amplitude parameter ϵ are computed from the plane estimates, saving computational cost.

3.4 Experiments

In this section, we present some results obtained from real archaeological pieces. The first point we like to emphasize is the fact that three of them have high curvature regions. When applying differential operators to this sort of surfaces with high curvature regions we can obtain misleading results due to the entanglement noise-geometrical features. The algorithm we have presented clearly overcomes this sort of regions, since data has been treated under a global and Bayesian approach.

Actually, the role played by the stoping condition in equation 3.7 is to make a clear distinction between noise and curvature. This is carried out through considering the amount of data $\frac{5}{2}\log N$ on one hand, and, on the other the actual error $\mathcal{E}_0$ from fitting. A direct consequence of this Bayesian treatment is that curvature is well defined when there is a sufficiently large amount of data $\log N$. More accurately, we can see an example considering an average error per point $\bar{\mathcal{E}}$ related to the error $\mathcal{E}_0$ by $\mathcal{E}_0 = N\bar{\mathcal{E}}$. The stoping condition is written now as follows:

$$\bar{\mathcal{E}} > \frac{5}{2}\frac{\log N}{N}$$

From this comparison, consider we like to study two different situations: on one hand a data set with a few number of points, and another with a large number of points -this example is illustrated in figure 3.5. Consider both sets, with the same average error $\bar{\mathcal{E}}$. Since $N \to \infty$ implies $\frac{\log N}{N} \to 0$, therefore, the algorithm considers a more complex model is needed to explain the observations, i.e., the presence of curvature is pointed out. On the other hand, for the same average error $\bar{\mathcal{E}}$, dealing with a small number of points in the data set will be taken as noise. Thus, the definition of curvature is more accurate the more points we have. Of course, this is a natural and obvious conclusion from a scale-space point of view, however, the interesting point here is the fact that the Bayesian treatment yields a manner of automatically deciding when there is noise and when we are facing a high curvature region. No ad hoc thresholds are given to the process.

3.4.1 Discrete Distribution Effects

With the aim of extracting some representative features from data, in this experiment, we analyzed several instances of the ϵ parameter from the different leaves of the

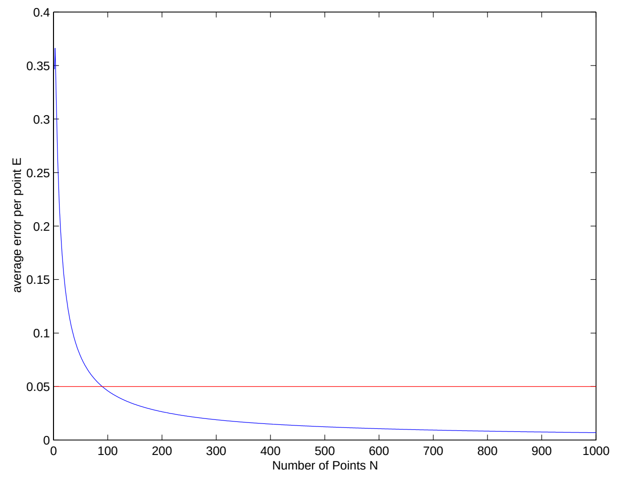


Figure 3.5: For a given fixed error per point $\bar{\mathcal{E}}$ there is a certain number of points N where the stoping condition $\bar{\mathcal{E}} > \frac{5}{2}\frac{\log N}{N}$ is accomplished.

tree for a given scanned piece. When the process arrives to a stop splitting point we collect the ϵ parameter for that specific subset of points. The manner data is scanned and geometrical features such as curvature can be captured though studying the instances of ϵ for different regions. Figures 3.6(f),3.7(f),3.8(f),3.9(f),3.10(f) and 3.11(f) show the corresponding histograms of the ϵ distance measure. In this case, a narrow peaked histogram means a more uniform sampling. While, a broader peak is related to different effects: scanning orientation and curvature complexity. In this case, both effects are entangled, and from these measurements (ϵ-histogram) there is no possible manner of distinguishing each contribution. Nonetheless, we can take some advantages from these features. Actually, these provide a sort signature for each type of piece, yielding a manner for storage and retrieval in archeological databases.

The way data is obtained through scanning determines the process of extracting geometrical features. Not only noise has a certain effect on the final resulting estimates, but also the manner data is sampled. In previous section 3.3, we made use of the ϵ parameter in order to check the breaking curve condition (eq. 3.2.1). This parameter computes comes from a rough estimation of the density of points for a specific considered patch. It helps to check in a chosen direction (principal axis) the boundary condition avoiding extra-checking. For a large number of splits, checking out just in one direction is enough, however, the algorithm can be modified to widen the range of directions for inspection. The experiments we have performed show the application of our algorithm onto a variety of shapes; from nearly flat to highly curved ones. In all of them, checking in one direction yields a sufficiently accurate representation of the breaking curve.

Actually, the higher is ϵ value, the larger is the number of point to be analyzed through equation (3.2.1), the likelihood of selecting a boundary point at that level is lower. Also, a larger density implies a narrower band for inspection, which means that the likelihood of finding a boundary point is higher.

From both sampling and noise-model points of view, the conclusion is the same, the more points for a fixed volume of space, more accurate are the definition of curvature and breaking curve.

3.4.2 Computational Cost

This divide-and-conquer technique in multidimensional space decomposes a geometric problem on N points into two (or a, many) problems on $N/2$ (or N/b) points. The analysis of the algorithm's computational cost is carried out through the use of the *Master Theorem* [9], which is quite helpful for solving recurrences.

Master Theorem 3.4.1 *Let a and b be constants with $a >= 1$ and $b > 1$. Let $f(n)$ be a function and let the time-cost function $T(n)$ be defined for integers by the recurrence:*

$$T(n) = aT(n/b) + f(n)$$

where n/b is rounded (either up or down). Then $T(n)$ can be bounded asymptotically as follows:

1. *If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.*
2. *If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.*
3. *If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ (regularity condition) for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Omega(f(n))$.*

The asymptotic notation for O, Θ and Ω correspond to the following definitions:

Definition 3.4.1 *$f(n)$ is $O(g(n))$ if $\exists$ constants $c, n_0 > 0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$.*

Definition 3.4.2 *$f(n)$ is $\Omega(g(n))$ if $\exists$ constants $c, n_0 > 0$ such that $f(n) \geq cg(n)$ for all $n \geq n_0$.*

Definition 3.4.3 *$f(n)$ is $\Theta(g(n))$ if $\exists$ constants $c_1, c_2, n_0 > 0$ such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$.*

The case we are dealing with corresponds to $a = b$. Data is divided in approximately an equal number of points in each subset for each level. In the algorithm we present, the time-cost when dividing a group into a subgroups is lineal $f(n) = O(n)$. As explained in the pseudo-code in table 3.1, the non-recursive cost consist of the time consumed in computing the sample mean (3N sums) and the covariance matrix (sum of 3N+9N elements). Projecting onto the principal directions is lineal again

(3N) and splitting data into positive and negative projections is lineal (N) as well. Thus, $f(n)$ is $O(n)$. Moreover, using the *Master Theorem* with these parameters, the complexity of our algorithm is $\Theta(n \log n)$, with a data set of n points.

In order to study this behavior, we have performed a experiment consisting of computing the time-consumed as a function of the number of points n in each distribution (figures from 3.6(e) to 3.11(e)). This was performed, by randomly sampling data at different percentages: 10%, 20%, ..., 100% of N. For each selected amount of data, we built 100 random samples from the original distribution. We define, here, random sampling as taking a certain number of points from an original collection of N points. For each sample, we computed the elapsed time when computing the boundary points. For each selected percentage, we computed the sample mean and the standard deviation of the time-cost. In this experiment, we forced the tree to explore a maximum number of 13 levels, since the aim was isolating the contribution of the number of points from the contribution of shape complexity. The recursion was performed by splitting into 2 subsets. Given that at each leaf we can obtain at most 2 extreme points, there are 16384 points candidate to boundary points. The reason of selecting this specific number of levels is that the distribution with less data has 11802 points and the distribution with a maximum number of points has 26436. We chose an intermediate number from the possible powers of 2 in order to compare computational cost attempting to avoid the contribution of shape. Analyzing figures 3.6(e) to 3.11(e), we can see that figures 3.7,3.8,3.9 and 3.10 correspond to distributions with a number of points bigger that 2×2^{13} showing approximately $2sec$ of computational cost. On the other hand, figure 3.6 and figure 3.11 have less than 2×2^{13} point, and thus, the final time-cost is lesser at the 100% sampling.

Even though we fixed the number of recursion levels with the aim of avoiding computational cost due to shape, there is still a small contribution which explains the standard deviation at each sampling percentage.

3.5 Summary and Conclusions

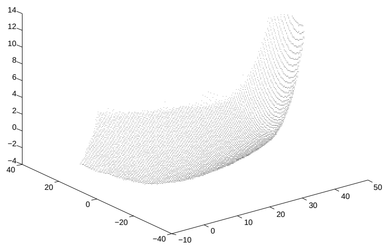
In this chapter, we have presented a technique that computes the breaking curves of $2D$ open surfaces from unorganized sets of $3D$ points. The algorithm approximates a surface through planar patches. For each planar patch the boundary conditions are checked out in order to obtain the breaking curve points. We consider three points of view in order to analyze the contributions of the chapter.

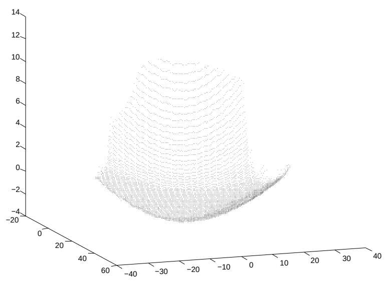
Model selection The contribution of the chapter is to provide the reconstruction of an open surface in terms of taking into account the trade off between error in the estimates and model complexity. In this particular framework, the distinction is performed between noise and geometrical features such as curvature. Moreover, the manner the tree structure is generated permits an incremental model selection at each level. This fact generates the algorithm to be optimum in terms of a Bayesian approach, where the plausibility of different alternative models is evaluated in terms of complexity and degrees of freedom. The introduction of the lower bounds approximation in section 3.2.4 makes feasible taking advantage of computations already performed in order to automatically identify the difference between noise and high curvature regions.

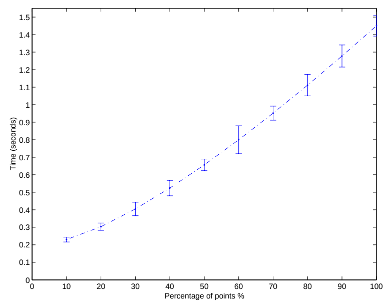
Discrete sampling effects The idea of granularity is based on the fact that data is obtained from a discrete sampling process. The introduction of a parameter related to the density of data has been motivated in order to make the algorithm simple, fast and under low computational cost. Nonetheless, the technique we present allows the introduction of techniques for inspecting boundary points. From experiments, we obtained that a few number of boundary points from each subregion yields to a large number of points when putting together all the estimates. Another contribution presented in the experiments is that the computation of the granularity leads to a manner of extracting signatures for posterior storage and retrieval in archeological databases.

Number of points The experiments have shown an analysis related to the contribution of the number of points N representing a sherd. The analysis has three approaches. One corresponding to the noise/curvature entanglement. A second study related to the discrete distribution effects. From both studies, the conclusion is coherent with a scale-space point of view: the more points representing a surface for a fixed volume of space, the more accurate are the definitions of curvature and breaking curve. The third part is the computational cost analysis, where it is shown the behavior of the algorithm in terms of spent time as a function of the number of points N. Here, the recursive nature of the algorithm yields a simpler way of computing breaking curves $\mathcal{O}(N \log N)$. Local approaches need for neighboring-based computations, i.e., organized points which makes the computational cost $\mathcal{O}(N^2)$.

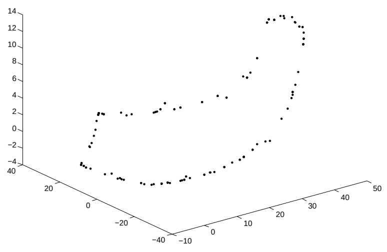
We like to point out that defining a specific standard manner of scanning pieces would provide a suitable framework for classifying pieces through the ϵ-histograms, since the scanning effects would be notably reduced. Defining these specific requirements is a matter of future work.

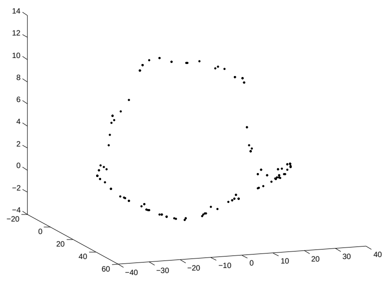

(a)

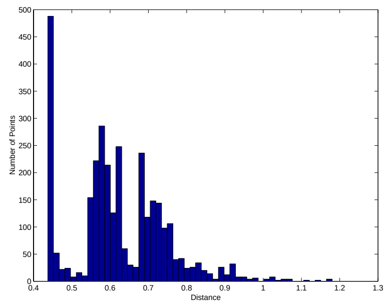

(b)



(e)


(c)


(d)



(f)

Figure 3.6: Two points of view of a range open surface in (a) and (b)where the number of points is 11802. Breaking curve points detected in (c) and (d) corresponding views. Computational cost analysis in (e). ϵ-Histogram in (f).

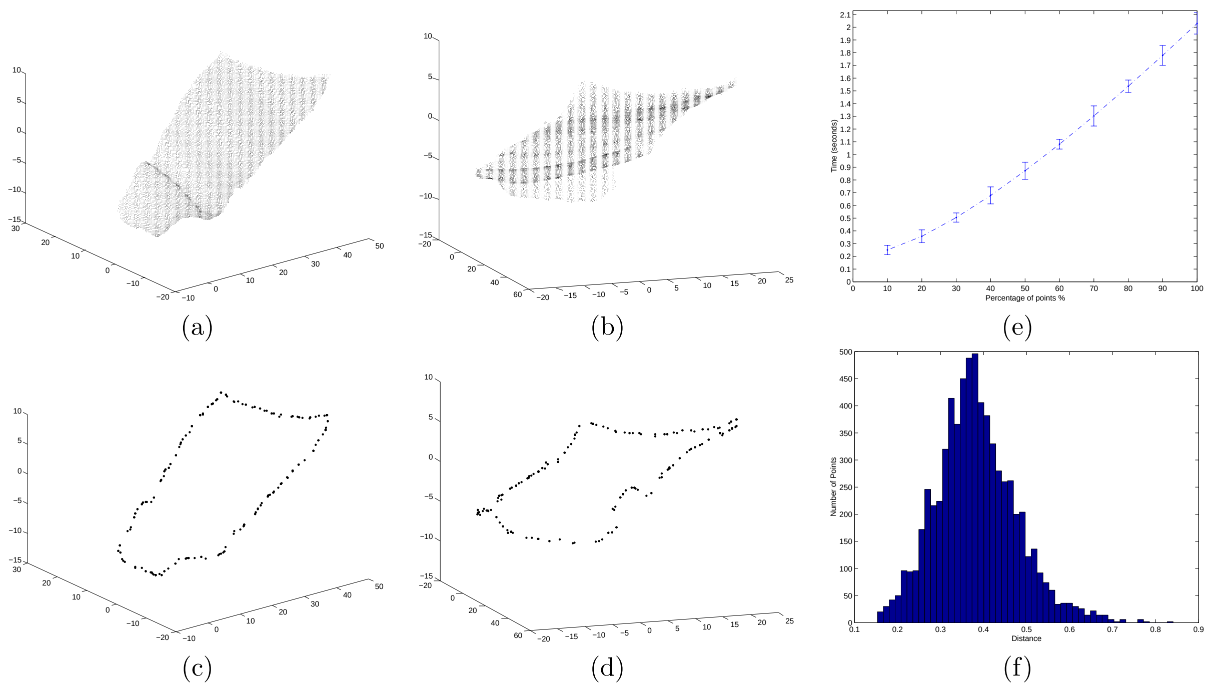
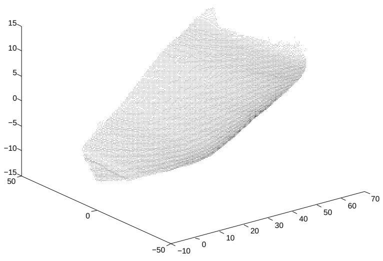
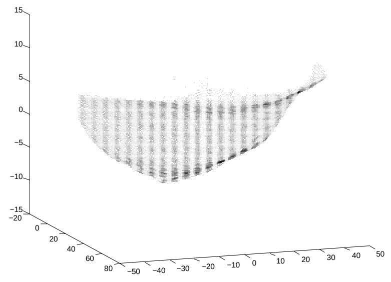


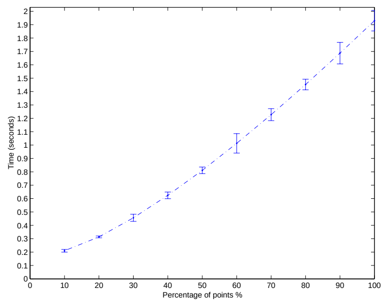
Figure 3.7: Two points of view of a range open surface in (a) and (b)where the number of points is 18471. Breaking curve points detected in (c) and (d) corresponding views. Computational cost analysis in (e). ϵ-Histogram in (f).



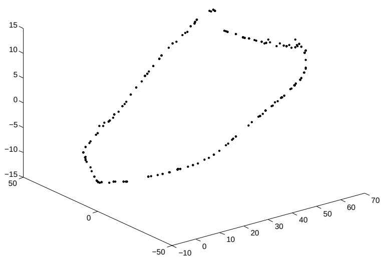
(a)



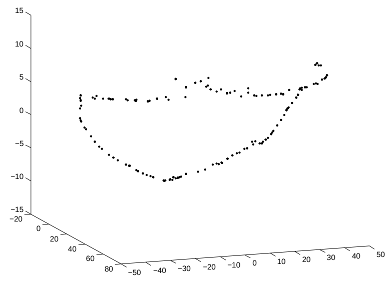
(b)



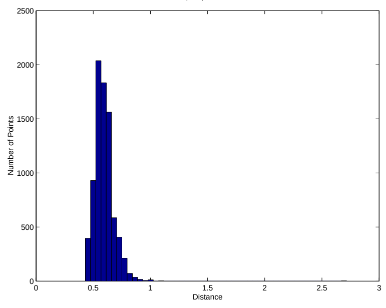
(e)



(c)



(d)



(f)

Figure 3.8: Two points of view of a range open surface in (a) and (b)where the number of points is 26004. Breaking curve points detected in (c) and (d) corresponding views. Computational cost analysis in (e). ϵ-Histogram in (f).