Parametric Curve Reconstruction from Point Clouds using Minimization Techniques

Oscar E. Ruiz[1], C.Cortés[1], M. Aristizábal[1], Diego A. Acosta[2], Carlos A. Vanegas[3]

[1]*Laboratorio de CAD/CAM/CAE, Universidad EAFIT, Carrera 49 No 7 Sur - 50, Medellín, Colombia*

[2] *Grupo de Diseo de Productos y Procesos - DPP, Universidad EAFIT, Carrera 49 No 7 Sur - 50, Medellín, Colombia*

[3]*City & Regional Planning,University of California, Berkeley, 228 Wurster Hall #1850, Berkeley, CA 94720-1850, USA*

{*oruiz,ccortes2,maristi7,dacostam*}*@eafit.edu.co, cvanegas@gmail.com*

Keywords: parametric curve reconstruction, noisy point cloud, principal component analysis, minimization

Abstract: Smooth (C^1-, C^2-,...) curve reconstruction from noisy point samples is central to reverse engineering, medical imaging, etc. Unresolved issues in this problem are (1) high computational expenses, (2) presence of artifacts and outlier curls, (3) erratic behavior at self-intersections and sharp corners. Some of these issues are related to non-Nyquist (i.e. sparse) samples. Our work reconstructs curves by minimizing the accumulative distance curve cs. point sample. We address the open issues above by using (a) Principal Component Analysis (PCA) pre-processing to obtain a topologically correct approximation of the sampled curve. (b) Numerical, instead of algebraic, calculation of roots in point-to-curve distances. (c) Penalties for curve excursions by using point cloud to - curve and curve to point cloud. (d) Objective functions which are economic to minimize. The implemented algorithms successfully deal with self - intersecting and / or non-Nyquist samples. Ongoing research includes self-tuning of the algorithms and decimation of the point cloud and the control polygon.

Glossary

PL	:	Piecewise Linear.
PCA	:	Principal Component Analysis
C_0	:	Unknown planar curve to be fit.
S	:	$\{p_1, p_2, ..., p_n\}$ noisy point sample of C_0.
$C(u)$	:	Parametric planar curve approaching C_0.
S_c	:	$\{c_1, c_2, ..., c_w\}$ PL disjoint curves approaching local portions of C_0.
L	:	PL curve, which integrates S_c.
P	:	$[q_1, q_2, ..., q_m]$ Control polygon of $C(u)$.
μ	:	Stochastic component of S.
ε_k	:	Error of minimization process at iteration k.
$B(p,r)$	:	Disk of radius r centered at p.
$\Lambda(\lambda)$	:	$p + \lambda.\hat{v}$ Parametric line starting at point p, with direction $\hat{v}$ and parameter λ.
$pca(S_E)$	:	PCA of point set S_E. Returns p, $\hat{v}$ and correlation coefficient ρ.
r_i	:	Residual associated to point $p_i \in S$
C_i	:	Point on $C(u)$ closest to cloud point p_i.
$d(p,S)$	:	distance from point p to the point set S.
A_j	:	set of points closest to p_j.
Simple Curve	:	Curve without self-intersections.

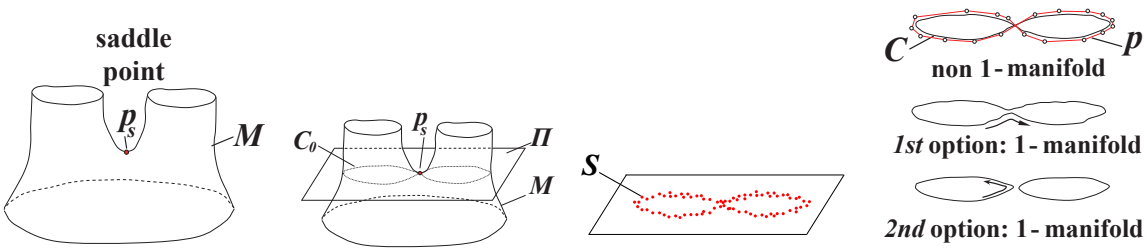
1 Introduction

A considerable number of applications in CAD, CAM, Medical Imaging and Geographic Information Systems, deal with the sectioning of a non-self intersecting shell (a 2-manifold) in $\mathbb{R}^3$ with cutting planes. In Computer Axial Tomography (CAT), Coordinate Measurement Machine (CMM) samples and Magnetic Resonance Imaging (MRI), planar curves must be recovered from noisy unordered point samples.

Consider a planar curve C_0 sampled with a point set S, which includes a stochastic noise with mean μ. This article presents implemented algorithms to find a *b-spline* curve C, which is a single connected parametric approximation for C_0. The curve C is determined by its control polygon P, which we choose to minimize $f = \sum_{i=1}^{n} r_i^2$. r_i is obtained using the distances between the cloud points $p_i \in S$ and the curve C.

1.1 Self-intersecting Curves

A typical scenario for self - intersecting curve reconstruction appears when a shell M (Fig. 1(a)), is cross sectioned by planes. When a plane Π cuts the shell



(a) Saddle Point in 2-manifold M. (b) Cross Cut at Saddle Point. (c) Self-intersecting Point Sample. (d) Self-intersecting C and 1-manifold Derivations.

Figure 1: Rationale for non-manifold curve reconstruction.

M the result is a set of (open or closed) planar curves. In general, those curves do not self - intersect. However (Fig. 1(b)), if a horizontal plane Π passing very near or at the saddle point p_s cuts M, (nearly) self - intersecting curves C_0 are produced. If the curves are point-sampled, the sampling noise blurs their exact geometry and topology. In such conditions it is irrelevant whether or not the plane Π exactly contains the saddle point p_s: the point sample indicates a self - intersecting curve (Fig. 1(c)).

Because of this reason, the recovery of self - intersecting curves is relevant. The algorithms implemented in this article are able to recover a self - intersecting curve C -via its control polygon P- that resembles C_0 (Fig 1(d), upper). The mutation of C into 1-manifold curve or curves is simple, as shown in the lower curves in Fig 1(d).

1.2 Non-Nyquist Samples

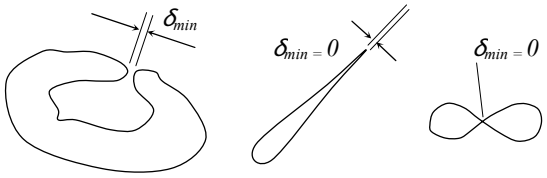


Figure 2: Examples of Nyquist / Non-Nyquist Shapes.

To recover a curve from point samples it is essential that the sample be a Nyquist one. This means, sufficiently dense fo the curve at hand ((Nyquist, 1928), (Shannon, 1949)). In Fig. 2-left, it is possible to sample the curve to correctly recover it, since δ_{min} is different from zero. In Figs. 2-center and 2-right , however, it is impossible to sample the curve tightly enough to be able to recover it. No samlple is Nyquist-compliant for the curves at center and right. Fig. 2-center shows a non-Nyquist curve. Fig. 2-right a non-manifold (therefore non-Nyquist) curve.

2 Literature Review

The problem of fitting a parametric curve $C(u)$ to a point cloud S has been addressed by many authors in recent decades. However, as seen in this section, there are many open issues in the solutions for such a problem.

2.1 Objective function

Eq.1 is the general representation of the objective function in curve fitting problems. Reference (Flöry and Hofer, 2010) employs first order residuals ($w = 1$) while references (Wang et al., 2006; Liu et al., 2005; Gálvez et al., 2007; Liu and Wang, 2008) use second order residuals ($w = 2$).

$$f = \sum_{i=1}^{n} r_i^w \tag{1}$$

Some references ((Wang et al., 2006; Liu et al., 2005; Flöry and Hofer, 2008; Flöry and Hofer, 2010; Flöry, 2009)) add a smoothing term f_c to the objective function in order to adjust the roughness of the curve:

$$f = \sum_{i=1}^{n} r_i^w + \lambda f_c. \tag{2}$$

The term f_c contains information on the curve's first and/or second derivatives and λ determines its influence, penalizing large curvatures. Note that (1) f_c prevents from reconstructing curves with sharp corners and (2) it is necessary to find the proper value for λ for each case of study. If the data to recover presents smooth and sharp-cornered neighborhoods at the same time, the strategy using λf_c alone will not be able to recover the proper curve.

Some authors have explored constrained approaches. Reference (Flöry, 2009) presents constrained curve and surface fitting to a set of noisy points in the presence of regions that the curve or surface must avoid. Reference (Flöry and Hofer, 2008) considers the problem of curves that must lie on a

2-manifold (surface) with forbidden regions. These procedures are implemented using a constrained non-linear optimization strategy.

2.2 Distance measurement

Eq.3 corresponds to the calculation of the distance d_i, which is usually used in the objective function in Eq. 1 as the residual r_i. In curve fitting algorithms norm b is usually chosen to be $b = 2$ (i.e., Euclidean distance) as in (Wang et al., 2006; Liu et al., 2005).

$$d_i = \min_{C(u)\in C} \|C(u) - S_i\|_b \tag{3}$$

However, the exact calculation of d_i is expensive, since it is obtained by a minimization procedure at each fitting iteration. The procedure consists of finding the parameter u_i which associates a point on the curve $C(u_i)$ with the i-th cloud point p_i such that d_i is a minimum. Namely,

$$\|C(u_i) - p_i\|_b = \min_{C(u)\in C} \|C(u) - p_i\|_b \tag{4}$$

The minimum distance is obtained by performing an orthogonal projection of the point p_i to the curve C, which occurs when $C'(u_i) \cdot d_i = 0$ in Eq. 5.

$$g(u) = \left| C'(u) \cdot (C(u) - p_i) \right| \tag{5}$$

To sort out this problem, one could solve for u in $g(u) = 0$ using Newton's Method ((Piegl and Tiller, 1997; Liu et al., 2005)), or minimize $g(u)$ using numeric methods ((Wang et al., 2006; Liu and Wang, 2008; Flöry and Hofer, 2010; Saux and Daniel, 2003)) or using genetic algorithms ((Gálvez et al., 2007)).

The mentioned approaches have drawbacks inherent to numerical methods, such as the need of a good initial guess, poor convergence and stagnation at local minima. These may lead to poor approximations of the distance d_i yielding unsatisfactory results of the fitting procedure.

Different methodologies to measure the point-to-curve distance have been proposed: (i) Point distance, which preserves the Euclidean distance between the cloud point and the paired point of the curve ((Wang et al., 2006; Flöry and Hofer, 2008)), (ii) Tangent distance, which only preserves the distance between the cloud point and the tangent line projected at the paired point (Blake and Isard, 1998), and (iii) Squared distance, which is a curvature-based quadratic approximation of d_i^2 ((Wang et al., 2006)). Reference (Liu and Wang, 2008) presents a comparison of these methodologies.

It should be noticed that using only point-to-curve distances to calculate f allows the formation of spurious curls and outliers (Fig. 8). Because of this reason, we include also curve-to-point distances in f. This double estimation allows us to avoid curls and outliers.

In the literature reviewed, we have not found the usage of a PL approximation of a curve C to estimate the distance between a point and C. Although this is an intuitively simple procedure, it seems that it has been turned down in favor of more sophisticated methods. In this article, however, we discuss and implement its usage.

2.3 Initial guess

In the present article the term *initial guess* refers to a PL or smooth curve which approximates (possibly with errors) the global point cloud. Numerical optimization strategies require an initial guess to set the decision variables. This set of values are chosen to produce a reasonably good value of the function to optimize, to then start the iterations. A good initial initial guess has the capacity to offset the difficulties posed by non-convex functions. If the initial guess falls in a neighborhood in which the function is locally convex and a solution exists, the minimization algorithms will be able to reach such solution, regardless of non-convexities of the function elsewhere.

References (Lee, 2000; Cheng et al., 2005; Furferi et al., 2011) provide methods to find a proper initial guess of C when reconstructing simple curves from noisy point clouds. However, in domain of self - intersecting curve fitting to noisy point clouds there are very few references that address this issue.

References (Flöry and Hofer, 2010; Wang et al., 2006) start with a user - defined initial guess of the curve sought. On the other hand, reference (Wang et al., 2006) proposes to compute the quadtree partition of the point cloud and then extracts a sequence of points which approximates the target shape. These points are then used as initial control points of the fitting curve. However, the autors do not report the implementation of such method.

Ref. (Zhao et al., 2011) mentions the usage of a 2D grid of uniform cells (i.e., not a quadtree), on which the cloud points fall. The curve, therefore, must lie on the set of cells which receive sampled points. The centers of these cells are calculated and interpolated to obtain the initial curve. However, the strategy to *order* these center points is not presented. It must be noted that the fitting of a PL or smooth curve is precisely the problem of introducing a *total order* among points which are representative of local neighborhoods (either quadtree-based or cell-based) of the point sample. The *total order* issue is particularly pressing for self-intersecting and non-Nyquist

curves.

2.4 Complexity Analysis

Liu et al. in (Liu and Wang, 2008) argue that the computation of the Hessian using direct second order derivatives of the parametric curve is a very expensive operation. Because of this, they classify different approaches to calculate the point-to-curve distance, based on the usage or avoidance of the second derivatives of C (see section 2.2). This reference does not calculate the complexity of the different approaches used. The other literature reviewed ((Park and Lee, 2007), (Flöry, 2009), (Flöry and Hofer, 2010), (Song et al., 2009), (Liu et al., 2005), (Wang et al., 2006)) reports the execution times but does not conduct a formal complexity discussion.

It is the case that pre-processings (e.g. initial guess findings) are usually assessed in terms of $O(n)$ while minimizations are assessed in terms of ε_k (admissible convergence error). In order to correctly estimate the computational expenses for optimized curve fitting to point clouds, we require the description of the computational expenses for: (i) pre-processing (initial guess generation) and (ii) optimization, on the same grounds (either convergence speed of ε_k or complexity $O(n)$). Such an integration is not a trivial task.

In this article we present an algorithm which conducts pre-processing (i.e., initial guess for P) and the optimization of P. The expensive pre-processing is fully justified by the topological and geometrical correctness of the result, and the increased speed of the optimization process.

2.5 Conclusions Literature Review and Contribution of this Article

According to the taxonomy conducted in this literature review, there are several issues that remain open in optimized curve fitting to point clouds. These aspects include: (a) Finding of topologically correct initial guesses to offset the fact that possibly f and / or the minimization region are non - convex. (b) Usage of a function f which efficiently fits the point set, allowing for self - intersections and non-Nyquist samples. (c) Decimation of the number of control points for C given that m control points imply a minimization space of dimension $\mathbb{R}^{2m}$. (d) Formal assessment of the computational time and space required for the solution (e.g. $O(n)$ analysis).

In response to the previous considerations, this article reports the implementation of: (i) An initial PCA-based initial guess for P and therefore C, which is topologically faithful to the point set, being able to follow self - intersections and non-Nyquist samples. (ii) Decimation of excessive control points by the implementation of a curvature-based re-sampling, to reduce the dimension of the search space. (iii) A discrete calculation of the distance point-curve by using a re-sampling of the curve. (iv) A double penalization included in the objective function, based on the distances cloud-point-to-curve and curve-to-cloud-point, therefore avoiding the existence of spurious curls and outliers in C.

It must be pointed out that, although a significant amount of work is required in the study of computational complexity, we do not intend to make a contribution in such an aspect.

3 Methodology

The methodology used comprises the following stages: (1) data pre-processing and (2) optimization procedure. Stage (1) leads to a PL approximation of the data to obtain a topologically correct initial guess of the b-spline control polygon. Stage (2) implements a Gauss-Newton optimization algorithm to minimize the distance between the curve and the points S using a penalized objective function. Fig. 3 summarizes the procedure.

3.1 Data Preprocessing

Given the S point set, an *initial guess* L for P (and hence for C) is required, which is a rough PL approximation of the curve C sought. This initial guess is fundamental for the success of the optimization algorithm that seeks C. L has correct topology (produces the same number of self-intersections as in C_0), although it might not precisely follow C_0. This initial guess is calculated by a PCA pre-processing stage of our algorithm.

Our approach is an extension of the work by (Ruiz et al., 2011). The algorithm in (Ruiz et al., 2011) *estimates* the tangent to C at a particular point p of C by running a PCA (i.e., generalized linear approximation $\Lambda(\lambda) = p_{cg} + \lambda.\hat{v}$ with starting point at p_{cg}, direction $\hat{v}$ and parameter λ) on the sample points of such neighborhood. Those points are the subset of S contained in a small circular ball $B(r,p)$ based on p. The goodness of the PCA can be evaluated using a variation of the linear regression correlation coefficient ρ. In a 'normal' (i.e., 1-manifold conditions) curve neighborhood, the linear approximation is very good, and therefore $\rho \approx 1.0$. In self - intersecting (i.e., non-manifold) or non-Nyquist curve neighborhoods, the linearity falls and $\rho \approx 0$. At such neighborhoods,

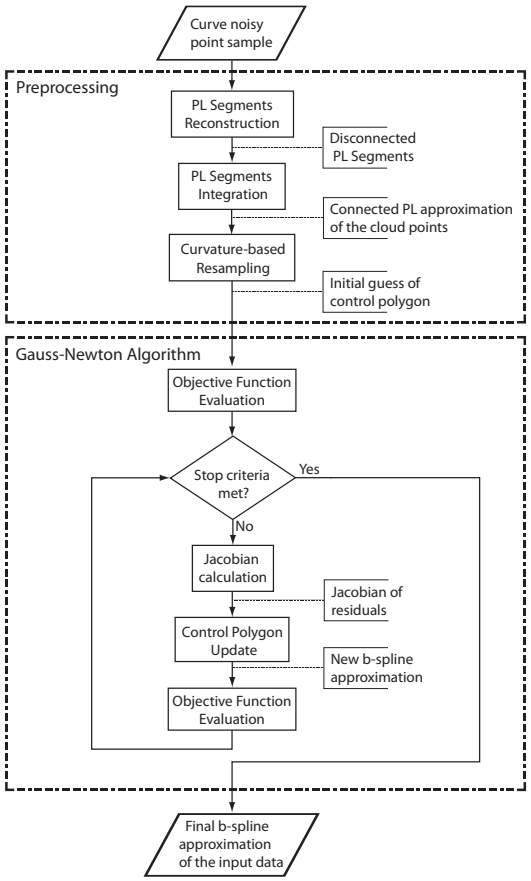
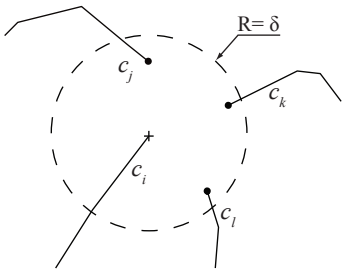


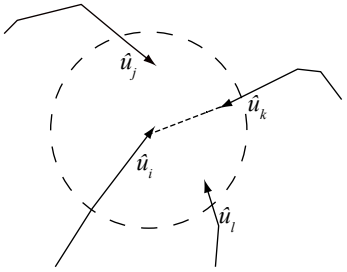
Figure 3: Block diagram of the proposed methodology.

the support region choosing the points to participate in the PCA mutates from circular into an elliptical one. This is a key feature of the algorithm because it permits to deal with self-intersections. The process continues with the adjacent neighborhoods until a dead end is found (i.e., no more points are available in the current direction of search), and another region is explored. The procedure is repeated until S is completely traversed, delivering a set of disconnected PL curves, $S_c = \{c_1, c_2 \ldots, c_h\}$, which approximates S.

Notice that regardless of the initial curve being connected or disconnected, any curve reconstruction algorithm must be prepared to integrate different fragments which are intermediate results of the reconstruction. In the present article, we improve (Ruiz et al., 2011) by adding a novel integration strategy for the c_is to obtain a single connected approximation of C_0, defined as L. Let c_i, c_j, c_k, c_l be PL curve fragments to be merged, which happen to have their endpoints close to each other (see Fig. 4(a)). In this case, the distance criterium is obviously insufficient to decide which curve fragments should join. Therefore, an additional criterium is used, namely the similarity of tangent vectors at the curve endpoints. Based on it, it is concluded that the pairs to join are (c_i, c_k) and (c_j, c_l) for the example shown in Fig. 4(b). In the general case, however, the c_i curve fragments integrate unambiguously. This procedure is a heuristic one and therefore it has no theoretical guarantee for correction.



(a) Geometrically close endpoints.



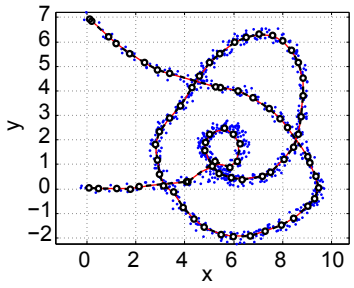
(b) Similar end-tangents.

Figure 4: Criteria for integration of curve fragments c_i

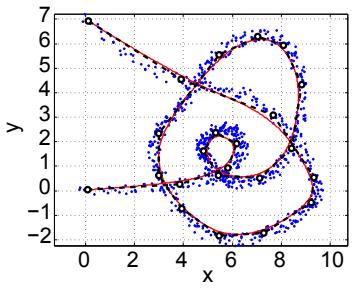
The PL curve, L, integrating the c_is is topologically equivalent to C_0, meaning that L has self intersections if C_0 does. However, L is not a good approximation of C_0 in the geometrical sense (i.e., it does not lie in the 'center' of the point set S). Therefore, L by itself does not solve the problem.

The initial guess of the control polygon P for C is obtained by re-sampling L, since one wants to use the bare minimum necessary control points that retain the topology of C. The re-sample of L is controlled by the curvature: low curvature (large curvature radius) regions can be represented by fewer control points, and vice versa. Given a point p_i of L, the local radius of curvature, R_{p_i} may be approximated by the radius of the circumference defined by three adjacent point samples, p_{i-1}, p_i, p_{i+1}. The curvature, K_{p_i} is given by $K_{p_i} = \frac{1}{R_{p_i}}$. The larger the K_{p_i}, the tighter of the re-sample of L. The initial and final points of L are always included in the resulting re-sampling. This curvature-based re-sampling strategy strongly contributes to lower the computation time in the subsequent optimized fitting since less variables need to be estimated (see Fig. 5).

We name as P the re-sampled version of L. Also



(a) Initial control polygon using L.



(b) Re-sampled version of L.

Figure 5: Curvature-based control point decimation to estimate the initial control polygon.

to keep notation simple, we use P to note the different instances of itself (P_1, P_2, .., P_k, ...), formed as its vertices are relocated during the optimization iterations k. Notice also that P is the control polygon for curve C.

3.2 Optimization Problem

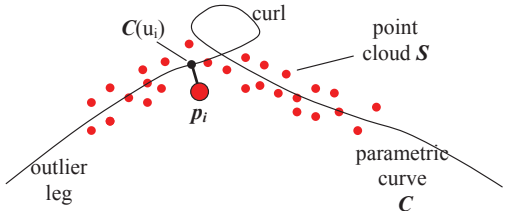
The control polygon P resulting from the data preprocessing in section 3.1 determines the topology of C. The geometrical quality of P is, as expected, poor. This means, the parametric curve controlled by polygon P is not a principal curve for the point set S (i.e., does not cross the 'center' of S). Therefore, the control points of P must now be used as decision variables to minimize the summation of the squared distances from the cloud points (i.e., points in S) to the candidate curve $C(u)$. A Gauss-Newton algorithm is used to minimize the f function which expresses the distances between the point cloud S and its approximating curve C.

The minimization problem is stated as follows:
GIVEN: A noisy point sample $S = \{p_1, p_2, ..., p_n\}$ of a planar parametric smooth (possibly self-intersecting and with non-nyquist neighborhoods) curve C_0.
GOAL: To determine the control polygon P that produces a b-spline curve C, minimizing

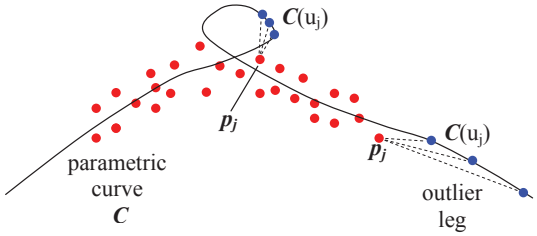
$$f = \sum_{i=1}^{n} r_i^2 \tag{6}$$

■

where r_i is the residual or distance between cloud point p_i and the curve C. Informally, r_i depends on the distances from the cloud points in S to the curve C and on the distances from the curve C to the point clouds in C. As discussed later, these distances differ. Depending on the definition of r_i, two different strategies to minimize f arise, as discussed next.



(a) Distances Cloud Point to Curve.



(b) Distances Curve to Cloud Point.

Figure 6: Distances Cloud Points to/from Curve.

3.2.1 Strategy 1. Distance from Cloud Point to Curve

We define the *residuals* as

$$r_i = ||p_i - C(u_i)|| \tag{7}$$

where u_i is the parameter in the domain of C which defines the point $C(u_i)$ closest to p_i. The term r_i represents the distance measured from each cloud point to the curve C (see Fig. 6(a)). This calculation of the distance between a point and an algebraic curve is a very expensive proposition because it implies the calculation of common roots of a polynomial ideal (see (Ruiz and Ferreira, 1996), (Kapur and Lakshman, 1992)). Notice that the vector $p_i - C(u_i)$ is normal to the curve C at the point $C(u_i)$.

To address this problem, we approximate $C(u)$ in PL manner and calculate r_i simply by an iterative process. We sample the domain for $C(u)$, ($[0,1]$) getting $U = [0, \Delta_u, 2\Delta_u, ..., 1.0]$ and approximate the current C curve with the poly-line $[C(0), C(\Delta_u), C(2\Delta_u), ..., C(1.0)]$. Calculating an approximation of $C(u_i)$ for a given p_i simply entails to traverse $[C(0), C(\Delta_u), C(2\Delta_u), ..., C(1.0)]$ to find

the $C(N\Delta_u)$ closest to p_i. This is an $O(n)$ process (n=number of points), which is sufficiently accurate and avoids the high computational expenses and numeric precision required to solve an algebraic system of equations ($O(e^{e^n})$, n=number of polynomials, (Kapur and Lakshman, 1992)). Also, we avoid dealing with the problem of finding the cloud point-to-curve distance as a new minimization problem, which is the most adopted method in literature.

Fig. 6(a) displays the distance from a particular (emphasized) cloud point p_i to its closest point $C(u_i)$ on the current curve C. Such a distance has influence in f as per Equation 6. Notice, however, that p_i and $C(u_i)$ (and hence f) do not change if large legs and curls appear in the synthesized C. Therefore, considering only the distance *from cloud points to the curve* in Equation 6 permits the incorrect formation of outlier legs and curls. The following section corrects such a shortcoming.

3.2.2 Strategy 2. Inclusion of Distance from Curve to Cloud Point

This section discusses how to include in the r_i residuals the distances from *the curve points* C_i *to the cloud points* p_i (see Fig. 6(b)) to penalize in f the growth of spurious outlier legs and / or curls just described. If one can make spurious legs and curls to inflate the objective function f, the minimization of f avoids them. For any point $p \in \mathbb{R}^n$, the distance of this point to S is a well defined mathematical function: $d(p,S) = \min_{p_j \in S}(||p - p_j||)$. For the current discussion the points p are of the type $C(u_i)$ (i.e., they are points of curve C). The u_i parameters to use are the sequence $U = [0, \Delta_u, 2\Delta_u, ..., 1.0]$, already mentioned.

Notice that $d(p,S) = ||p_j - p||$ for some cloud point $p_j \in S$. Let us define the point set A_j (on the curve C) as:

$$A_j = \{C(u) \mid u \in U \ \wedge \ d(C(u),S) = ||p_j - C(u)||\} \quad (8)$$

The set A_j contains those points in the sequence $[C(0), C(\Delta_u), C(2\Delta_u), ..., C(1.0)]$ that are closer to the point $p_j \in S$ than to any other point of S. We note with Z_j the cardinality of A_j. Observe that some Z_j might be zero, since p_j could be far away from be curve C and no point on the curve would have p_j as its closest in S. The set of all A_js could also be understood as a partition of the curve C.

With the previous discussion, a new definition of the residuals r_i, to be used in Eq. 6, is possible:

$$r_i = ||p_i - C(u_i)|| + (\frac{1}{Z_i}) \sum_{C_\omega \in A_i} ||C_\omega - p_i|| \quad (9)$$

The $||p_i - C(u_i)||$ in Eq. 9 considers the distance from cloud points in S to the curve C. The term $(\frac{1}{Z_i}) \sum_{C_\omega \in A_i} ||C_\omega - p_i||$ expresses distances from the curve C to the cloud points in S. This term penalizes the length of the curve, by increasing f.

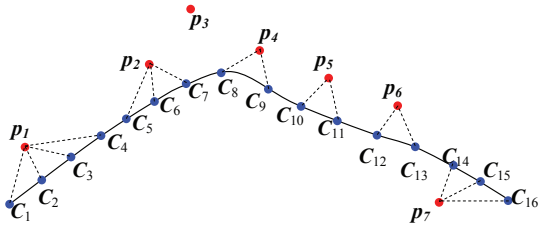


Figure 7: Clusters of Distances form Curve to Cloud Points.

Fig. 7 presents a simplified materialization of the situation. The previous discussion applied to Fig. 7 implies the calculations shown in Table 1. Observe that $C_i = C(u_i)$, the point on C closest to p_i is not the exact one but the approximation mentioned in previous paragraphs (using a tight PL approximation of C).

3.2.3 Jacobian in the Gauss-Newton Method

The Gauss Newton method uses the Hessian approximation $H \approx J * J^T$, which works well in the cases in which the function or region Ω are not convex. Because in our case f is not convex, we choose this method for the minimization. The Gauss - Newton method presents better convergence with small residuals r_i (Chong and Żak, 2008). Because of this reason, we use a good quality initial guess (in this case, a PCA-based one), which indeed produces low values in the residuals.

The Gauss-Newton optimization procedure employs an approximation to the Hessian matrix H. The calculation of H implies several aspects: (i) it is expensive, (ii) it is actually a discrete approximation, (iii) its usage in the search algorithm may mislead it. Because of these reasons, for the present article the Hessian matrix will be approximated as $H \approx \mathbf{J}^T\mathbf{J}$, where $\mathbf{J}$ is the Jacobian of the residuals with respect to the decision variables. This approximation produces a faster convergence of the decision variables to a local minima than using the exact Hessian. The variables to tune f are the x and y coordinates of the control points ($q_i = (x_i, y_i)$) contained in the control polygon $P = [q_1, q_2, ..., q_m]$. The Jacobian of residuals is cal-

p_i	A_i	Z_i	$C(u_i)$	r_i
p_1	$\{C_1, C_2, C_3, C_4\}$	4	C_2	$\|\|p_1 - C_2\|\| + \frac{1}{4}(\|\|p_1 - C_1\|\| + \|\|p_1 - C_2\|\| + \|\|p_1 - C_3\|\| + \|\|p_1 - C_4\|\|)$
p_2	$\{C_5, C_6, C_7\}$	3	C_6	$\|\|p_2 - C_6\|\| + \frac{1}{3}(\|\|p_2 - C_5\|\| + \|\|p_2 - C_6\|\| + \|\|p_2 - C_7\|\|)$
p_3	$\{\}$	0	C_8	$\|\|p_3 - C_8\|\|$
p_4	$\{C_8, C_9\}$	2	C_9	$\|\|p_4 - C_9\|\| + \frac{1}{2}(\|\|p_4 - C_8\|\| + \|\|p_4 - C_9\|\|)$
p_5	$\{C_{10}, C_{11}\}$	2	C_{10}	$\|\|p_5 - C_{10}\|\| + \frac{1}{2}(\|\|p_5 - C_{10}\|\| + \|\|p_5 - C_{11}\|\|)$
p_6	$\{C_{12}, C_{13}\}$	2	C_{12}	$\|\|p_6 - C_{12}\|\| + \frac{1}{2}(\|\|p_6 - C_{12}\|\| + \|\|p_6 - C_{13}\|\|)$
p_7	$\{C_{14}, C_{15}, C_{16}\}$	3	C_{14}	$\|\|p_7 - C_{14}\|\| + \frac{1}{3}(\|\|p_7 - C_{14}\|\| + \|\|p_7 - C_{15}\|\| + \|\|p_7 - C_{16}\|\|)$

Table 1: Calculations using *curve to cloud-point* distances for example in Fig. 7.

culated as follows:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial y_1} & \frac{\partial r_1}{\partial x_2} & \frac{\partial r_1}{\partial y_2} & \dots & \frac{\partial r_1}{\partial x_m} & \frac{\partial r_1}{\partial y_m} \\ \frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial y_1} & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{\partial r_n}{\partial x_1} & \frac{\partial r_n}{\partial y_1} & \frac{\partial r_n}{\partial x_2} & \frac{\partial r_n}{\partial y_2} & \dots & \frac{\partial r_n}{\partial x_m} & \frac{\partial r_n}{\partial y_m} \end{bmatrix} \quad (10)$$

The dimension of the $\mathbf{J}$ matrix is $(n \times 2m)$ where $n =$ number of cloud points and $m =$ number of points in the control polygon. The calculation of each component of the Jacobian is made numerically, using the approximation of the partial derivative $\frac{\partial r_n}{\partial x} = \frac{\Delta r_n}{\Delta x}$. Notice that the variation of all decision variables are the same Δx.

The transformation of the control polygon is made using the Jacobian of the residuals with the expression

$$\mathbf{X}_{k+1} = \mathbf{X}_k - \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\mathbf{r} \quad (11)$$

The dimensions associated with Eq. 11 are: $\mathbf{X}$ $(2m \times 1)$, $\mathbf{r}$ $(n \times 1)$. $\mathbf{X} = [x_1, y_1, x_2, y_2, ..., x_m, y_m]^T$ is just a convenient form for writing P. Notice that $\mathbf{r} = [r_1, r_2, ..., r_n]^T$.

3.2.4 Criteria for Algorithm Termination

The Gauss-Newton procedure will continue finding new control polygons until either of the following conditions is met: (a) the variation of the objective function from iteration k to $k+1$ falls under a threshold ($|f_{k+1} - f_k| \leq f_L$), (b) the values of the decision variable do not significantly change between iteration k and $k+1$ ($|X_k - X_{k+1}| < \delta_{min}$), (c) the iterations executed surpass a limit: ($N_{iter} > N_{max}$).

4 Results and Discussion

4.1 Case Study 1. Simple Curves

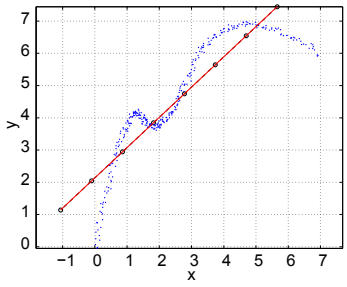
4.1.1 Strategy 1. f Based on Distance Cloud-to-curve

The input data in this case of study corresponds to a simple curve, as per Fig. 8(a). The initial (naive) guess for the control polygon P is a straight segment. It is obtained from an overall linear regression using all cloud points. A sequence of points is sampled on such a straight segment which constitutes the initial control polygon P. The rationale for this initial guess is that the optimization process will, progressively, relocate the control points, until the parametric curve C resulting from P, approximating C_0 is achieved. The mathematical problem solved has the form discussed in section 3.2.1.

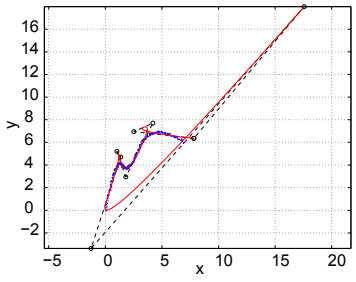
Notice that the case currently discussed uses the simple form of the f function, in which only distances from S to C are considered (excluding distances from C to S), as per Eq. 7. The results, in Fig. 8(b), show that the b-spline curve tends to follow the shape of the cloud points at some regions, but its endpoints are located far away from their correct positions and some curls appear. As discussed before in section 3.2.1 and displayed in Fig. 6(a), such curls and otulier legs appear because the extent of C is not penalized in Eq. 7.

4.1.2 Strategy 2. f Suplemented with Distance Curve-to-cloud

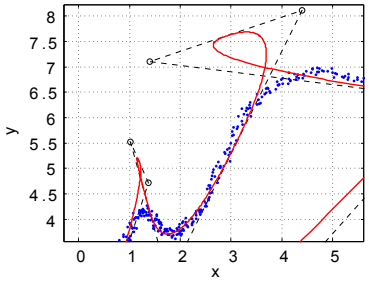
With point set S as per Fig. 9(a), the minimized curve fitting is carried out now by adding curve-to-cloud distances to the f objective function, as specified by Eq. 9. This run uses the naive initial guess for P (i.e., sequence of colinear points). The resulting curve C is correct, suppressing the curls and keeping the curve



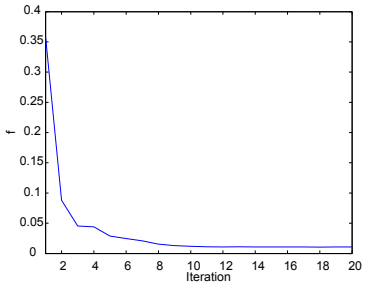
(a) Point Cloud and Naive Initial P guess.



(b) Synthesized C with curls and Outlier Legs.



(c) Close-up at curls of (b).



(d) Objective Function f as function of iteration number.

Figure 8: Case Simple Curve C. Optimization uses naive initial guess and adds curve-to-cloud distance factor.

attached to the ends of the input data. It is observed that very few iterations are needed to yield satisfactory results, showing fast convergence.

4.2 Case Study 2: Self-intersecting curve

The point set for this run series appears in Fig. 10(a). As seen, it comes from a self - intersecting curve C. The goal of this section is to evaluate the effects of a smart initial guess for P, on the curve reconstruction results. From now on, the form of the f function fo minimize is the one in Eq. 9. This means, the distances cloud-to-curve and curve-to-cloud are used.

4.2.1 Naive Initial Guess for P

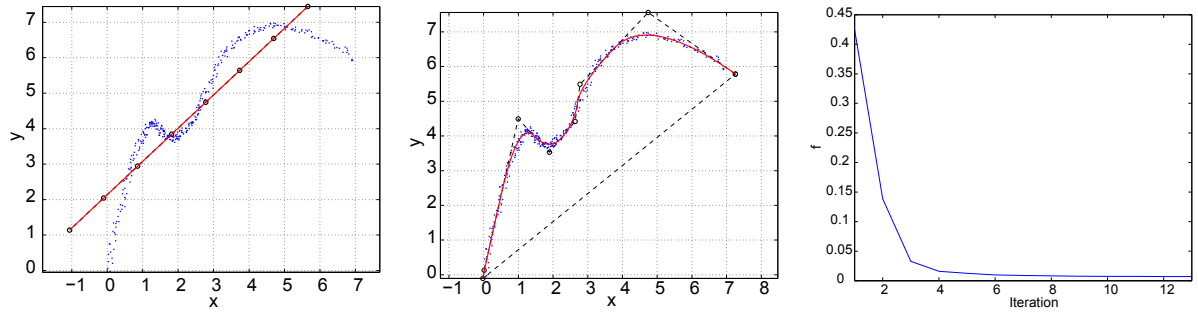
Fig. 10(a) shows the naive initial guess for the control polygon P (i.e., sequence of colinear points). Notice that a sufficient number of points must be sampled on the segment. Too few sampled points obviously prevent following the topological evolutions of the curve. Too many control points considerably degrade the performance of the minimization algorithm, since this number of points equals the dimension of the solution space. Fig. 10(b) shows that with this naive guess, the reconstruction of C_0 dramatically fails, even if a sufficient number of control points are provided on the straight segment of Fig. 10(a). Fig. 10(c) shows the evolution of f along the iterative process.

4.2.2 PCA-based Initial Guess for P

The goal of this run is to test the effect of having a topologically correct initial guess for P, the control polygon of C. Fig. 11(a) shows the point set sampled on a self-intersecting C. This figure also shows an initial guess for the control polygon P found by the PCA-based pre-processing. Observe that this initial P captures the correct *topology*, although not the right *geometry*, of C. Therefore, one requires a minimization stage to relocate the vertices of P. The resulting control polygon P and its corresponding curve C appear in Fig. 11(b). Fig. 11(c) shows the history if f as the iterations take place.

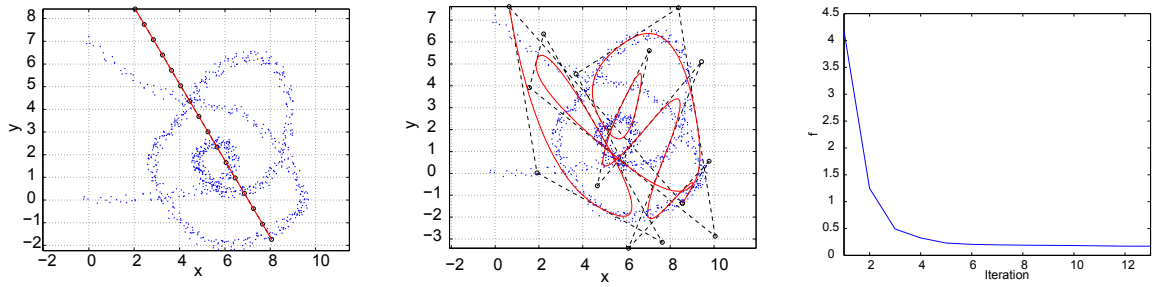
4.3 Case Study 3: Sharp-cornered curve

Fig. 12(a) presents a point sample for a curve that has only C^0 continuity. The Nyquist principle ((Shannon, 1949),(Nyquist, 1928)) applied to geometry sampling requires that at this point the sampling distance raises to infinity and the sampling interval drops to zero, implying that an infinitely tight sample would be required to recover all the geometric information of the original curve C_0. The obvious compromise indicates that, since only a finite sample is available, part of the needle is amputated in the curve-reconstruction.



(a) Point Cloud and Naive Initial P guess. (b) Final control polygon P and its fit curve C. (c) Objective Function f as function of iteration number.

Figure 9: Case Simple Curve C. Optimization uses naive initial guess and adds curve-to-cloud distance factor.

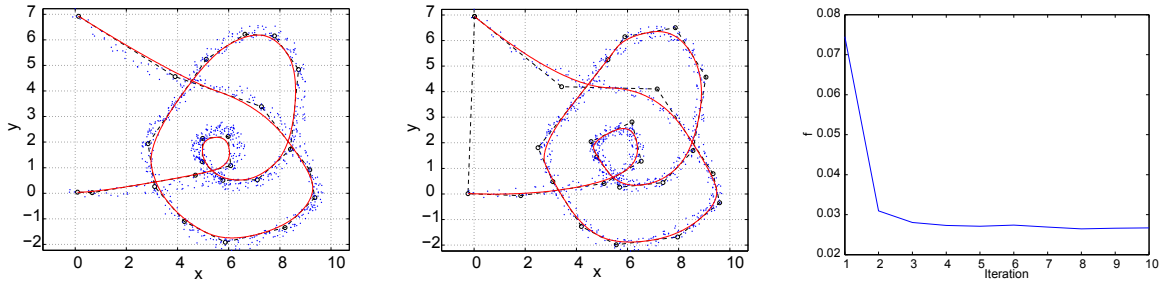


(a) Point Cloud and Naive Initial P guess.

(b) Final control polygon P and its fit curve C.

(c) Objective Function f as function of iteration number.

Figure 10: Open Curve with Self - intersection. C fit without PCA pre-processing.



(a) Point cloud S and PCA-based initial control polygon P

(b) Final control polygon P and its fit curve C.

(c) Objective Function f as function of iteration number.

Figure 11: Open Curve with self-Intersections. C fit using a PCA-based initial guess.

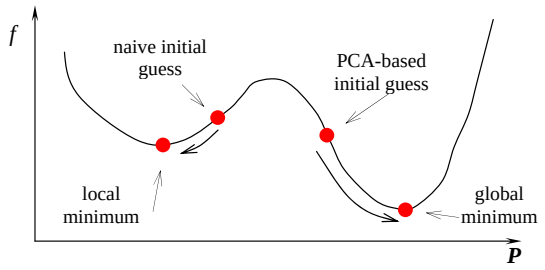


Figure 13: Need of sensible guess with non-convex f.

Unfortunately, even accepting such an amputation, is very likely that in such cases (of non-Nyquist samples), the *topology* of the curve cannot be recovered. The present example represents a very difficult (non-Nyquist) data set.

Fig. 12(a) presents the initial guess for the control polygon P originated using PCA. Fig. 12(b) displays the C the parametric curve finally achieved by the minimization algorithm. Fig. 12(c) shows the history of the minimized f as function of the iteration. As seen, since the application of PCA pre-processing is able to correctly recover the topology of the curve C, it is relatively straightforward for the minimization process to tune up the vertices of P for the correct placement of C. This efficiency is evident from the fast convergence in very early iterations in Fig. 12(c).

4.4 Discussion

4.4.1 Initial Guess and Globality

The Hessian matrix for the problem in Eq. 6 is:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial y_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial y_m} \\ \frac{\partial^2 f}{\partial y_1 \partial x_1} & \frac{\partial^2 f}{\partial y_1^2} & \frac{\partial^2 f}{\partial y_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial y_1 \partial y_m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial y_m \partial x_1} & \frac{\partial^2 f}{\partial y_m \partial y_1} & \frac{\partial^2 f}{\partial y_m \partial x_2} & \cdots & \frac{\partial^2 f}{\partial y_m^2} \end{bmatrix} \quad (12)$$

with the minimum search domain Ω being a subset of $\mathbb{R}^{2m}$. If the eigenvalues of $\mathbf{H}$ are all positive in Ω, one would be in presence of an overall convex function f and the minimization algorithm would rapidly find the solution, guaranteed to be a global minimum. Unfortunately, f as per Eq. 6 is not globally convex. Notice that $\mathbf{H}$ in Eq. 12 is not available and we use $H \approx \mathbf{J}^T\mathbf{J}$ instead. This replacement obviously does not change the non-convexity of f.

Fig. 10 illustrates the effect of not having a globally convex function f. The initial guess for the polygon P is not a sensible one, which means that the initial location in the Ω space is far from the minimum. A poor initial guess for P produces a worng fitting result, as shown in Fig. 10(b). This wrong result explains our need of the PCA-based pre-processing. Fig 10(c) shows that the solution is a *local* minimum. f is indeed minimized in a *wrong* neighborhood, and the solution for P (and C) is equally wrong. Fig. 13 shows that, precisely because the non-convexity of f, an intelligent initial guess for P is required for the minimization algorithm to find a global minimum for f. We claim here that the PCA-based pre-processing improves the chances for an initial guess sufficiently close to the global minimim and therefore partially offsets the inherent difficulty represented by the non-convexity of f.

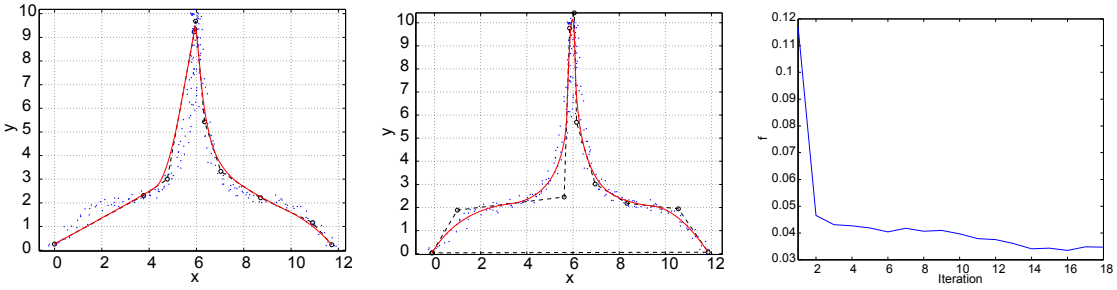
4.4.2 Distance Curve to Point Cloud

As mentioned, the minimization using Eq. 7 permits the formation of curls and leg outliers as per Fig. 8(b). This calculation of f includes only the distance from the cloud points p_i to the curve C. To prevent these spurious formations we include in the f function the distance from the curve C to the point clouds p_i. The distance cloud-point-to-curve and curve-to-cloud-point are not the same. Including the second one (Eq. 9) makes f increase when outliers and curls appear. This is a preventive rather than corrective strategy, which ensures a correct topology and geometry of the curve C.

4.4.3 Comparative Convergence

Fig. 14 displays, for a simple curve, a comparison between convergence speeds achieved using only distance cloud-point-to-curve (Eq. 7) versus including also distances curve-to-point-cloud (Eq. 9). The two cases are called *without* and *with* penalization, respectively. Fig. 14(a) shows that using the two distances produces a faster reduction of the f factor. Fig. 14(b) indicates that the reduction in f values more monotonous when penalization is used. In other words, the optimization algorithm converges faster to the solution. Notice that the convergence when double-distance penalization is used (blue line), the number of iterations required is 60% of the ones required when only distance cloud-point-to-curve is used (green line).

Fig. 15 illustrates, for a self - intersecting curve, the comparison of convergence speeds by (1) using PCA-based initial guess for the control polygon P and by (2) abstaining from it. Fig. 15(a) indicates that, for self - intersecting curves, the usage of a PCA-based initial guess dramatically improves the convergence speed of the optimization algorithm. Fig. 15(b) also shows that the trend of the convergence is definitely more monotonous when a PCA-based initial guess is

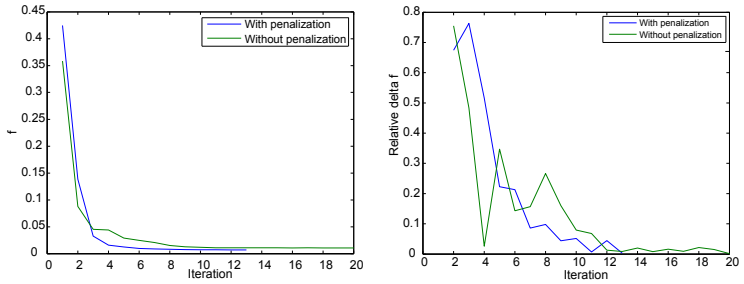


(a) Point cloud S and PCA-based initial control polygon P.

(b) Final control polygon P and its fit curve C.

(c) Objective Function f as function of iteration number.

Figure 12: Non-Nyquist point set. C fit using a PCA-based initial guess.



(a) Double-distance Influence. Simple Curve.

(b) Double-distance Influence. Simple Curve. Convergence Speed.

Figure 14: Simple Curve Case. f as a function of iteration count. Influence of double-distance penalization.

used. As in the discussion related to usage of double-distance penalization of f, in this case (blue line) the usage of PCA-based initial guess has an number of iterations being 71% of the one required (green line) when no sensible initial guess is used.

5 Complexity of the algorithm

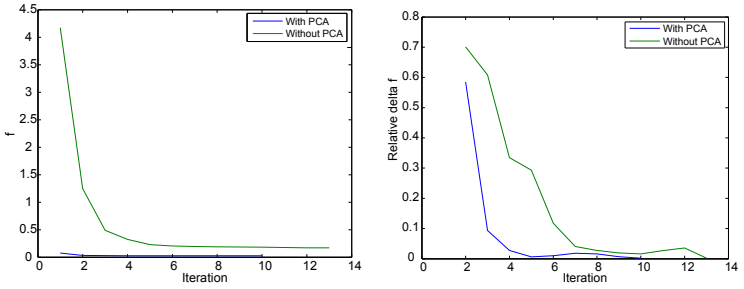
We do not report execution times because they are an unreliable indicator, wich depends on many variables (HW, SW, etc). Instead we discuss the computational expenses of our algorithm in terms of complexity analysis ($O(n)$).

Pre-processing: In Fig. 3, this stage refers to the finding of an initial quess for P (and therefore C). A sensible initial guess is found by (a) applying a PCA algorithm to find PL fragments c_i, (b) joining the disconnected c_i fragments into one (L), and (c) re - sampling the L to obtain a PL approximation for P. If we assume that the number of cloud points is n, the complexity of the ellipse-based PCA is $O(n^4)$ ((Ruiz et al., 2011)). This complexity is dominant and therefore includes the integration of PCA-based curve fragments and the curvature-based re-sample of the initial guess for P.

Optimization: In Fig. 3, this stage refers to the minimization of the f function by tunning the decision variables (i.e., control points of P) so that C fits the point cloud S. This algorithm is controlled by a `while` loop that stops when ε reaches a low value. The calculation of the Jacobian, the update (which implies a PL approximation) of the curve C and the update of the function f together cost $O(n^2)$. Therefore, the optimization stage costs $O(g(\varepsilon) * n^2)$. Since typically, $g(\varepsilon) \propto 1/(\varepsilon^2)$, the complexity of the optimization stage is $O((n/\varepsilon)^2)$.

Contributions of other authors (Piegl and Tiller, 1997; Liu et al., 2005; Wang et al., 2006; Liu and Wang, 2008; Flöry and Hofer, 2010; Saux and Daniel, 2003) typically use iterative methods for solution of simultaneous non - linear equations to find the distance point-curve. As a consequence, their computational expense is multiplied by the term $(1/\varepsilon_i^2)$, with ε_i being the error of convergence prescribed to stop. This expense is in addition to the one caused by the calculation of an initial guess for the distance point-curve. On the other hand, our method does not require the solution of non-linear equations for point-curve distance calculation, and in particular, does not need of initial guesses for such an estimation.

According to the previous discussion, the overall complexity of our algorithm in Fig. 3 is $O(n^4 + (n/\varepsilon)^2) = max(n^4, (n/\varepsilon)^2)$. Additional work is re-



(a) Initial Guess Influence. Self - intersecting curve.

(b) Initial Guess Influence. Self - intersecting curve. Convergence speed.

Figure 15: Self - intersecting Curve case. f as a function of iteration count. Influence of PCA-based initial guess.

quired to determine which one of n^4 and $(n/\varepsilon)^2$ is dominant. A valid question to be posed is whether the pre - processing might be removed, and its computational cost ($O(n^4)$) spared. The answer is negative, since a good quality initial guess for P is essential for the optimization algorithm to converge to a topologically and geometrically valid solution. Fig. 10 illustrates the result for a run lacking a good quality initial guess for P.

In the previous discussion, it must be pointed out that the complexities considered are *worst case* ones. Therefore, we present here a very conservative estimation. For complexity estimation *expected values* are also used.

6 Conclusions and Future Work

This article has presented the implementations of algorithms to synthesize a parametric planar curve C as an optimized fit for a point cloud S sampled from an unknown initial curve C_0. The curve C is therefore an approximation for C_0.

The implemented method presents several novelties with respect to the existing contributions by other authors: (1) It is successful in recovering self-intersecting and non-Nyquist curves. (2) It implements a Principal Component Analysis preprocessing which finds a topologically faithful PL approximation for P, the control polygon of C. (3) This initial guess for P is optimized in the sense of using a very reduced number of vertices, which are chosen according to the local curvature of P. (4) The algorithm avoids the expensive calculation of distance-point-curve, which implies algebraic roots, by calculating a PL approximation of curve C in each iteration and solving the distance-point-curve with this proxy approximation. (5) The algorithm penalizes f by considering not only the distances of cloud points $p_i \in S$ to C but also the distances from curve points C_i to S. Since C is finite, theses distances are not equal. By doing so, the implemented method avoids the generation of spurious curls and outlier legs. Finally, (6) the implemented algorithms lower the computation time required to solve the problem by introducing features (2), (3), (4), above.

Our approach avoids the appearance of loops and erratic excursions, while keeping the capability to reconstruct sharp features without the use of a term to penalize the curvature of the fitting curve. In this way, we can perform the reconstruction of complex data sets containing smooth and sharp features.

In the present article we use a PL approximation of a curve to estimate the point-to-curve and curve-to-point distances. In building a PL approximation $PL(C)$ of a curve C, it is mandatory to respect the Nyquist criterion. A condition sufficient to use a thinner sample of the curve occurs when a point $p_i \in PL(C)$ is nearer to a point $q \in PL(C)$ than to its $predecessor(p_i)$ or its $successor(p_i)$ in $PL(C)$. In such a case, the Nyquist criterion has been violated, and a finer re-sample is needed.

Ongoing work includes exploration of other minimization algorithms (Quasi-Newton for large values of residuals), usage of different objective functions to obtain faster convergence, improvement in self-tunning of the algorithms, usage of unequal weights in the residuals of Eq. 9 and a more aggressive strategy to minimize the number m of the control points of P. Since the search space is $\mathbb{R}^{2m}$, lowering m considerably cuts the computing time for the minimization of f.

REFERENCES

Blake, A. and Isard, M. (1998). *Active contours: the application of techniques from graphics, vision, control theory and statistics to visual tracking of shapes in motion*. Springer.

Cheng, S., Funke, S., Golin, M., Kumar, P., Poon, S., and Ramos, E. (2005). Curve reconstruction from noisy samples. *Computational Geometry*, 31(1):63–100.

Chong, E. and Żak, S. (2008). *An introduction to optimization*. Wiley-Interscience series in discrete mathematics and optimization. Wiley-Interscience.

Flöry, S. (2009). Fitting curves and surfaces to point clouds in the presence of obstacles. *Computer Aided Geometric Design*, 26(2):192–202.

Flöry, S. and Hofer, M. (2008). Constrained curve fitting on manifolds. *Computer-Aided Design*, 40(1):25–34.

Flöry, S. and Hofer, M. (2010). Surface fitting and registration of point clouds using approximations of the unsigned distance function. *Computer Aided Geometric Design*, 27(1):60–77.

Furferi, R., Governi, L., Palai, M., and Volpe, Y. (2011). From unordered point cloud to weighted b-spline: a novel pca-based method. In *Proceedings of the 2011 American conference on applied mathematics and the 5th WSEAS international conference on Computer engineering and applications*, pages 146–151. World Scientific and Engineering Academy and Society (WSEAS).

Gálvez, A., Iglesias, A., Cobo, A., Puig-Pey, J., and Espinola, J. (2007). Bézier curve and surface fitting of 3d point clouds through genetic algorithms, functional networks and least-squares approximation. In *Proceedings of the 2007 international conference on Computational science and Its applications-Volume Part II*, pages 680–693. Springer-Verlag.

Kapur, D. and Lakshman, Y. (1992). *Elimination Methods: An Introduction*, pages 45–88. Academic Press.

Lee, I. (2000). Curve reconstruction from unorganized points. *Computer aided geometric design*, 17(2):161–177.

Liu, Y. and Wang, W. (2008). A revisit to least squares orthogonal distance fitting of parametric curves and surfaces. In Chen and Juttler, editors, *Advances in Geometric Modeling and Processing*, volume 4975 of *Lecture Notes in Computer Science*, pages 384–397. Springer Berlin / Heidelberg.

Liu, Y., Yang, H., and Wang, W. (2005). Reconstructing b-spline curves from point clouds–a tangential flow approach using least squares minimization. In *Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 4–12. IEEE Computer Society.

Nyquist, H. (1928). Certain topics in telegraph transmission theory. *Bell System Technical Journal*.

Park, H. and Lee, J. (2007). B-spline curve fitting based on adaptive curve refinement using dominant points. *Computer-Aided Design*, 39(6):439–451.

Piegl, L. and Tiller, W. (1997). *The NURBS book*. Springer Verlag.

Ruiz, O. E. and Ferreira, P. (1996). Algebraic Geometry and Group Theory in Geometric Constraint Satisfaction for Computer Aided Design and Assembly Planning. *IIE Transactions. Focussed Issue on Design and Manufacturing*, 28(4):281–204. ISSN 0740-817X.

Ruiz, O. E., Vanegas, C. A., and Cadavid, C. (2011). Ellipse-based principal component analysis for self-intersecting curve reconstruction from noisy point sets. *The Visual Computer*, 27(3):211–226.

Saux, E. and Daniel, M. (2003). An improved hoschek intrinsic parametrization. *Computer Aided Geometric Design*, 20(8-9):513–521.

Shannon, C. (1949). Communication in presence of noise. *IRE*, 37:10–21.

Song, X., Aigner, M., Chen, F., and Jüttler, B. (2009). Circular spline fitting using an evolution process. *Journal of computational and applied mathematics*, 231(1):423–433.

Wang, W., Pottmann, H., and Liu, Y. (2006). Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics (TOG)*, 25(2):214–238.

Zhao, X., Zhang, C., Yang, B., and Li, P. (2011). Adaptive knot placement using a gmm-based continuous optimization algorithm in b-spline curve approximation. *Computer-Aided Design*.