Conselho Nacional de Desenvolvimento Científico e Tecnológico

Instituto de Matemática Pura e Aplicada

Computational Morphology of Implicit Curves

Luiz Henrique de Figueiredo

Tese apresentada para obtenção do título de

Doutor em Ciências

Rio de Janeiro

1992

Instituto de Matemática Pura e Aplicada—IMPA

Orientador: Jonas de Miranda Gomes

Comissão Examinadora

Pedro J. de Rezende

Marcelo Gattass

Dan Marchesin

Paulo Cezar Pinto Carvalho

Jonas de Miranda Gomes

Autor:

Luiz Henrique de Figueiredo

to Nina and Alexandre,
in memoriam.

Acknowledgements

Although this work correctly bears my name as sole author, a thesis is much more than just a text, and cannot be done in isolation. I have been chasing a doctorate for the last seven years, and I got much help along the way. The help I got from the following people and institutions had direct influence on what I finally wrote down.

Jonas de Miranda Gomes introduced me to physically-based modeling and to the use of implicit objects in solid modeling. His firm supervision was, at the same time, loose enough to suit my dispersive personality. His pursuit of excellence and his vision of IMPA as a natural place for research in computer graphics form a driving force that make us all work.

Paulo Cezar Pinto Carvalho taught me computational geometry, and suggested scattered data interpolation as a research topic. His patience with my unending flow of questions and hasty conclusions seemed infinite and was invaluable.

Dan Marchesin introduced me to simplicial methods for implicit objects, and was a reliable source of honest advice on all matters.

Marcelo Gattass taught me computer graphics, and much of everything else. That this thesis came into being is due to his continuous support throughout the last ten years.

Pedro J. de Rezende was an attentive examiner and suggested many improvements to the text.

Demetri Terzopoulos generously shared with me his thoughts on physically-based surface reconstruction, in an informal conversation during the Workshop on Geometric Modeling at IMPA, in January 1991. Experimentation with some of his ideas later resulted in the physically-based sampling method presented in Chapter 3.

Godfried Toussaint taught me computational morphology via e-mail and through many papers he kindly sent me.

Hal Sadofsky read an early draft of the text and corrected my faulty English.

The VISGRAF project at IMPA provided good computers and financial support, especially for allowing me to present part of this thesis at the Second SIAM Conference on Geometric Design, in November 1991.

IMPA provided not only a calm atmosphere in beautiful surroundings, but also a flexible structure which accepted many of my limitations, allowing me to take leave for almost two years, and then to come back to do a thesis in computer graphics. The staff at the library and at the computer laboratory were particularly helpful.

CNPq provided most of the financial support during these years.

Abstract

We divide the computation of polygonal approximations of implicit objects into two phases: sampling and structuring. Unlike classical treatments, we study each phase separately.

Classical sampling methods are reviewed, and a new sampling method is proposed; this method uses physically-based particle systems, and is more robust than classical enumeration and continuation methods.

We describe a broad taxonomy of structuring problems. According to this taxonomy, the presence of an independent structuring phase casts the modeling problem as a problem in non-parametric scattered data interpolation, which we propose to solve using computational morphology.

The use of global structuring is described in detail for curves: we prove that minimal spanning trees are good morphological tools for arcs. A couple of methods are also suggested for surfaces.

Resumo

Dividimos o cálculo de aproximações poligonais de objetos implícitos em duas fases: amostragem e estruturação. Em contraste com os tratamentos clássicos, estudamos cada fase separadamente.

Após uma revisão dos métodos clássicos de amostragem, propomos um novo método de amostragem; este método usa sistemas de partículas pseudo-físicos, e é mais robusto que os métodos clássicos de enumeração e continuação.

Descrevemos uma ampla taxonomia de problemas de estruturação. De acordo com esta taxonomia, a presença de uma fase de estruturação independente de amostragem situa o problema de modelagem como um problema de interpolação não-paramétrica de dados esparsos, que propomos resolver usando morfologia computacional.

O uso de estruturação global é descrito em detalhes para curvas: provamos que árvores geradoras minimais são boas ferramentas morfológicas para arcos. Também são sugeridos alguns métodos para superfícies.

Contents

1

Introduction

Polygonal approximations are frequently used to represent discrete versions of geometric objects inside computers. Such approximations are useful in practice not only for solving numerical problems, such as partial differential equations arising in engineering, but also for rendering, because many graphics systems have special hardware for handling polygons.

The computation of polygonal approximations of geometric objects can be conceptually divided into two phases: **sampling**, which is the computation of points on the object, and **structuring**, which is the creation of a data structure representing a polygonal approximation interpolating these points (see Figure 1).

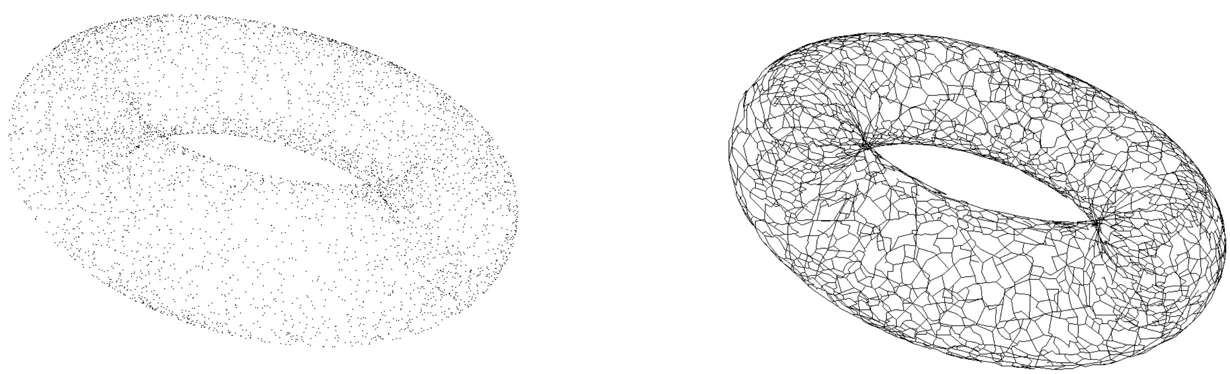

Figure 1: sampling and structuring.

The way these sub-problems are solved depends on how geometric objects are defined: the two most common ways of defining geometric objects—parametric and implicit equations—require very different methods for sampling and structuring.

For parametric curves, structuring can be done concurrently with sampling, because the structure is linear and sample points need to be processed only once. For parametric surfaces, sampling is usually done on a mesh in the parameter plane (see Figure 2). Because the structure is no longer linear, the mesh needs to be scanned twice, once in each main direction, to structure the sample into quadrilateral approximants. This technique works for parametric manifolds of higher dimension; a mesh sample is accordingly scanned several times and structured into hypercubical approximants.

The simplest meshes used in parametric sampling are regularly spaced and parallel to the coordinate axes in parameter space (see Figure 2). However, regular meshes need to be very fine to be able to capture zones of high curvature in the object. It is possible—but not easy—to adapt the sampling to the geometry of the object, thus generating meshes that are refined only around features [Vlassopoulos (1990), von Herzen–Barr (1987)].

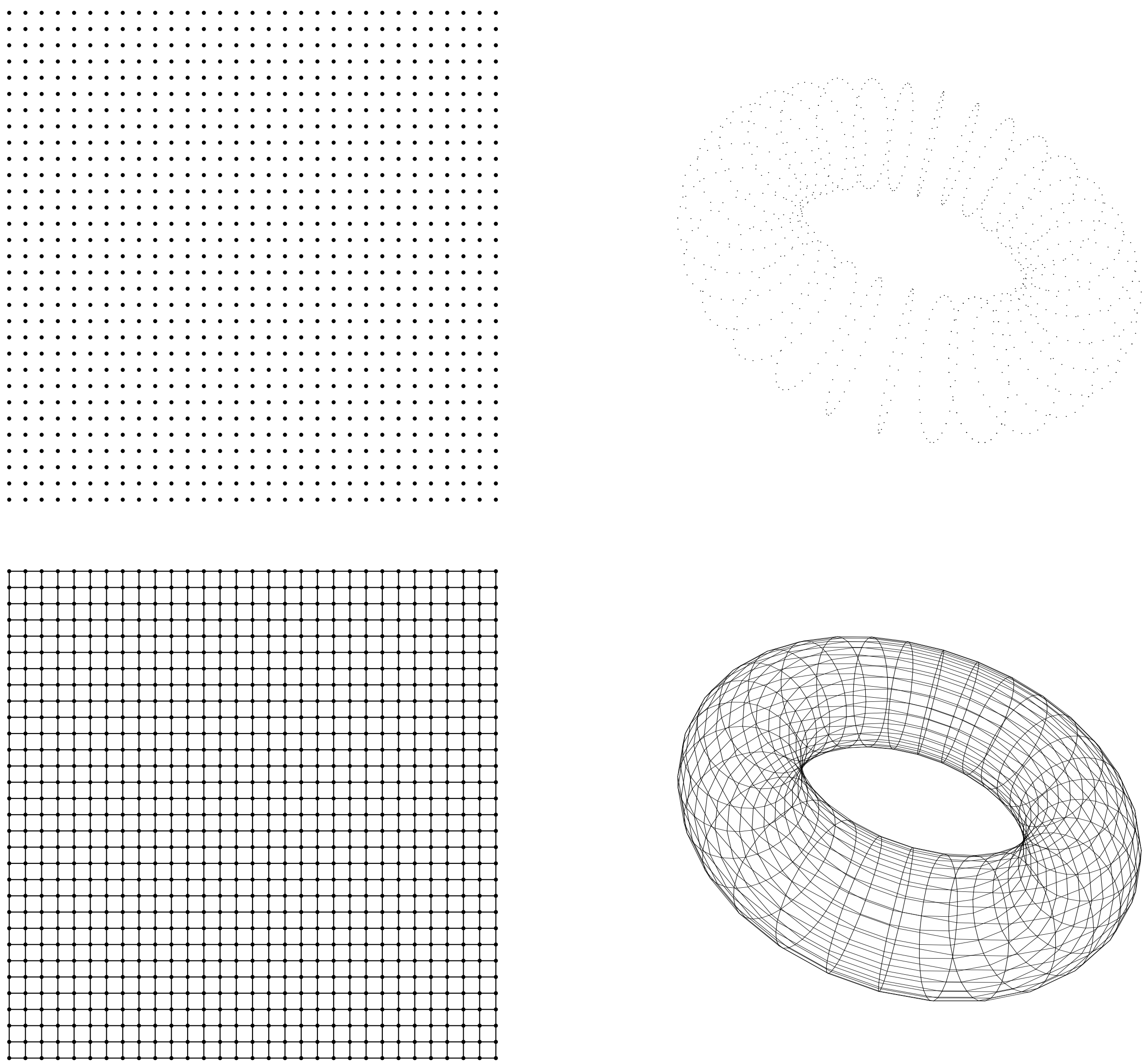

Figure 2: parametric sampling and structuring.

Therefore, constructing polygonal approximations for parametric objects is easy not only because it is easy to generate points on them, but also because it is easy to structure mesh samples: they have *a priori* structure. On the other hand, constructing polygonal approximations for objects defined implicitly is hard because both sampling and structuring are hard: sampling requires the solution of many non-linear equations, and structuring is a difficult problem because there is no guiding mesh [Allgower–Schmidt (1985)].

In contrast, the exact reverse is true for ray tracing: the intersection of rays with implicit objects can be reduced to single variable equations, which are more easily solved than the systems of equations required to compute the intersection of rays with parametric objects [Glassner (1989)]. For similar reasons, when using scan line methods, implicit surfaces are easier to render than parametric surfaces [Lane–Carpenter–Whitted–Blinn (1980), Sederberg–Zundel (1989)].

In geometric modeling, parametric objects have been traditionally used for local modeling, such as fairing and shape refinement. Global modeling, such as Boolean operations, blending, and constraint satisfaction, is more naturally done with implicit objects.

Parametrizations are very useful when they are known; however, in general, global parametrizations either do not exist or are too complicated: for instance, a sphere has no global parametrization, and general algebraic curves cannot be parametrized using rational polynomial functions only (nevertheless, they have simple approximate algebraic parametrizations [Waggenspack–Anderson (1989)]). Algebraic parametrizations can always be transformed into implicit equations by using classical elimination theory, at the cost of degree explosion: for instance, the implicit equation for a general bicubic patch has degree 18 and 1330 terms [Sederberg (1987)].

This conflict between the use of parametric and implicit methods for defining objects in geometric modeling is similar to the conflict between constructive and boundary representation schemes [Hoffmann (1989)]. One solution to this conflict is to acknowledge that hybrid modeling cannot be done seamlessly: parametric and implicit methods should not compete at the same level of modeling, but rather provide alternative definition methods. The challenge in hybrid modeling is the integration of different definition methods into a functional system.

The thesis

Polygonal approximations for implicit objects are classically computed with methods that combine sampling and structuring (see Chapter 3). In this thesis, we study sampling and structuring as independent problems. This separation aims to identify the problems which are particular to each phase. When structuring is done concurrently with sampling, these problems tend to lose their original source and merge into a set of problems that is characteristic of the combined method used. The goal in separating structuring from sampling is not to claim that the computation of polygonal approximations of geometric objects should be done in two phases, although we propose such a method for curves in Chapter 5.

The main problem in structuring is to give structure to a set of points, regardless of how they were generated [Boissonnat (1984)]. In this sense, the problem is not so much giving a structure, but rather finding the right structure—a typical problem in a new area that has been tentatively called "reconstruction from unorganized data" [Hoppe–DeRose–Duchamp–McDonald–Stuetzle (1991)] "non-parametric scattered data interpolation" (see Chapter 4), and "computational morphology" [Toussaint (1988)].

The thesis is organized as follows: this introduction is followed by a brief review of the classical mathematics of implicit objects and their use in solid modeling. Chapter 3 discusses sampling by first reviewing three classical methods, and then describing a new sampling method that uses physically-based particle systems. The samples obtained by this method are not based on scanning cellular decompositions, but are instead "randomly" generated by simulating physical motions of particles. Chapter 4 discusses structuring in the light of the use of computational morphology in non-parametric scattered data interpolation. A taxonomy of the problems in this area is presented. This chapter also introduces minimal spanning trees as our main structuring tool, in the context of connected component identification. Chapter 5 draws on these two chapters to propose a method for modeling implicit curves and surfaces. The final chapter contains our conclusions and suggests directions for future research.

The main original contributions of the thesis are:

- a physically-based sampling method for implicit objects, presented in Chapter 3;
- a taxonomy of structuring problems, presented in Chapter 4;
- a proof that minimal spanning trees are good morphological tools for arcs,
- a robust method for modeling implicit curves, and
- two new ideas for modeling implicit surfaces, all presented in Chapter 5.

2

Background

In order to establish the terminology used, this chapter reviews the classical mathematics of implicit objects and their use in solid modeling.

Implicit objects

We say that an object is **given implicitly** when it is the set of points that are the common zeros of some real functions. More precisely, we say that $V \subseteq \mathbf{R}^n$ is given implicitly by functions $h_1, h_2, \ldots, h_m \colon \mathbf{R}^n \to \mathbf{R}$ when V is the set of all points $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$ satisfying

$$
\begin{aligned}
h_1(x_1, \ldots, x_n) &= 0, \\
h_2(x_1, \ldots, x_n) &= 0, \\
&\vdots \\
h_m(x_1, \ldots, x_n) &= 0.
\end{aligned}
$$

For simplicity, all functions are supposed to be **smooth** or **differentiable**, in the sense that they have continuous partial derivatives of all orders.

To simplify the notation, define a function $h \colon x \in \mathbf{R}^n \mapsto (h_1(x), h_2(x), \ldots, h_m(x)) \in \mathbf{R}^m$. Then the set V, given implicitly by functions $h_1, h_2, \ldots, h_m$, is the solution of the equation $h(x) = 0$. The **inverse image** of a point $y \in \mathbf{R}^m$ is the solution of the equation $h(x) = y$ and is denoted by $h^{-1}(y)$; the set V is $h^{-1}(0)$, the inverse image of zero.

We say that a point $x \in \mathbf{R}^n$ is a **regular point** of h if the derivative of h at x is surjective. This means that the Jacobian matrix of h at x has maximal rank m. Regular points can only exist if $n \geq m$, i.e., if there are at least as many unknowns as equations in the system above; we shall assume that this is the case. A point $y \in \mathbf{R}^m$ is a **regular value** of h if all points in $h^{-1}(y)$ are regular. Note that regular values need not be values of h: points outside the image of h are automatically regular. Points which are not regular are called **singular** or **critical**.

There are many classical theorems relating the set V to the function h. For our purposes, the most fundamental of these is the **implicit function theorem**, which relates the local topology of V to the partial derivatives of h: it says that V is locally parametrizable around regular points. More precisely, the implicit function theorem says that, around regular points, the equation $h(x_1, \ldots, x_n) = 0$ can be solved locally for m of the variables in terms of the remaining $n - m$ variables, thereby making explicit the implicit relation given by $h(x) = 0$ [Spivak (1965), Lima (1981)].

A consequence of the implicit function theorem is that if $0 \in \mathbf{R}^m$ is a regular value in the image of h, then V has local parametrizations at each of its points and hence is a differentiable manifold of dimension $n-m$ and **co-dimension** m. Taking inverse images of regular values is an important way of constructing manifolds. A classical theorem by Sard guarantees that this construction can be done frequently because almost all values are regular, in the sense that they form an open and dense set [Milnor (1965)]. This important theoretical result is also valuable from a computational point of view: it is important to know that values near regular values are regular, not only because of the errors inherent to floating point computations, but also because no robust algorithm is known for deciding whether a given value is regular or not [Yomdin (1990)].

Manifolds of co-dimension 1 are called **hypersurfaces**; they will be the main object of study in this work. From now on, **dimension** will mean the dimension n of the **ambient space** $\mathbf{R}^n$, which is the domain of h.

An implicit hypersurface is the set of zeros of a single real function h. This function defines a scalar field on the ambient space, which may be interpreted as temperature or, in dimension 2, as height measured from a fixed horizon. For this reason, a set of points on which h is constant is called a **level set** of h; it is also called a **contour** of h. Sometimes, the function h is called a **potential function**, and level curves are called **equipotential lines**. A level set is called regular when all its points are **regular**; otherwise, it is called **singular**.

Every implicit object can be given as a level set: if V is given implicitly by $h_1, h_2, \ldots, h_m$, then V is also given implicitly by $h = h_1^2 + h_2^2 + \cdots + h_m^2$. However, even if V is a manifold, $0 \in \mathbf{R}$ is never a regular value of h because the gradient of h is identically zero on V. Nevertheless, we shall assume throughout that all implicit objects are given by a single equation, even if they are not hypersurfaces.

Implicit objects in solid modeling

Implicit surfaces (hypersurfaces in dimension 3) are very useful in solid modeling because the interior, the boundary, and the exterior of solids can be conveniently associated to the sign of real functions on $\mathbf{R}^3$. For example, the sign of the function $h(x,y,z) = x^2+y^2+z^2-1$ defines the interior, the boundary, or the exterior of a solid sphere.

Moreover, the results of Boolean operations on implicit objects can also be expressed implicitly: if U is given implicitly by g and V is given implicitly by h, then the union $U \cup V$ is given implicitly by gh and the intersection $U \cap V$ is given implicitly by $g^2 + h^2$, as seen in the previous section. Strictly speaking, the complement $\mathbf{R}^n \setminus V$ cannot be given implicitly because it is an open set, whereas implicit objects are closed. Nevertheless, $\mathbf{R}^n \setminus V$ is the projection on $\mathbf{R}^n$ of the object given implicitly in $\mathbf{R}^{n+1}$ by the equation $h(x_1, \ldots, x_n)x_{n+1} = 1$.

One of the drawbacks of these expressions for Boolean operations is that they are suitable for boundaries but not for interior or exterior. For instance, the interior of $U \cap V$ cannot be associated to the sign of $g^2 + h^2$ because this expression is never negative. Nevertheless, as noted in the previous section, this expression shows that implicit objects of any codimension can be written as the intersection of hypersurfaces.

One way to avoid these difficulties is to define solids as the solution of $h(x) \leq 1$ for positive real functions h, instead of defining solids as the solution of $h(x) \leq 0$ for arbitrary real functions h. This keeps the interior, the boundary, and the exterior of solids associated to the sign of real functions ($h - 1$, in this case). Boolean operations on solids are now done as follows: the union $U \cup V$ is given implicitly by $\min(g, h)$, the intersection $U \cap V$ is given implicitly by $\max(g, h)$, and the complement $\mathbf{R}^n \setminus V$ is given implicitly by $1/h$. These expressions, in contrast with the ones presented earlier, are not smooth. Nevertheless, they can be arbitrarily approximated by smooth functions: for instance, $\max(g, h)$ can be approximated by $(g^p + h^p)^{1/p}$ for $p > 1$, in the sense that

$$\lim_{p \to \infty} (g^p + h^p)^{1/p} = \max(g, h),$$

for many important function norms [Ricci (1973), Gomes–Tavares (1989)].

Thus, implicit surfaces form a good set of primitives for CSG systems [Requicha (1980), Hoffmann (1989)]. More sophisticated modeling, such as blending, can also be done implicitly [Hoffmann–Hopcroft (1985), Rockwood–Owen (1987)]. Furthermore, design constraints are naturally expressed as additional equations [Barr–Barzel (1988)]. Despite the usefulness of implicit objects in solid modeling, the interest in the computational aspects of implicit manifolds is very recent [Bloomenthal (1988), Allgower–Georg (1990)].

3

Sampling

Let h be a differentiable real function on $\mathbf{R}^n$ defining an implicit object $V = h^{-1}(0)$. Sampling points on V means finding solutions of the equation $h(x) = 0$. In practice, sampling means finding enough solutions so that the topology of V can be reconstructed and the geometry can be approximated.

A sampling method can impose a structure that does not correspond to the geometry of the object being sampled [Boissonnat (1984)]. For instance, sampling an object by slices, as done in computer-aided tomography, impose a sweep structure on the object; accordingly, reconstructing a solid from a series of slices is a difficult problem [Giertsen–Halvorsen–Flood (1990)].

Classical methods compute polygonal approximations for implicit objects by performing structuring concurrently with sampling. As mentioned in Chapter 1, we prefer to study sampling and structuring separately. This chapter reviews the sampling technique of three classical polygonization methods and then describes a new, physically-based approach to sampling. The problems of structuring are discussed in the next chapter.

Sampling by ray-casting

A naive way of finding solutions of the equation $h(x_1, \ldots, x_n) = 0$ is to reduce it to single-variable equations by computing the intersection of V with a family $\mathcal{R}$ of straight lines, which we shall call **rays**.

The simplest rays correspond to fixing some of the variables. For instance, we could fix the first $n-1$ variables and solve $h(a,t) = 0$ for a sample of points $a = (a_1, \ldots, a_{n-1})$ in $\mathbf{R}^{n-1}$. This computes the intersection of V with the "vertical" rays $\{a\} \times \mathbf{R}$; accordingly, we call this method **vertical ray-casting** (see Figure 3).

Sampling by ray-casting needs the solution of many single-variable equations. Even if a good equation solver is available, there are several problems with this approach, showing how the sample obtained on V will depend on the sample of rays $\mathcal{R}$ (see Figure 3):

- because there is no *a priori* criterion for choosing rays intersecting V, a majority of the rays in $\mathcal{R}$ may not contribute sample points on V;
- even when a ray intersects V, the equation solver may not find all intersections, resulting in a partial sample of V;
- several rays in $\mathcal{R}$ may intersect V at the same point (this is is not a problem if the rays in $\mathcal{R}$ are parallel, as in vertical ray-casting).
- a common choice for vertical ray-casting is to sample $\mathbf{R}^{n-1}$ on a uniform mesh, as in parametric sampling (see Chapter 1). The resulting sample on V is biased against regions where the tangent space of V is vertical.

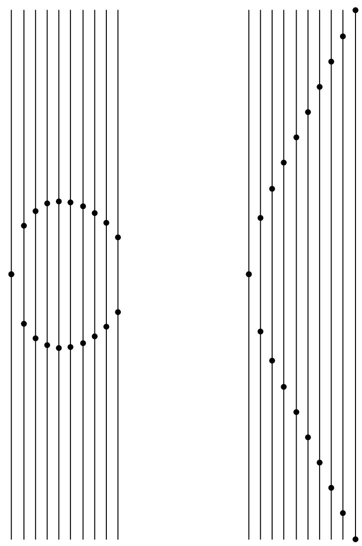

Figure 3: ray-casting sampling for $h(x,y) = y^2 - x^3 + x$.

In general, it is too hard to relate the size and density a sample of rays $\mathcal{R}$ to the size and density of the corresponding sample of V. Nevertheless, as mentioned in Chapter 1, vertical ray-casting is useful for rendering implicit surfaces.

Sampling by continuation

The **gradient** of h,

$$\nabla h = (\frac{\partial h}{\partial x_1}, \ldots, \frac{\partial h}{\partial x_n}),$$

is the Jacobian matrix of h in co-dimension 1. The gradient is a vector field on the ambient space $\mathbf{R}^n$, which, at the regular points of h, points in the direction of local growth of h. Moreover, ∇h is orthogonal to the level sets of h.

In dimension 2, the associated **Hamiltonian** vector field $\mathcal{H}(h) = (-\partial h/\partial y, \partial h/\partial x)$ is orthogonal to the gradient $\nabla h = (\partial h/\partial x, \partial h/\partial y)$, and is therefore tangent to the level curves of h. Thus, the level sets of h are the integral curves of $\mathcal{H}(h)$, and hence can be traced by solving an ordinary differential equation (see Figure 4). The integration of ordinary differential equations related to the gradient is also the main theme of the physically-based sampling method described later in this chapter.

To cast the computation of a level curve as an initial value problem, a point on the curve is needed. Actually, a point on each connected component of the level curve is needed, if all components are to be found. In this case, for each such point (x_0, y_0), the corresponding connected component is the solution of the following Cauchy problem:

$$\begin{aligned} \frac{dx}{dt} &= -\frac{\partial h}{\partial y} & \qquad x(0) &= x_0, \\ \frac{dy}{dt} &= \frac{\partial h}{\partial x} & \qquad y(0) &= y_0. \end{aligned}$$

Several classical numerical methods exist for solving initial value problems [Forsythe–Malcolm–Moler (1977), Lambert (1973)]. Although they are well-known and easily implemented, these general methods cannot exploit the fact that the solution of the Cauchy problem above is a level curve of a smooth function. One way to use this additional information to help stabilize the integration is to do a few iterations of Newton's method for the solution of non-linear equations as a correction after each prediction [Allgower–Georg (1990)]: if p is the current point on the curve, then Euler's prediction of the next point is $p \leftarrow p + \delta\mathcal{H}(h)(p)$, which is possibly on another level curve; we use Newton's method to bring the point p back to the correct level along the straight line orthogonal to $\nabla h(p)$ (see Figure 4).

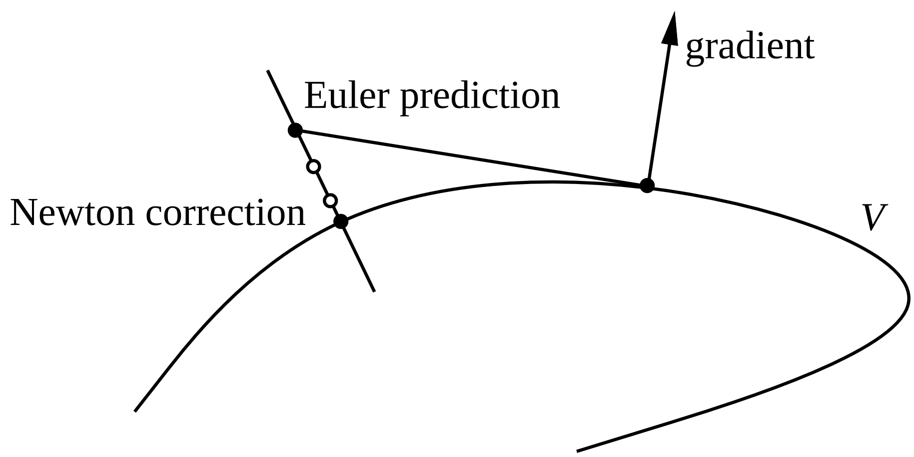

Figure 4: sampling by continuation using Euler predictors and Newton correctors.

Thus, we have the following **continuation** algorithm:

$$
\begin{array}{ll}
p \leftarrow p + \delta\mathcal{H}(h)(p) & \textit{Euler predictor} \\
u \leftarrow \nabla h(p) & \\
\text{while } |h(p)| > \varepsilon & \\
\quad p \leftarrow p - \dfrac{h(p)}{\langle \nabla h(p), u\rangle} u & \textit{Newton corrector}
\end{array}
$$

The combination of single-step predictors (such as Euler's) and Newton correctors provides a robust method for tracing level curves in several circumstances, despite the following restrictions:

- it is only applicable to plane curves;
- it needs starting points on each connected component;
- it needs special care with closed orbits.

It is possible to extend the continuation method described above to higher-dimensional manifolds by starting with an orthogonal complement of the gradient and carefully integrating along each vector in this complement; this is called the **moving frame** method [Allgower–Georg (1990)].

Sampling by enumeration

Instead of computing an approximation of the level sets of the exact h, we can compute the exact level sets of an approximation of h. If this is to be easier than dealing with the exact h, then the approximation should be sufficiently simple so that its exact level sets are easy to compute. This is achieved by taking a piecewise smooth approximation that is very simple on each piece: the pieces are the cells of a cellular decomposition of the ambient space, and the most common approximations are piecewise linear, obtained by scanning the decomposition and computing the intersection of the level set with each cell. The need for starting points is thus avoided and all connected components are found with no special processing. Methods that solve equations by scanning cellular decompositions are called **enumeration** methods; when the cells are simplices—a very common choice—they are called **simplicial** methods (see Figure 5).

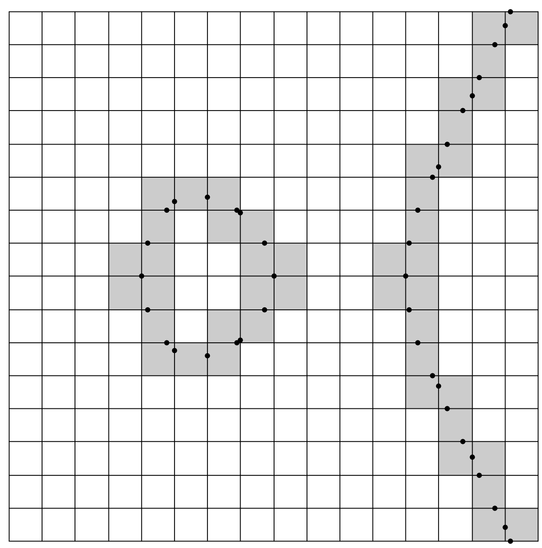

Figure 5: sampling by enumeration.

The simplest decompositions are regular, obtained by translating and scaling a single prototype cell, e.g., a unit hypercube aligned with the coordinate axes. The next level of complexity allows rotations of the prototype cell, as in regular triangular decompositions. In principle, the geometry of the cells could be arbitrary, but if a cell has many facets, then there are many possibilities for its intersection with the level set. Moreover, if the cells have complex geometry, then the decomposition has a complex topology, making it harder to coordinate the scan. In practice, only hypercubical or simplicial cells are used, arranged in cellular decompositions having well understood combinatorics [Allgower–Schmidt (1985), Loreson–Cline (1987), Hall–Warren (1990)].

As usual, tolerance is a problem and the size of the cells must be carefully chosen to avoid missing features because of undersampling. However, choosing a very small cell size will greatly increase the number of cells to be scanned. Thus, regular cellular decompositions rapidly increase in complexity under the demands of precision. One attempt to overcome this problem is to exploit the geometry of the object and make adaptive cellular decompositions, which reduce the total number of cells to be scanned by concentrating small cells around features [Bloomenthal (1988), Velho (1990), Hall–Warren (1990)].

Independently of how the cells are chosen, the sample provided by a scan is only structured locally at each cell; global structuring, such as the correct glueing of pieces and the identification of connected components, must be done *a posteriori*. This is an additional reason for requiring simple decomposition topology.

Enumeration methods can be very expensive because only a few cells intersect the object, especially in high co-dimension. Nevertheless, enumeration methods are applicable not only to hypersurfaces but to submanifolds of all co-dimensions. However, the higher the co-dimension, the fewer the cells intersected by the object. This is a serious problem because, when the cells are simplices, the total number of cells grows exponentially with the dimension: a simplicial decomposition of a hypercube in dimension n needs $\Omega(c^n\sqrt{n!})$ simplices [Haiman (1991)].

Piecewise linear approximations for implicit objects can be found by a continuation variant of the enumeration method that follows the solution at each of the intersecting cells and only at those, by using "pivoting" procedures to choose the cells to be scanned. This method provides global structuring concurrently with sampling; however, it brings back the need for starting points on each connected component. Moreover, it is now necessary to keep track of all visited cells, in order to identify closed components. Despite these restrictions, piecewise linear continuation methods are very successful, and widely used in practice [Allgower–Georg (1990)].

Physically-based sampling

The main theme of our physically-based method for sampling an object V given implicitly by a function h is the integration of ordinary differential equations related to the gradient vector field ∇h.

Unlike continuation methods, which integrate orthogonal complements of the gradient, and need starting points correctly placed on each connected component of V, our method integrates differential equations based on a modified gradient field, and can start at an arbitrary sample of points on the ambient space: the ω-limit of the orbit of each point will be on V, and will provide the desired sample.

The modified gradient vector field we shall use is $F = -\mathrm{sign}(h)\nabla h$, obtained by reversing the sense of ∇h when h is positive: this provides a vector field pointing locally in the direction of V (see Figure 6). The attractors of F are the set V and the points where h has a positive local minimum. Because we are mainly interested in V, we shall call these latter points **spurious attractors**. Note that spurious attractors need not be isolated points: in general, they will be a submanifold of V. Although some sample points can land on spurious attractors, this does not seriously affect the sampling because we can discard points where h is not zero.

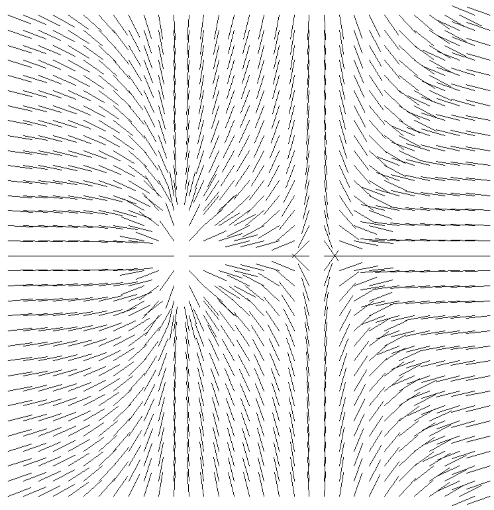

Figure 6: modified gradient vector field.

The field F corresponds to the potential function $U = |h|$. The points of global minimum potential energy are exactly those on V. However, the points where h has a positive local minimum are also local minima of the potential energy; spurious attractors correspond to particles that get trapped at these local minima.

The potential function U is not smooth on V, where $h = 0$. This implies that the field F is not continuous at regular points of V; in fact, the field F is continuous on V only at singular points. One way to avoid this discontinuity is to consider h^2 instead of h: the two functions have the same set of zeros but the modified gradient for h^2 is $-\mathrm{sign}(h^2)\nabla h^2 = -2h\nabla h$, which is continuous everywhere. However, the convergence of the numerical methods used for integration is slower for h^2 than it is for h. Moreover, the field for h^2 has more spurious attractors than the field for h because all local extrema of h^2 are positive.

We consider two physical interpretations for the vector field F; they provide autonomous dynamical systems that are integrated to simulate Newtonian mechanics [Synge–Griffith (1949), Hirsch–Smale (1974), Perko (1991)]. Discrete models for physical systems are well suited for computer simulation [Greenspan (1973)] and have recently been successfully used in geometric modeling [Terzopoulos–Fleischer (1988), Szeliski–Tonnesen (1991)] and mesh generation for finite-element analysis [Gossard (1991)].

The first physical interpretation is **kinematic**: F describes velocity in terms of position. The equation of motion corresponding to this interpretation is

$$\frac{dx}{dt} + \mathrm{sign}(h)\nabla h = 0.$$

The other interpretation is **dynamical**: F is a force field in a dissipative medium. A particle released at rest into this medium is subjected to forces induced by F which make it move towards V and then oscillate around it. Adding friction to the movement guarantees that the particle will tend to equilibrium at a point on V. The resulting equation of motion for a unit mass particle is then

$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \mathrm{sign}(h)\nabla h = 0,$$

where γ is a positive real number representing friction proportional to the velocity. The presence of friction is important because, when $\gamma = 0$, this second-order dynamical system can have additional spurious attractors, other than the ones described above. Incerti, Parisi and Zirilli (1979) have proposed a similar differential equation for finding zeros of functions $\mathbf{R}^n \to \mathbf{R}^n$; however, they were interested in finding any one solution, not many, as required in sampling for geometric modeling.

By releasing a large set of randomly placed particles into either field and simulating the corresponding physics, we can generate a "random" sample of points on V (see Figure 7). However, there seems to be no clear relation between the initial position of the particles and the final distribution of points on V, in the sense that we do not know how to predict where a particle will land on V, even though we are dealing with deterministic systems. Nevertheless, points tend to concentrate near high curvature and singularities. On the other hand, no portion of V is consistently missed by this process, in the sense that every open subset of V is contained in the ω-limit of some open subset of $\mathbf{R}^n$.

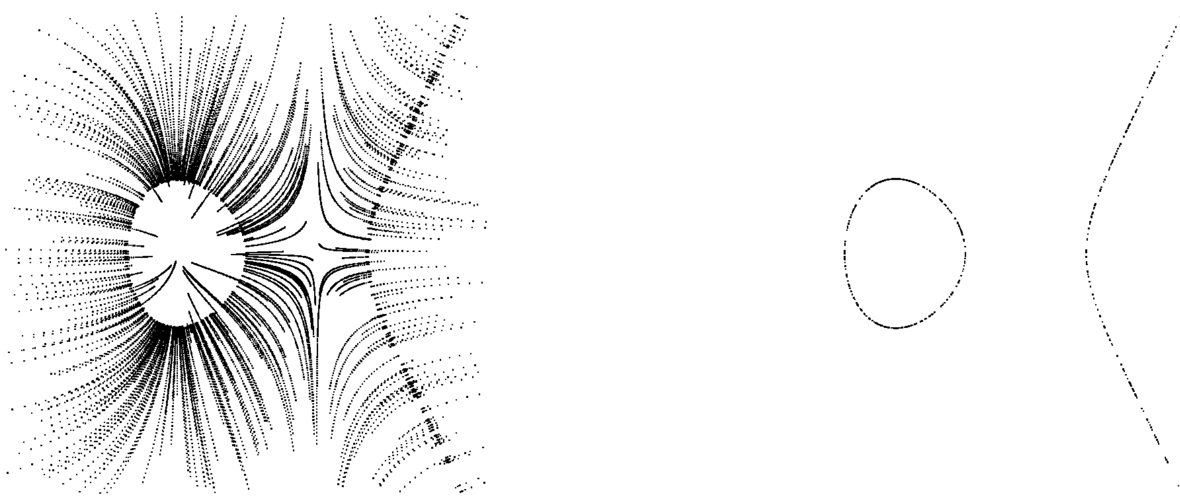

Figure 7: orbits and final sample.

The equations of motion for each particle are solved numerically, using one of the classical numerical methods, such as those of Euler or Runge–Kutta. However, if particles are to approach V steadily, adaptive variations of these methods are required, in which step control is used to avoid divergent oscillations. Such oscillations are mainly due to the discretizations used in the numerical methods, although they are also inherent to the physics in the dynamical case: indeed, whereas the orbits in the kinematic case do not cross V and oscillations are the artifact of numerical methods, in the dynamical case, particles approach

V with non-zero velocity, causing them to cross V and only then being forced back to it because the force field changes sign. Controlling integration step size causes such particles to approach V more carefully each time they cross it, thus guaranteeing convergence. On the other hand, approaching attractors with non-zero velocity can actually help a particle to avoid spurious attractors.

Note that the equations of motion are uncoupled and hence can be solved in parallel. In particular, it is not necessary to keep all particles in a single time frame, and we may use a different step size for each particle. This contrasts with other particle systems used in computer graphics and animation, where the joint movement of the particles is important for realism [Reeves (1983)]. Therefore, we can implement independent step control by making the step size of a particle depend on its position; we do this by halving the step size every time the particle crosses V. Moreover, we can stop the simulation of a particle's movement as soon as it has reached a desired level of equilibrium (measured, for instance, by the value of h at the position of the particle or by the value of its the step size).

The following algorithm combines a classical single-step Euler integration with step control as described above, to provide a simple and robust method for the integration of the equations of motion in the dynamical case. If x is the position of a particle, v is its velocity, and δ is its current step size, then its next position, velocity and step size are computed as follows:

$$
\begin{array}{ll}
y \leftarrow h(x) & \textit{remember current level} \\
v \leftarrow v + \delta(F(x) - \gamma v) & \textit{Euler predictor for velocity} \\
x \leftarrow x + \delta v & \textit{modified Euler predictor for position} \\
\text{if } \operatorname{sign}(y) \neq \operatorname{sign}(h(x)) & \textit{check crossing} \\
\qquad \delta \leftarrow \delta/2 & \textit{step control} \\
\qquad v \leftarrow 0 & \textit{re-start from rest}
\end{array}
$$

Note that this algorithm is a variant of the strict Euler method, in the sense that the predicted value of v is used to predict x, instead of predicting both in parallel. Step control is done when h changes sign at two consecutive positions: it is assumed that the particle has crossed V at this time. As a further refinement, we re-start a particle from rest after such crossings; this allows oscillating particles to approach V very carefully.

The corresponding algorithm for the kinematic case is simply:

$$
\begin{array}{ll}
y \leftarrow h(x) & \textit{remember current level} \\
x \leftarrow x + \delta F(x) & \textit{Euler predictor for position} \\
\text{if } \operatorname{sign}(y) \neq \operatorname{sign}(h(x)) & \textit{check crossing} \\
\qquad \delta \leftarrow \delta/2 & \textit{step control}
\end{array}
$$

Although both prediction and step control are very simple, these algorithms perform well in practice. The only real problem is the choice of initial step size. In our explorative implementation, this choice is made by trial and error, but it should be possible to estimate a good step size based on the initial position. A naive estimate would be the value of h at the point, but this is not a good estimate if h is flat near this point (see Figure 8). Better estimates would take the gradient of h into account, in a way similar to Newton's method.

See Chapter 5 for a fuller discussion on the use of this sampling method in practice.

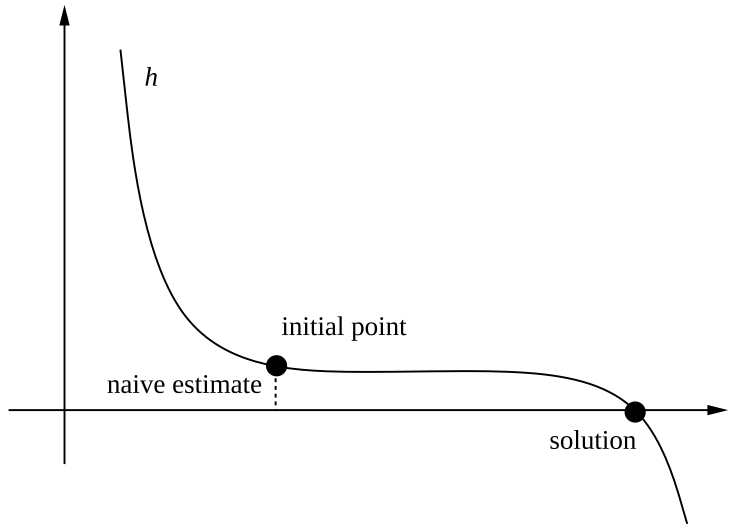

Figure 8: naive estimate of initial step size.

Although both methods integrate ordinary differential equations, the sampling method we have described is more robust than the continuation methods described earlier because:

- continuation methods need starting points on V, whereas arbitrary samples are adequate as initial conditions for our method;
- care must be taken when integrating the Hamiltonian in order to stay on level 0: single-step numerical methods without correction tend to jump to other levels. On the other hand, our method starts at arbitrary positions in the ambient space and aims for a submanifold—an easier task than having to start on a submanifold and trying to stay on it. Level jumping is no longer a problem because all trajectories lead to V; accordingly, single-step integration methods are adequate.
- our method is naturally parallel;
- our method does not need a predefined region of interest, because the particles will track V wherever it is in the ambient space.

The two interpretations used in our sampling method have complementary advantages. In the kinematic case, the differential equation is simpler, and the convergence is faster. Moreover, particles approach V orthogonally. On the other hand, in the dynamical case, spurious attractors are more easily avoided, making the sampling more robust. Moreover, convergence can be controlled by modifying the value of friction or by re-starting particles from rest when they seem not to be converging to equilibrium.

4

Structuring

Having sampled points on an implicit object, we need to structure the sample into geometrically meaningful descriptions, from which true geometrical models can be constructed.

This chapter begins by proposing a broad taxonomy of structuring problems. A discussion of the main issues and techniques in computational morphology will then lead to the identification of connected components, in preparation for the structuring of samples on implicit curves and surfaces.

A taxonomy of structuring problems

The structuring problem we face after sampling can be stated as follows:

Find a computational description of a geometric object from a finite set of points on it.

Clearly, the geometric correctness of a computational description will depend on the density and distribution of the sample.

If we make no assumptions about the sample—other than the geometrical location of the points—then we have a problem that we shall call **zero knowledge structuring**. This problem is not as hopeless as it might seem at first: at least, it has perceptual solutions, if the sample has an inner structure and is sufficiently dense (see Figure 9). Nevertheless, our main interest is not in perceptual solutions, but rather in computationally feasible solutions that are perceptually meaningful. There is some indication that minimal structures interpolating the data, such as polygons of minimum perimeter or polyhedrons of minimum area, have perceptual meaning [Zahn (1971), O'Rourke (1981)]. However, most minimal structures are not computationally feasible: for instance, finding a closed polygon of minimum perimeter is the famous traveling salesman problem, which is NP-hard [Garey–Johnson (1979)]. Moreover, minimal structures tend not to vary continuously with the geometry of the sample, being too sensitive to the actual position of each point [Boissonnat (1984)].

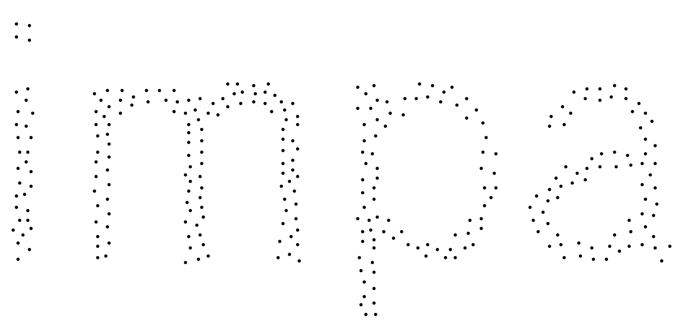

Figure 9: zero knowledge structuring and perceptual solutions.

If the sample is assumed to lie on a submanifold of the ambient space, then we shall call the problem **scattered data interpolation**. The problem traditionally known by this name is actually more restricted, because it assumes that the points are on the graph of a bivariate real function [Schumaker (1976), Pratt (1985)]. Accordingly, when the points are assumed to be on the graph of a function, whose domain is a computable manifold known *a priori*, then we shall call the problem **parametric scattered data interpolation**. In the classical case, the domain is a subset of the Euclidean plane $\mathbf{R}^2$ embedded in $\mathbf{R}^3$, typically the convex hull of the projection of the sample onto the xy-plane (see Figure 10). Other manifolds, such as the two-dimensional sphere, have only recently been considered as domains for parametric scattered data interpolation; this problem has been called **surfaces on surfaces** [Lawson (1984), Foley–Lane–Nielson–Franke–Hagen (1990)].

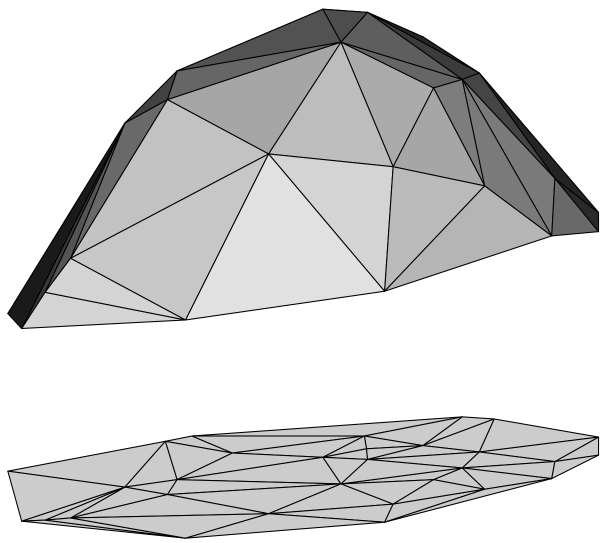

Figure 10: parametric scattered data interpolation over $\mathbf{R}^2$.

One important requirement of parametric problems is that the domain must be computable in the sense that it must be possible to compute the point in the domain corresponding to each sample point; we shall call this computation the **parametric projection**.

If no suitable domain is known—e.g., when the sample lies on an implicit object—then we shall call the problem **non-parametric scattered data interpolation**. For samples on two-dimensional surfaces in three-dimensional space, this problem has been called **surface reconstruction from unorganized point sets** [Hoppe–DeRose–Duchamp–McDonald–Stuetzle (1991)]. Note that some non-parametric problems admit parametric solutions: for instance, structuring samples on the "horizontal" parabola $x = y^2$ does not admit a solution parametrized by the x-axis, but it does admit a solution parametrized by any non-horizontal line, the natural one being the y-axis. In general, however, non-parametric problems do not admit a natural parametric solution: for instance, the only natural parametric solution for a sample on a circle is the circle itself. One of the main problems in non-parametric scattered data interpolation is precisely finding which—if any—are the natural parameters for the object. In this sense, we see non-parametric scattered data interpolation as a problem in computational morphology (see next section).

Even for plane curves, which are submanifolds of dimension 1 in dimension 2, non-parametric interpolation is a difficult problem (see Chapter 5), whereas parametric interpolation reduces to sorting along the parameter axis (see Figure 11). Research in one-dimensional parametric interpolation is consequently concentrated on choosing interpolating functions that are better than piecewise linear; thus the wealth of literature on spline approximation.

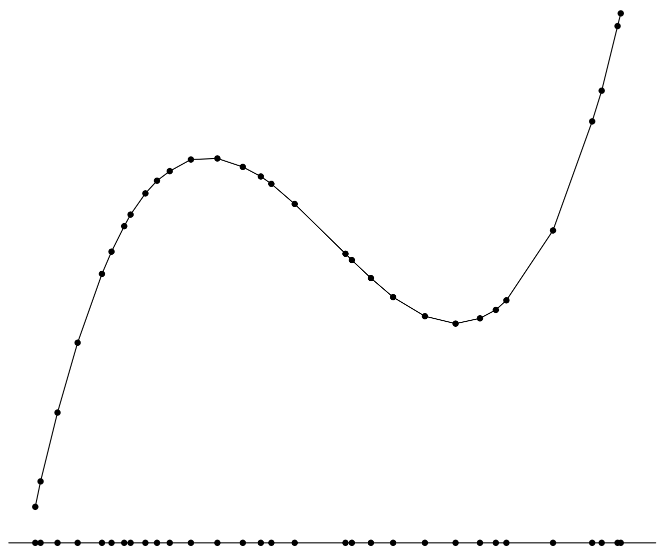

Figure 11: one-dimensional parametric scattered data interpolation.

For submanifolds of higher dimension, finding interpolants with good properties is still the main problem in parametric interpolation [Franke (1982), Schumaker (1976)], even though structuring is not entirely trivial because some kind of triangulation of the domain must be chosen (see Figure 10). The challenge is to find triangulations on which good interpolants can be based [Dyn–Levin–Rippa (1990), Brown (1991)]: for instance, Delaunay triangulations are optimal for piecewise linear interpolants, with respect to several natural criteria [Omohundro (1990), Rippa (1990)]; the triangulation of the domain in Figure 10 is a Delaunay triangulation.

Parametric and non-parametric scattered data interpolation further differ in their density requirements for samples: samples need to be denser and more uniformly distributed for non-parametric interpolation than for parametric interpolation. This is a consequence of the additional structure present in parametric scattered data interpolation.

The taxonomy given above is intentionally broad. It can be expanded to include problems with varying amounts of *a priori* structure: such information is likely to be very useful in choosing a solution strategy. We describe some typical examples below.

The domain of a parametric scattered data interpolation problem is usually—but not necessarily—embedded in the same space as the data points. Classical scattered data interpolation is an example of such problems, where, moreover, the parametric projection is a geometric orthogonal projection. Problems in this class could be called **embedded parametric scattered data interpolation**.

On the other hand, knowing the dimension of the target manifold in a non-parametric scattered data interpolation problem is not a requirement. The class of problems where this dimension is known to be d and the data points are in $\mathbf{R}^n$ could be called d-n **non-parametric scattered data interpolation**. If the dimension of the target manifold is not known, then it can be estimated from the data [Schwartzmann–Vidal (1975)].

The obvious additional information that we have for implicit objects is the tangent plane at each data point, given as an orthogonal complement of the gradient. We could call problems with tangent or normal data **oriented non-parametric scattered data interpolation**.

Computational morphology

Suppose a finite set of points on a plane curve is given. If the sample is large enough and well distributed, then it is an easy task for a human to perceive the shape of the curve (see Figure 12). In fact—given adequate visualization tools—it is equally easy to perceive the shape of surfaces from dense samples on them.

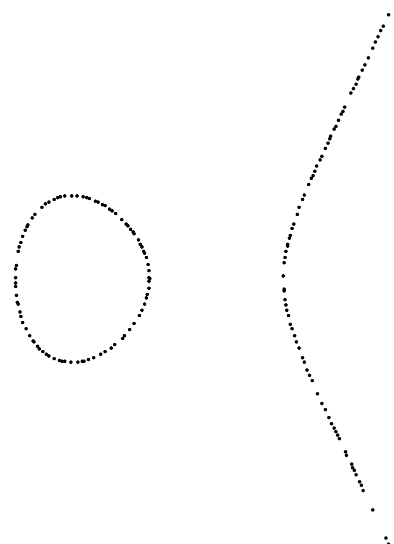

Figure 12: easy perception of the shape of a curve.

Human perception of curve shape includes not only topological aspects, such as the identification of connected components and the differentiation between open and closed pieces, but also geometrical aspects, such as qualitative measures of curvature and winding. A machine, on the other hand, has no such natural perception: the sample has no *a priori* structure that can be exploited to provide a computational description of the curve; the sample is merely a list of coordinates. A fundamental problem, which we would expect to solve using such computational descriptions, is exactly how to sort the points in an order compatible with the natural trace of the curve, as perceived by a human; this order may be used to structure the sample into a polygonal approximation of the curve (see Chapter 5).

Reproducing human perception of shapes from **dot patterns** (dense uniform samples of "fat" points on plane regions) is a classical problem in low-level computer vision, pattern recognition, and cluster analysis [Zahn (1971), Toussaint (1980)]. Much of the work in computational geometry was originally motivated by problems in these fields. Toussaint (1980) coined the term **computational morphology** for the use of computational geometry for extracting shape information. Whereas mathematical morphology deals with discrete digital images and aims to deduce shape from binary values on regular grids [Serra (1982)], computational morphology uses methods from computational geometry to deal with continuous geometric objects. Curiously, the name "computational geometry" appears in the subtitle of a classical book by Minsky and Papert (1988) about binary images and the theoretical limitations of computer vision. We use "computational geometry" in the sense of Preparata and Shamos (1985).

There are two overlapping aspects to the shape of dot patterns: **external shape**, which is the macro structure of the boundary, and **internal shape**, which is the micro structure of the interior [Kirkpatrick–Radke (1985)].

Some simple computational descriptions of external shape are: the bounding box, the bounding rectangle, the bounding circle, and the convex hull. Edelsbrunner, Kirkpatrick and Seidel (1983) defined a parametrized family of external shape descriptors, called α-hulls, which include bounding circles and convex hulls. Unlike the simple descriptors listed above, α-hulls are able to describe shapes with holes and multiple components. Moreover, discrete versions of α-hulls, called α-shapes, can be efficiently computed from Delaunay triangulations.

A possible description of internal shape is a triangulation; however, unless the sample lies in a straight line, triangulations will have co-dimension 0 and hence cannot accurately describe curves, which have high co-dimension. On the other hand, Delaunay triangulations describe both internal and external shape (see Figure 13).

Graphs for shape description

Most shape descriptors are graphs, and, like many graphs occurring in geometric problems, are actually **geometric graphs**: the vertices are points in a Euclidean space and the edges are straight-line segments connecting the points. For geometric problems, the main interest is in geometric graphs whose edges do not intersect; such geometric graphs are said to be **embedded**. However, not all geometric graphs used for shape description are embedded (see Figure 13).

The various graphs used to describe shape differ, of course, in their choice of edges. This choice depends on the notion of proximity that one intends to capture in the graph; such shape descriptors are accordingly called **proximity graphs**.* Proximity varies with the type of description sought: describing curves is different from describing plane regions of co-dimension 0.

Several criteria have been proposed for choosing a proximity edge between two points p and q in a set S:

*Boissonnat (1984) says that "a structure is needed to make proximity relationships explicit". Kirkpatrick and Radke (1985) say that describing shape needs the identification of "essential neighbours".

- p and q are **nearest neighbours**, i.e., p is at least as close to q as any other point of S, or vice-versa;
- p and q are **reciprocal nearest neighbours**, i.e., p is at least as close to q as any other point of S, and vice-versa;
- pq is an edge of a Euclidean minimal spanning tree for S.
- p and q are **relative neighbours**, i.e., $d(p,q) \leq \min\{\max(d(p,r), d(q,r)) | r \in S\}$.
- p and q are **Gabriel neighbours**, i.e., $d(p,q) \leq \min\{\sqrt{d(p,r)^2 + d(q,r)^2} | r \in S\}$.
- p and q are **Delaunay neighbours**, i.e., their Voronoi polygons share an edge.
- p and q suffer **mutual influence** i.e., the spheres centered at p and q and passing through their nearest neighbours intersect.

The graphs corresponding to these criteria are the nearest neighbour graph (NNG), the minimal spanning tree (MST), the relative neighbourhood graph (RNG), the Gabriel graph (GG), the Delaunay triangulation (DT) and the sphere of influence graph (SIG). Some of them form a discrete hierarchy:

$$NNG \subseteq MST \subseteq RNG \subseteq GG \subseteq DT.$$

This hierarchy corresponds to an increase in the completeness of the description provided: from sparse disconnected (NNG) to connected acyclic one-dimensional (MST) to maximal two-dimensional (DT). The increase in completeness follows a decrease in the co-dimension of the description (see Figure 13).

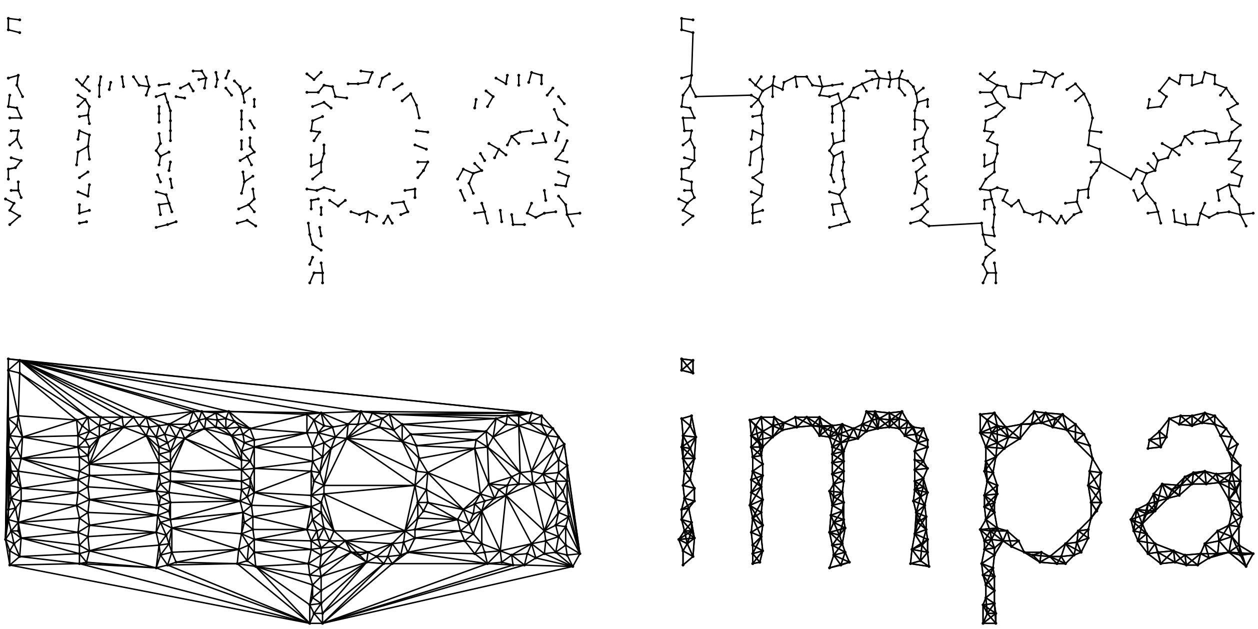

Figure 13: internal shape described by NNG, MST, DT and SIG.

Inspired by the work on α-shapes as external shape descriptions, Kirkpatrick and Radke (1985) defined a parametrized family of graphs, called β-skeletons, which they propose as descriptors of internal shape. These β-skeletons include and generalize some of the graphs above (NNG, RNG, GG and DT). However, not all β-skeletons are embedded.

The notion of proximity used in β-skeletons is a local one. Like most minimal structures, minimal spanning trees cannot be characterized locally (see the bridge theorem later in this chapter). Thus, Kirkpatrick and Radke had to exclude minimal spanning trees as descriptors, despite their widespread use, especially in cluster analysis [Gower–Ross (1969)].

If the objective is a two-dimensional description of dot patterns, then graphs such as RNG, GG, DT and SIG can be better descriptors than minimal spanning trees, which always give connected acyclic one-dimensional descriptions (see Figure 13). Nevertheless, as we shall see in Chapter 5, such descriptions are (almost) exactly what is needed for structuring samples on curves.

Cluster analysis

The identification of connected components is a basic problem in the description of dot patterns. Cluster analysis deals with this problem and motivated our choice of minimal spanning trees as shape descriptors for curves [Duda–Hart (1973), Gower–Ross (1969), Godehardt (1990)].

Informally, a **cluster** in a set of points is a subset of points having strong internal similarities and a **clustering** is a partition of the set of points into clusters [Duda–Hart (1973)].

There are two main issues in cluster analysis: measuring similarity and evaluating partitions. Similarity—or rather, dissimilarity—is measured computationally by assigning a real number to each pair of points: the greater the number, the less similar the points. In some applications, the actual values assigned are not as important as their relative order. Once a similarity measure has been chosen, partitions are evaluated by computing the total dissimilarity between their induced clusters; clustering is then realized by minimizing total dissimilarity.

Given a dissimilarity measure d, a common choice for cluster dissimilarity between two clusters X and Y is Hausdorff distance:

$$d(X, Y) = \min_{x \in X, y \in Y} d(x, y).$$

A common choice for total dissimilarity in a partition is the sum of squares of cluster dissimilarities. For these choices, the objective of clustering is to find a partition $\mathcal{P}$ that solves the following optimization problem:

$$\arg\min_{\mathcal{P}} \sum_{X, Y \in \mathcal{P}} d(X, Y)^2.$$

For general similarity measures, the minimization problem posed by clustering is NP-hard [Garey–Johnson (1979), Shamos (1976)]; all known algorithms for this kind of problems have to search an exponentially long list of candidates to find a solution. Nevertheless, if the number of clusters is fixed, then the minimization problem has polynomial solutions for a wide class* of similarity measures [Capoyleas–Rote–Woeginger (1991)]. Unfortunately, this result is not useful for our purposes because fixing the number of clusters assumes that the number of connected components is known.

For geometric problems, a natural measure of similarity is Euclidean distance: points and clusters are declared similar if their distance does not exceed a certain threshold δ. The graph induced by this relation is called the **proximity graph with threshold** δ. At the lowest thresholds, we find nearest neighbours. The graph induced by this relation is a crude descriptor of internal shape, which is too sparse to be useful (see Figure 13). Clustering methods that are based on the identification of nearest neighbours are called **single-linkage**: they identify the shortest links connecting clusters.

Threshold clustering provides a continuously parametrized but discrete family of descriptors of both internal and external shape. The main drawback of threshold clustering is that there is no automatic way for finding the value of a "natural" threshold. The need for pre-selecting a threshold can be avoided by grouping the clusters found at each threshold into a single **hierarchical clustering**, which is described by a tree called a **dendogram**. At lower levels in this tree, one finds nearest neighbours graphs; minimal spanning trees appear at higher levels, linking clusters of nearest neighbours.

Reversing this process, we can find clusters by removing atypically long edges from minimal spanning trees; such edges are called **inconsistent** by Zahn (1971), who suggests two natural heuristic criteria for deciding when an edge is too long. One criterion is global: remove edges in order of decreasing length. The other criterion is local: remove edges that are much longer than the average length of neighbour edges, e.g., at least twice as long. The problem with both approaches is one that pervades cluster analysis and was encountered above in threshold clustering: the presence of parameters that have to be tuned. With the global criterion, one has to decide when to stop removing edges, unless the number of clusters is known beforehand; we have seen that this is not a realistic assumption. With the local criterion, one has to define what is meant by "much longer". Nevertheless, inconsistent edge removal is adequate for identifying connected components of implicit objects, as we shall see in Chapter 5, where we use minimal spanning trees not only for identifying connected components, but also for modeling curves.

Minimal spanning trees

The computation of minimal spanning trees arose in network planning [Prim (1957)] and became a classical problem in graph theory. A **spanning tree** for a graph is a subtree passing through all vertices of the graph. A **minimal spanning tree** for a weighted graph is a spanning tree for which the sum of edge weights is minimal. The geometric version of a minimal spanning tree is called a **Euclidean minimal spanning tree**: the graph is a complete geometric graph and the weight of an edge is its length. The terminology for

*All monotone functions of the diameters or radii of clusters, assuming d is Euclidean distance.

geometric graphs is adopted for non-geometric weighted graphs, even though the weights are not distances: thus, we talk about "shortest" edges when we mean edges of least weight.

We said earlier that, for geometric problems, the main interest is in embedded graphs, but it is not entirely obvious that embedded spanning trees exist. Although it is easy to see that any Euclidean minimal spanning tree is embedded, a direct constructive proof of existence of embedded spanning trees is given by the following sweep algorithm, which actually constructs an embedded spanning path in the generic case:

· sort the points by height;
· recursively construct a spanning tree for the points on each level;
· select one point on each level;
· join selected points in adjacent levels by straight line segments.

Depth-first search is a fundamental exploration method for graphs, which runs in time linear in the size of the graph (the **size** of a graph is the total number of vertices and edges) [Sedgewick (1988)]. The edges visited in a depth-first search form a **spanning forest**, i.e., a spanning tree for each connected component of the graph. It follows that a graph has a spanning tree if and only if it is connected; in this case, a spanning tree can be found in linear time by depth-first search. For Euclidean problems, this time is then quadratic in the number of points. However, as it frequently happens in geometric problems, there are faster algorithms for Euclidean spanning trees. For example, the sweep algorithm described above finds an embedded spanning tree for n points in optimal time $\Theta(n \log n)$.

There are many good algorithms for finding minimal spanning trees in general graphs; this is a surprising result given that the complete graph with n vertices has n^{n-2} spanning trees [Cayley (1889)]. The performance of such algorithms varies with the complexity of the graph: some are better for sparse graphs, some are better for dense graphs. Because the Euclidean problem is phrased in terms of complete graphs, the best general algorithm would still be quadratic in the number of points. It is natural to expect that there are faster algorithms for Euclidean minimal spanning trees because the weights, being distances, cannot be arbitrary: they must obey the triangle inequality. As we shall see later, there are such algorithms for dimensions 2 and 3, but the situation for higher dimensions is not yet clear.

The fundamental property of minimal spanning trees, on which most algorithms are based, is due to Prim (1957); I call it the **bridge theorem**:

THEOREM. *Let $A \cup B$ be a partition of the set of vertices of a weighted graph G. Then any shortest edge connecting a vertex of A and a vertex of B is an edge of a minimal spanning tree for G.*

PROOF: Let T be a minimal spanning tree for G, and let e be a shortest edge connecting A and B. If e is not an edge of T, then adding e to T creates a cycle. This cycle contains an edge of T connecting A and B. Remove this edge: the result is still a spanning tree for G, having total edge length not greater than that of T. This tree is, therefore, a minimal spanning tree containing the bridge e between A and B. ∎

A direct implementation of this theorem provides the following $\Theta(n^3)$ algorithm for constructing a minimal spanning tree T for a graph G with n vertices (the complexity of the algorithm is actually $\Theta(nE)$, where E is the number of edges in the graph, but $E = \Omega(n^2)$):

$v \leftarrow$ any vertex of G
$T \leftarrow \{v\}$
$N \leftarrow G \setminus \{v\}$
while $N \neq \phi$
    $e \leftarrow$ an edge bridging N and T
    $v \leftarrow$ the extreme of e in N
    $T \leftarrow T \cup \{e\}$
    $N \leftarrow N \setminus \{v\}$

This algorithm is correct because it only selects edges which are guaranteed by the bridge theorem to be in a spanning tree, stopping after the right number of edges have been selected, and the tree T has spanned the graph G.

It is possible to reduce the complexity of this algorithm by finding the bridge e more efficiently in the main loop; the complexity is cubic only when a brute-force quadratic search is used. Prim's algorithm dynamically keeps track of all edges that could bridge N and T, replacing this quadratic search by a linear search, thus reducing the total running time to $O(n^2)$:

$v \leftarrow$ any vertex of G
$T \leftarrow \{v\}$
$N \leftarrow S \setminus \{v\}$
for each $n \in N$
    mate$(n) \leftarrow v$ *the* **mate** *of $n \in N$ is a vertex of T closest to n*
while $N \neq \phi$
    $v \leftarrow$ a vertex of N closest to T *v minimizes $d(v, mate(v))$*
    $w \leftarrow$ mate(v)
    $T \leftarrow T \cup \{vw\}$
    $N \leftarrow N \setminus \{v\}$
    for each $n \in N$ *update mates*
        if $d(n, v) < d(n, \text{mate}(n))$ then mate$(n) \leftarrow v$

Using algebraic decision trees as computational model, a lower bound for the complexity of finding minimal spanning trees in general weighted graphs is $\Omega(E)$ because, given that edge weights are unrestricted and that, by the bridge theorem, every shortest edge must be in a minimal spanning tree, any faster algorithm could be used to find the minimum of a set of real numbers in sublinear time, which is not possible. There are algorithms that actually take time $O(E)$, provided $E = \Omega(n^{1+\varepsilon})$ [Cheriton–Tarjan (1976)].

A lower bound for the complexity of finding Euclidean minimal spanning trees for a set of n points is $\Omega(n \log n)$ because any algorithm for computing Euclidean minimal spanning trees can be used for sorting. Indeed, to sort a set of real numbers $x_1, \ldots, x_n$, construct a Euclidean minimal spanning tree for the points $(x_1, 0), \ldots, (x_n, 0)$ in $\mathbf{R}^2$. Because these points are contained in a straight line, this tree is actually a **path**: every vertex has degree at most 2. The neighbours of $(x_i, 0)$ in this tree correspond to the predecessor and successor of x_i in $\mathbf{R}$. Thus, starting at the leftmost point, and following neighbours to the right, each point is visited in order. Because this is a linear step, finding Euclidean minimal spanning trees is at least as hard as sorting, which has complexity $\Omega(n \log n)$.

Any algorithm that computes a Euclidean minimal spanning tree by exploring complete graphs must take time $\Omega(n^2)$. As expected, there are faster algorithms for Euclidean minimal spanning trees; such algorithms exploit the geometric nature of the problem to reduce the number of edges that need to be considered. For example, in dimension 2, Euclidean minimal spanning trees are subgraphs of Delaunay triangulations, and these have size linear instead of quadratic in the number of points. Moreover, a Euclidean minimal spanning tree can be extracted in linear time from a Delaunay triangulation, and Delaunay triangulations can be found in optimal time $\Theta(n \log n)$. This provides an algorithm for finding Euclidean minimal spanning trees in the plane in optimal time $\Theta(n \log n)$ [Preparata–Shamos (1985)]; in fact, this is the only known optimal algorithm in dimension 2.

No optimal algorithms are known in dimension 3. It is not even known whether time $O(n \log n)$ can be attained; the best time to date is $O((n \log n)^{4/3})$ [Agarwal–Edelsbrunner–Schwarzkopf–Welzl (1991)]. Although it is true that Euclidean minimal spanning trees are subgraphs of Delaunay triangulations in all dimensions, the size of higher dimensional Delaunay triangulations is no longer linear in the number of points: a Delaunay triangulation of n points in $\mathbf{R}^d$ can have $\Omega(n^{\lceil d/2 \rceil})$ edges, being therefore quadratic in dimension 3 [Klee (1980)]. However, all hope is not lost because Euclidean minimal spanning trees are subgraphs of relative neighbourhood graphs in any dimension, and these always have linear size. Moreover, relative neighbourhood graphs can be computed in optimal time $\Theta(n \log n)$ in dimension 2 and in time $O(n^2)$ in higher dimensions. It has been suggested that better algorithms for relative neighbourhood graphs will lead to better—perhaps optimal—algorithms for Euclidean minimal spanning trees in arbitrary dimension [Supowit (1983)].

5

Modeling

In this chapter, we combine the techniques described in Chapters 3 and 4 into a modeling method for curves. Minimal spanning trees will be the basic structuring tool. We describe how the information provided by this tool can be used to structure samples on increasingly complex curves, starting with arcs and concluding with a method for modeling implicit curves. The main result in this chapter is that minimal spanning trees are able to correctly structure sufficiently dense samples on arcs. In the absence of a global structuring method, two ideas for structuring samples on surfaces are presented.

Arcs

We structure samples of points on a curve by means of graph models. A **graph model** for a curve is a graph having the same topology as the curve. However, the graph models that we shall describe are **geometric**, in the sense that the vertices correspond to actual points on the curve and the collection of edges provides a polygonal approximation to the curve. For such graph models, a homeomorphism from the model to the curve is given by a linear homotopy. Our geometric graph models can be easily simplified into a minimal graph model by contracting edges incident on vertices of degree 2. Minimal graph models have been used for the detailed study of algebraic curves [Arnon (1983)].

Graph models could be defined for surfaces too, but they would have to include faces, in a fashion similar to boundary representations of solid models. As we shall see, finding combinatorial models with good geometric properties is a major problem in the computational morphology of surfaces.

Despite the success of other graphs in describing plane dot patterns of co-dimension 0, our basic graph model will be the Euclidean minimal spanning tree, because it proved to be well suited to describing one-dimensional objects, while being well understood and easily computed. More precisely, we shall see that minimal spanning trees provide a way to construct graph models of curves.

A tree cannot model curves with multiple or closed components; a tree can only model the simplest curves, i.e., those diffeomorphic to the closed unit interval (requiring diffeomorphisms instead of homeomorphisms avoids the unpleasantness of wild embeddings). Such curves are called **arcs**, and tree models for arcs must be paths. Accordingly, we must investigate when a path is a minimal spanning tree.

A **chord** in a path is an edge joining non-consecutive vertices. A **short chord** is a chord that is shorter than some edge of the subpath connecting its extremes; such an edge is called a **witness** for the short chord (see Figure 14).

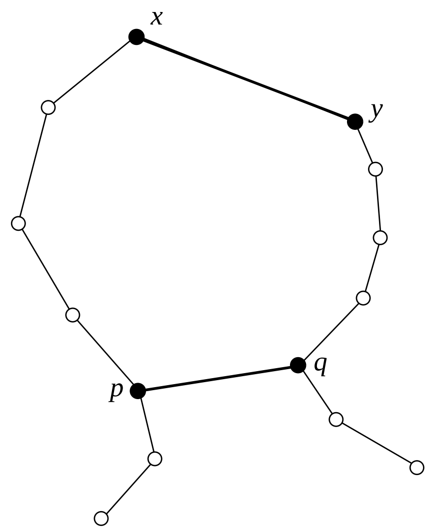

Figure 14: the edge xy is a witness for the short chord pq.

The absence of short chords characterize the paths that are minimal spanning trees, as the next theorem shows.

THEOREM. *A spanning path is a minimal spanning tree if and only if it has no short chords.*

PROOF: If a path has a short chord, then it cannot be a minimal spanning tree because a shorter spanning tree is obtained by replacing any witness by the chord.

Conversely, we shall show that if a path is not a minimal spanning tree, then it has short chord.

The proof is by induction on the number of vertices in graph spanned by the path. For graphs with 1 or 2 vertices, there is nothing to prove, because every path is minimal.

Otherwise, let P be a non-minimal spanning path for a graph G. Then, there is an edge $e \in P$ and an edge $f \in G$ such that the graph $T = P - e + f$, obtained by replacing e by f in P, is a tree shorter than P. Note that this implies that f is shorter than e.

Let u be an extreme of P, and let $P' = P - u$. We shall prove that either P has a short chord or P' has a short chord. Of course, any short chord in P' is also a short chord in P.

If u is an extreme of f, then f is a chord of P. Moreover, since T is a tree, the edge e must be in the subpath of P joining u and w, where w is the other extreme of f. Since f is shorter than e, this means that the edge f is a short chord in P, witnessed by e.

If u is not an extreme of f, then u is not an extreme of e: otherwise, $T = P-e+f = P'+f$ would not be tree because f is a chord of P'. Hence, the edge e is an edge of P', and $P - e + f = (P' - e + f) + uv$, where v is the (only) neighbour of u in P. Now, T is a tree shorter than P if and only if $P' - e + f$ is a tree shorter than P'. Thus, P' is not a minimal tree, and hence, by induction, P' has a short chord. This short chord is also a short chord in P. ∎

This theorem implies that if a sample on an arc is dense enough so that the induced polygonal path through the sample points has no short chords, then we can recover this path by computing a minimal spanning tree for the sample. It is striking that a global algorithm, which has no knowledge of the curve being sampled, is able to recover the local information needed to visit the sample points in an order compatible with the trace of the curve, thus constructing a correct polygonal approximation for a curve it does not know. This is the kind of result that we would like to classify as computational morphology for non-parametric scattered data interpolation. As we shall see, there is no comparable result for surfaces, yet.

A **tubular neighbourhood** for a curve C is a set T containing C such that every point of T belongs to exactly one line normal to C (see Figure 15). Tubular neighbourhoods always exist for regular curves [Milnor (1965)]. If the curve is given implicitly by a function h, then a tubular neighbourhood is given by $h^{-1}([-\varepsilon, \varepsilon])$, for ε sufficiently small, because the set of regular values is open, as mentioned in Chapter 2.

Tubular neighbourhoods are closely related to the notion of sample density that we need: we shall say that a sample of a curve C is **dense** if there is a real number $\varepsilon > 0$ such that the closed disks of radius ε centered at the sample points form a tubular neighbourhood of C. A disk centered on the curve and contained in a tubular neighbourhood of the curve will be called a **tubular disk** (see Figure 15).

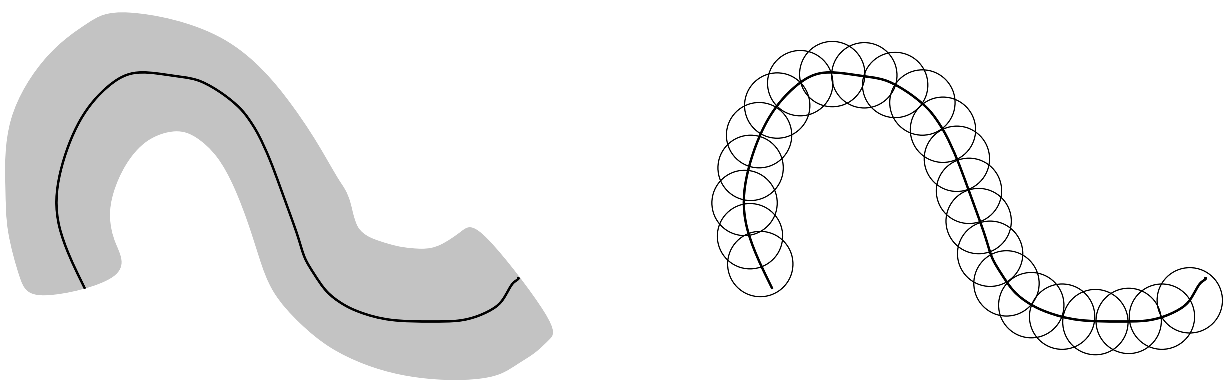

Figure 15: tubular neighbourhood and tubular disks.

We are now ready to prove our main structuring theorem.

THEOREM. *Let S be a dense sample on an arc C containing the extremes of C. Then the Euclidean minimal spanning tree of S coincides with the polygonal path induced by S on C, and is therefore a geometric graph model for C.*

PROOF: By the previous theorem, we have to show that the polygonal path P induced by S on C has no short cords. Suppose that pq is a chord shorter than an edge xy of the subpath pq. Let $\varepsilon > 0$ be the radius of the tubular disks centered on S and covering C. Then $d(x, y) < \varepsilon$ because y is a nearest neighbour of x on C. Hence, if $d(p, q) < d(x, y)$, then q is inside the tubular disk D centered at p. We shall show that, in this case, the arc

pq is actually completely inside the disk D' with diameter pq; then no chord can be longer than pq. This contradicts $d(p,q) < d(x,y)$.

First, we show that the arc pq is completely inside D; in fact, we show that $C \cap D$ is connected. Indeed, let K be a connected component of $C \cap D$. If K is not the component of p, then there is a point $k \in K$ closest to p. The line pk is normal to C at k, but it meets the normal to C at p; this cannot happen inside the tubular disk D (see Figure 16).

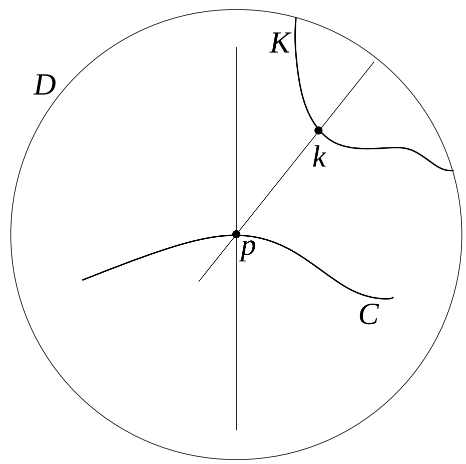

Figure 16: $C \cap D$ is connected.

To show that the arc pq is completely inside the disk D', place q vertically above p, for the purpose of exposition. Next, note that the arc pq remains inside the upper half of D: otherwise, the normal at the lowest point a would meet the normal at the point b farthest from the segment pq (see Figure 17).

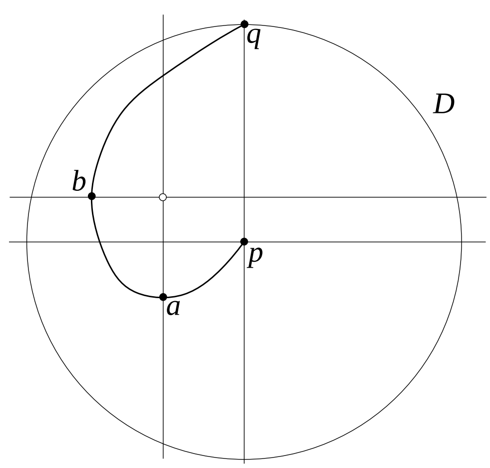

Figure 17: arc remains inside upper semi-circle.

Next, take the points r and s where the normal at b meets the circle, and consider the perpendiculars to pr and qr: they meet at u (see Figure 18). The angle puq is obtuse because, in the quadrilateral $prqu$, the angle prq is acute and the angles upr and uqr are equal to $\pi/2$.

An analogous construction with the point s instead of r gives a point v such that the angle pvq is obtuse. The point v is actually the reflection of u on pq (see Figure 18). The points u and v are inside D' because the angles puq and pvq are obtuse. For this reason, we shall show that the arc pq is completely inside D', by showing that the arc pq is actually completely inside the quadrilateral $puqv$.

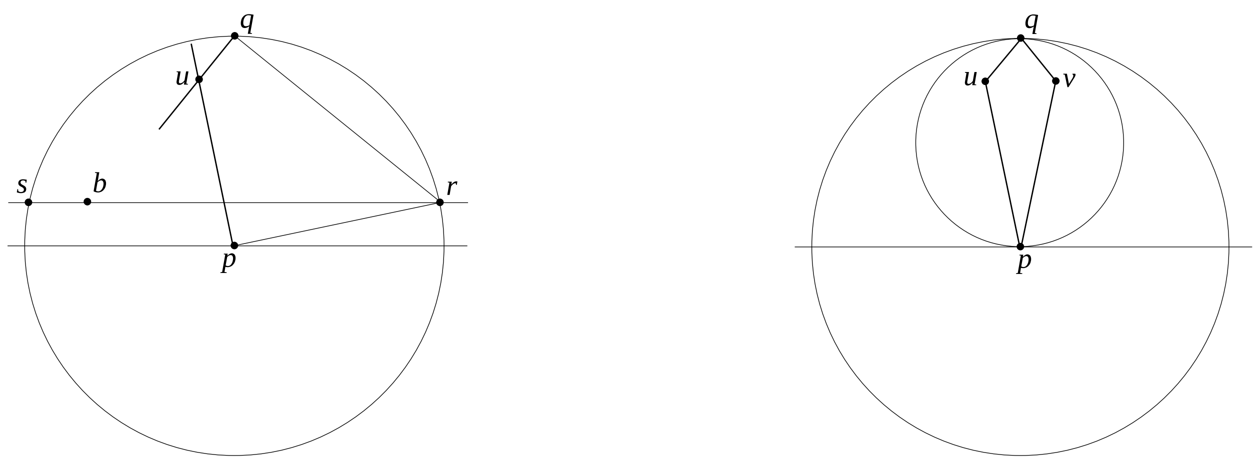

Figure 18: arc remains inside smaller circle.

Indeed, if c is a point on the arc pq but outside the quadrilateral $puqv$, then by Rolle's theorem, there exists a point d on the arc whose tangent is parallel to either the chord pc or the chord qc. But then normal at d meets the normal at b inside D, a contradiction. (see Figure 19). ■

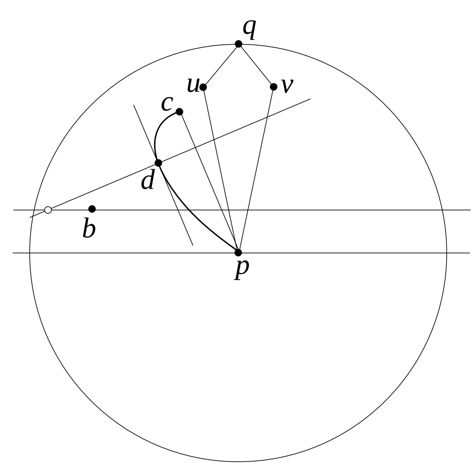

Figure 19: arc remains inside quadrilateral.

Note that the use of Rolle's theorem, and the existence of tubular neighbourhoods, which are essential for the proof of the theorem, strongly rely on the differentiability of the curve.

In fact, the theorem is false for C^0 curves: take a polygonal line abc such that the sides ab and bc have the same length and make an acute angle. Take a sample on ab containing b and replicate it on bc. Then, no matter how dense the sample is, the point b will always be a leaf of any minimal spanning tree (see Figure 20).

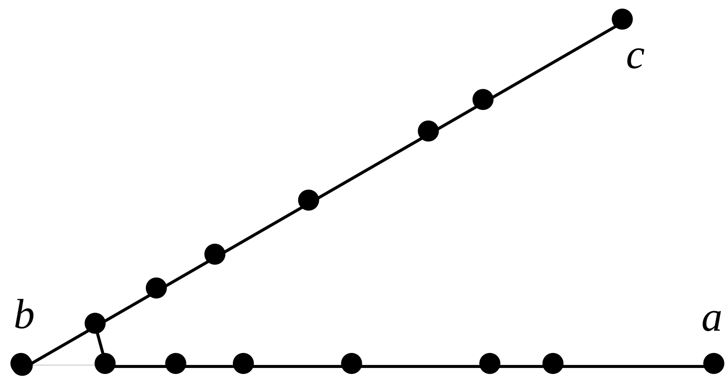

Figure 20: differentiability is essential for the theorem.

Loops

After arcs, the next simplest kind of curves are **loops**: they are diffeomorphic to the unit circle. A correct structure for a dense sample on a loop is obtained by computing a minimal spanning tree and then adding the edge joining the two vertices of degree 1 (called **leaves**). As we have seen above, a minimal spanning tree for the sample will model an arc of the loop, not the complete loop (see Figure 21). This arc is defined by removing the longest segment of the loop determined by the sample.

Thus, assuming that the sample is on a loop, a correct geometric graph model is given by the addition of a single edge to a minimal spanning tree on the sample.

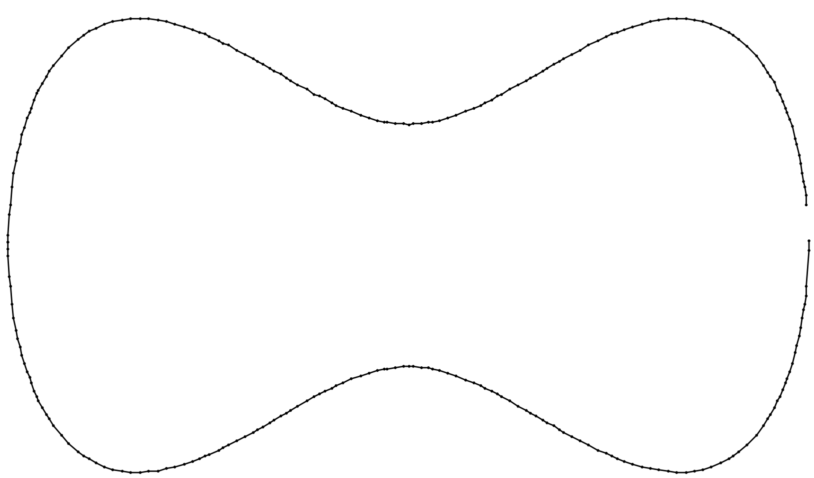

Figure 21: a minimal spanning tree on a loop.

For some time during this research, it seemed that relative neighbourhood graphs would be a good model for loops because, unlike minimal spanning trees, they seem to handle closed components directly, without postprocessing. However, it turned out that relative neighbourhood graphs are too sensitive to global data: for example, consider the curve given implicitly by $y^2 = a^2 - x^4 + x^2$, with a sufficiently small (say, $|a| < 0.7$), and take a sample on this curve containing the points $p = (0, a)$ and $q = (0, -a)$. Then, no matter how dense the sample is, the relative neighbourhood graph of this sample contains the chord pq (see Figure 22).

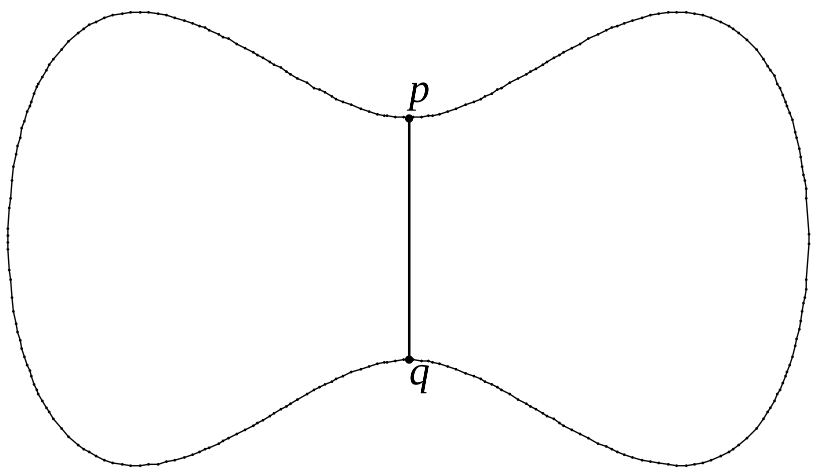

Figure 22: wrong chord in relative neighbourhood graphs.

Regular Curves

The connected components of regular curves are either arcs or loops. Accordingly, to structure dense samples on regular curves, it is sufficient to identify its connected components: each component can then be structured separately. As we have seen in Chapter 4 and up to now in this chapter, minimal spanning trees are useful for both phases.

Connected components are identified by finding and removing inconsistent edges from minimal spanning trees. The assumption that the sample lies on a regular curve allows the selection of inconsistent edges by performing simple topological tests instead of using the heuristic criteria suggested by Zahn, which we described in Chapter 4. More precisely, since a graph model for a regular curve cannot have vertices of degree greater than 2 (called **branching points**), the edges joining two such vertices in a minimal spanning tree are exactly the edges to be removed: they bridge different connected components (see Figure 23). There is no need for an inconsistency analysis *à la* Zahn.

Once the connected components have been identified, we need to identify the closed components. After removing the bridges from a minimal spanning tree, we are left with a set of paths. To identify which paths should be closed, we must decide to which paths we should add an edge joining their leaves; this edge will be called the **closing edge**. Unlike the previous cases, however, this step must be done heuristically.

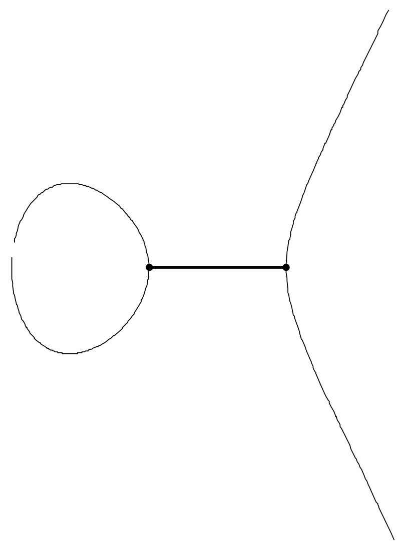

Figure 23: bridge in a minimal spanning tree on a regular curve.

We suggest the following heuristic criteria for closing a path:

- the angles defined by the closing edge at the leaves are both close to π;
- the closing edge is not much greater than others in the same component, being consistent in the sense of Zahn;
- the closing edge is an edge of the relative neighbourhood graph.

For the last criterion—which was suggested by experiments, as reported in the previous section—we can take either the whole sample or the sample corresponding to the connected component being considered. The first choice is more consistent with the philosophy of local structuring through global structuring.

Implicit Curves

General implicit curves can be very complex: they can be singular, and they can have multiple connected components, some of which may be closed.

We shall keep minimal spanning trees as our basic structuring tool. The additional knowledge provided by the simplifying assumptions made in the previous sections will be provided by the values of the function h that defines the curve implicitly.

The structuring method is essentially the same as the one for regular curves: identify connected components by identifying and removing bridges, and handle each component separately. However, we cannot immediately classify as bridges all edges joining branching points: such vertices may correspond to singular points of the curve. Nevertheless, all edges incident on branching points are considered potential bridges. A certificate is provided by sampling the function h on such edges: by definition, the value of h on true bridges is non-zero.

After removing the bridges, we close the remaining paths in exactly the same way as described for regular curves, except that we need to consider all pairs of leaves: singular components will have several leaves, some of which should be joined, some of which should not. The function h also provides a new criterion for the selection of closing edges: they should be approximately orthogonal to ∇h. This same criterion can be used for the identification of bridges: in this case, bridges will not only be far from orthogonal to ∇h, but will actually be approximately parallel to ∇h.

Practical modeling of implicit curves

We are now ready to combine the physically-based sampling method described in Chapter 3 and the structuring method described in the preceding sections into a modeling method for implicit curves. Given a function h, the following algorithm computes a good piecewise linear approximation for the implicit curve $V = h^{-1}(0)$:

· select initial sample randomly;
· select sampling tolerance $\varepsilon > 0$;
· apply physically-based sampling method;
· stop simulation when $|h| < \varepsilon$ on the sample;
· structure final sample by analyzing an Euclidean minimal spanning tree;
· perform depth-first search on the resulting graph to output polygonal lines.

The main practical problems with this method are the selection of the initial sample and the selection of the sampling tolerance ε. If the initial sample is too far away from $h^{-1}(0)$, then the sample after equilibrium might not cover all of V. We may therefore view the sampling step as a preliminary search for V. After V is located, we can then sample it more thoroughly by selecting another set of initial conditions near V, e.g., by sampling a bounding box twice as large as the bounding box of the preliminary sample. A few iterations of this process are usually enough to locate and sample all components of V without bias.

The next problem is intrinsic to the numerical solution of equations: tolerance. It does not make sense to try to compute exact solutions of $h(x) = 0$; the best we can do is to compute solutions of $|h(x)| < \varepsilon$. This is the equilibrium criterion selected for stopping the simulation. Because the resulting sampling is in $h^{-1}([-\varepsilon, \varepsilon])$, and not on $h^{-1}(0)$, it is possible that a minimal spanning tree has dangling edges, especially if ε is large. These edges link points where h is large to the curve. Such points can be classified as outliers, and can be identified and removed because outliers do not belong to the longest path in their connected component [Duda–Hart (1973)]. This same technique can be used to remove noise present in samples obtained with other sampling methods.

To get a good final sample with our physically-based method, we usually need many points, and this means thousands instead of hundreds, especially for higher dimensional objects. Whereas the simulation of uncoupled physical systems with thousands of particles is not too expensive in a workstation, geometric computations, such as minimal spanning trees, are usually tightly coupled. Moreover, the samples provided by the physically-based method usually contain many almost coincident points; such concentrations occur at singularities and points of high curvature, as mentioned in Chapter 3. We are thus faced with the problem of extracting a representative sample from the equilibrium sample.

One good way of doing this extraction is to select a real number $\delta > 0$ and collapse all δ-cliques: a **δ-clique** is a set of points such that the distance between any two points in the set is at most δ; δ-cliques are the cliques in the proximity graph with threshold δ defined in Chapter 4. A brute-force algorithm computes δ-cliques in cubic time. A good approximation is given by the following bucketing algorithm, which runs in linear time: divide the bounding box of the sample into square buckets of size δ, and collapse the points in each bucket. This collapsing can be done either by electing representative or by using the barycenter of the population of each bucket.

Note that most of the practical issues discussed in this section are not particular to curves, and can be used when modeling surfaces. Note, moreover, that the method is not restricted to plane curves: it works for curves of all co-dimensions. In particular, the method provides a new way to compute the intersection of surfaces. More precisely, as mentioned in Chapter 2, the intersection of two surfaces given implicitly by $f, g\colon \mathbf{R}^3 \to \mathbf{R}$ is the curve given implicitly by $h = f^2 + g^2$ (see Figure 24).

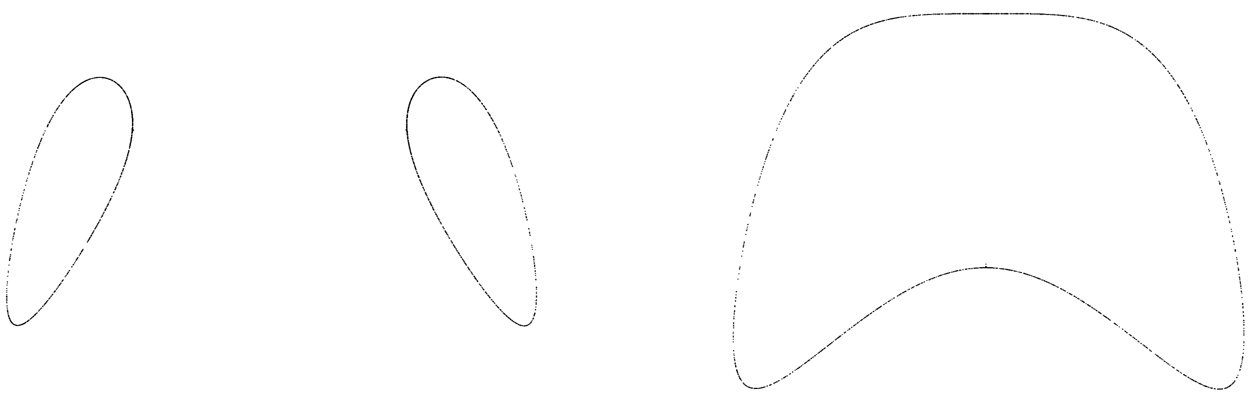

Figure 24: two instances of the intersection of a cylinder and a sphere.

Surfaces

The approximation theorem for curves given above cannot be directly extended to surfaces because trees are not good models for surfaces. The main problem in this extension is the definition of an analogous model for surfaces. One would immediately think of polyhedra of minimum area [O'Rourke (1981)], but this does not seem to make sense unless we restrict the genus of the models, something we do not want to do because an adequate morphological tool should be able to deduce the genus from the data [Hoppe–DeRose–Duchamp–McDonald–Stuetzle (1991)]. Moreover, it is not likely that efficient minimization algorithms exist. In the absence of a good class of models for surfaces, we suggest two methods which we believe to be practical, even though they lack the good global qualities of the method for curves.

The first method relies on the method we have just described for curves: it samples an implicit surface on several parallel plane slices (see Figure 25). This reduces the sampling of the implicit surface to the sampling of several implicit curves; we have shown how to do not only sampling, but also structuring for such curves. To avoid the problems of ray-casting sampling described in Chapter 3, it is useful to sample the surface with the physically-based method before sampling on each slice. This preliminary sampling gives valuable information for the selection of the slices to be sampled. Global structuring of these slices can be done by a general purpose integration algorithm. We suggest the method proposed by Giertsen, Halvorsen and Flood (1990), because it is not difficult to provide the connectivity information between consecutive slices that is needed by their algorithm. In fact, minimal spanning trees are useful here too: given dense samples on two consecutive slices, the corresponding pieces on each slice are joined by edges of a minimal spanning tree for the sample (see Figure 25).

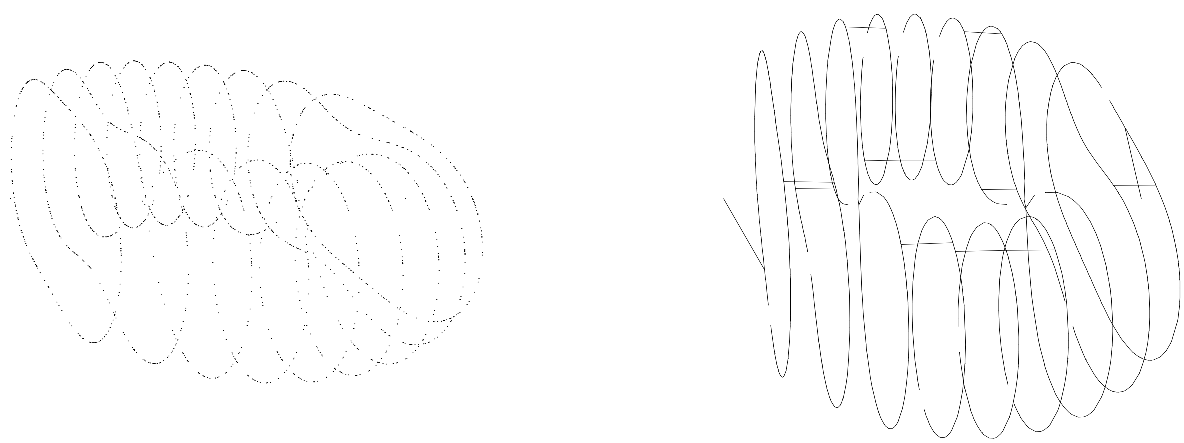

Figure 25: slice sampling and a minimal spanning tree.

The second method samples the surface using the physically-based method described in Chapter 3. The resulting sample, as explained before, is not evenly distributed across the surface. Whereas this was not a major problem for curves, it makes it harder to deduce the two-dimensional connectivity relations between the sample points that are needed to build a polygonal approximation. In order to make this deduction easier, the sample is subjected to a physically-based relaxation process similar to the one used by Turk (1991) in a different context: particles repel each other, with an intensity that rapidly decreases as the distance between the particles increases [Szeliski–Tonnesen (1991)]. Moreover, the movement of each particle is constrained to stay close to the surface by projecting repulsion forces onto the tangent plane. The result of this relaxation process is a more uniform sample of the surface. The desired polygonal approximation can be then obtained by computing the Delaunay triangulation associated to the points and choosing the triangles that approximate well the tangent planes at each of its vertices.

6

Conclusion

"No, I give it up", Alice replied. "What's the answer?"
"I haven't the slightest idea," said the Hatter.
Lewis Carroll, *Alice in Wonderland.*

We have separated the computational study of implicit objects into two problems: sampling and structuring. By concentrating on each problems separately, we have cast structuring as a problem in non-parametric scattered data interpolation, which we proposed to solve using computational morphology. The two phases were later merged into a proposal for modeling curves.

We have described a new, physically-based method for sampling implicit objects. This method is more robust than classical continuation methods because:

- arbitrary initial conditions are adequate;
- single-step numerical integration methods without correction suffice;
- the equations of motion can be solved in parallel;
- the final distribution of points has geometric meaning;
- there is no need for predefining a region of interest.

Although a sample of points is not a complete geometric model, dense samples have strong perceptual meaning (see Figure 1), especially if adequate visualization tools are available. It would be easier to use implicit objects in geometric modeling if samples on such objects could be computed fast. The method we have presented is an example of such a tool.

Much like splines, which are used not only for approximation, but also for free-form modeling, our sampling method provides not only an approximation tool, but also a dynamic modeling tool: by varying implicit equations, or by adding local potential fields, it is possible to dynamically control the shape of the model. After a satisfactory shape is found, a complete geometric model can then be constructed by using structuring methods.

We have shown that minimal spanning trees are good morphological tools for arcs, in the sense that a correct polygonal approximation to an arc can be obtained by computing a Euclidean minimal spanning tree on a sufficiently dense sample of points on this arc. This result enabled us to propose a complete modeling method for implicit curves. In the absence of an adequate computational morphology for higher-dimensional objects, we have suggested two methods for implicit surfaces:

- sampling and structuring implicit surfaces at several slices, using our results on implicit curves; the knowledge of the topology of each slice is very useful for global structuring.
- relaxation of samples so that structuring is easily done; the resulting uniform samples are also useful in modeling, because the polygonal approximations thus obtained are quasi-regular.

Future work

"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to," said the Cat.
Lewis Carroll, *Alice in Wonderland.*

The answers given for the problems discussed in this thesis actually raised more questions. We suggest the following problems as a natural continuation of this work:

- further study of the dynamical systems used in physically-based sampling, so that final distribution can be controlled, and better initial step sizes for integration can be chosen.
- identification of broad subclasses in the proposed taxonomy of structuring problems, and development of general methods for such classes.
- extension of the approximation theorems of Chapter 5 to singular curves.
- identification of which β-skeletons satisfy the approximation theorems for curves, hoping that some of them are able to find closed components automatically without adding extraneous edges.
- development of an adequate computational morphology for samples on objects of co-dimension 1, other than plane curves, especially two-dimensional surfaces in three-dimensional space.
- integration of the physically-based sampling method into a modeling system based on particles.

References

E. L. Allgower and K. Georg, Simplicial and continuation methods for approximating fixed points and solutions to systems of equations, *SIAM Review* **22** (1980) 28–85.

E. L. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*, Springer-Verlag, 1990.

E. L. Allgower and S. Gnutzmann, Simplicial pivoting algorithms for mesh generation of implicitly defined surfaces, *Computer Aided Geometric Design* **8** (1991) 305–325.

E. L. Allgower and P. H. Schmidt, An algorithm for piecewise-linear approximation of an implicitly defined manifold, *SIAM Journal on Numerical Analysis* **22** (1985) 322–346.

P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf and E. Welzl, Euclidean minimum spanning trees and bicromatic closest pairs, *Proceedings of the Sixth ACM Symposium on Computation Geometry*, 1991, 203–210.

D. S. Arnon, Topologically reliable display of algebraic curves, *Computer Graphics* **17** (1983) 219–227 (Proceedings of SIGGRAPH '83).

A. H. Barr and R. Barzel, A modeling system based on dynamic constraints, *Computer Graphics* **22** (1988) 179–188 (Proceedings of SIGGRAPH '88).

J. Bloomenthal, Polygonization of implicit surfaces, *Computer Aided Geometric Design* **5** (1988) 341–355.

J.-D. Boissonnat, Geometric structures for three-dimensional shape representation, *ACM Transactions on Graphics* **3** (1984) 266–286.

J. L. Brown, Vertex based data dependent triangulations, *Computer Aided Geometric Design* **8** (1991) 239–251.

V. Capoyleas, G. Rote and G. Woeginger, Geometric clustering, *Journal of Algorithms* **12** (1991) 341–356.

A. Cayley, A theorem on trees, *The Quartely Journal of Pure and Applied Mathematics* **23** (1889) 376–378.

D. Cheriton and R. E. Tarjan, Finding minimum spanning trees, *Communications of the ACM* **5** (1976) 724–742.

R. C. Dubes and A. K. Jain, *Clustering methodologies in exploratory data analysis*, *Advances in Computers* **19** (1980) 113–228.

R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, 1973.

N. Dyn, D. Levin and S. Rippa, Data dependent triangulations for piecewise linear interpolation, *IMA Journal of Numerical Analysis* **10** (1990) 137–154.

H. Edelsbrunner, D. G. Kirkpatrick and R. Seidel, On the shape of a set of points in the plane, *IEEE Transactions on Information Theory* **29** (1983) 551–559.

G. E. Farin (ed.), *Geometric Modeling: Algorithms and New Trends*, SIAM, 1987.

T. A. Foley, D. A. Lane, G. M. Nielson, R. Franke and H. Hagen, Interpolation of scattered data on closed surfaces, *Computer Aided Geometric Design* **7** (1990) 303–312.

G. E. Forsythe, M. A. Malcolm and C. B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, 1977.

R. Franke, Scattered data interpolation: tests of some methods, *Mathematics of Computation* **38** (1982) 181–200.

M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

C. Giertsen, A. Halvorsen and P. R. Flood, Graph-directed modelling from serial sections, *The Visual Computer* **6** (1990) 284–290.

A. Glassner (ed.), *An Introduction to Ray Tracing*, Academic Press, 1989.

E. Godehardt, *Graphs as Structural Models*, Vieweg, 1990.

J. de M. Gomes and G. Tavares, *Concordance operations for implicitly defined manifolds*, unpublished manuscript presented at the First SIAM Conference on Geometric Design, 1989.

J. de M. Gomes and G. Tavares, *Simplicial Methods in Computer Graphics*, course notes in portuguese, XVII Colóquio Brasileiro de Matemática, IMPA, 1989.

D. Gossard, *Product models with built-in physics*, unpublished manuscript presented at the Second SIAM Conference on Geometric Design, 1991.

J. C. Gower and G. J. S. Ross, Minimum spanning trees and single linkage cluster analysis, *Applied Statistics* **18** (1969) 54–64.

D. Greenspan, *Discrete Models*, Addison-Wesley, 1973.

M. Haiman, A simple and relatively efficient triangulation of the n-cube, *Discrete and Computational Geometry* **6** (1991) 287–289.

M. Hall and J. Warren, Adaptive polygonization of implicitly defined surfaces, *IEEE Computer Graphics & Applications* **10** (1990) 33–42.

B. von Herzen and A. H. Barr, Accurate triangulations of deformed intersecting surfaces, *Computer Graphics* **21** (1987) 103–110 (Proceedings of SIGGRAPH '87).

M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*, Academic Press, 1974.

C. M. Hoffmann, *Solid and Geometric Modeling*, Morgan Kaufmann, 1989.

C. M. Hoffmann and J. Hopcroft, Automatic surface generation in computer-aided design, *The Visual Computer* **1** (1985) 92–100.

H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, *Surface Reconstruction from Unorganized Points*, Technical Report 91-12-03, Department of Computer Science and Engineering, University of Washington, 1991.

S. Incerti, V. Parisi and F. Zirilli, A new method for solving nonlinear simultaneous equations, *SIAM Journal on Numerical Analysis* **16** (1979) 779–789.

A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

D. G. Kirkpatrick and J. D. Radke, A framework for computational morphology, in: [Toussaint (1985)] 217–248.

V. Klee, On the complexity of d-dimensional Voronoi diagrams, *Archiv der Mathematik* **34** (1980) 75–80.

J. D. Lambert, *Computational Methods in Ordinary Differential Equations*, Wiley, 1973.

J. M. Lane, L. C. Carpenter, T. Whitted and J. F. Blinn, Scan line methods for displaying parametrically defined surfaces, *Communications of the ACM* **23** (1980) 23–34.

C. L. Lawson, C^1 surface interpolation for scattered data on a sphere, *Rocky Mountain Journal of Mathematics* **14** (1984) 177–202.

E. Lima, *Curso de Análise*, Projeto Euclides, IMPA, 1981 (in portuguese).

W. Loreson and H. Cline, Marching cubes: a high resolution 3d surface construction algorithm, *Computer Graphics* **21** (1987) 163–169 (Proceedings of SIGGRAPH '87).

J. Milnor, *Topology from the Differentiable Viewpoint*, The University Press of Virginia, 1965.

M. L. Minsky and S. A. Papert, *Perceptrons: An introduction to computational geometry*, expanded edition, MIT Press, 1988.

S. M. Omohundro, *The Delaunay triangulation and function learning*, ICSI Technical Report TR-90-001, 1990.

J. O'Rourke, Polyhedra of minimal area as 3d objects models, *Proceedings of the Seventh IJCAI*, August 1981, 664–666.

L. Perko, *Differential Equations and Dynamical Systems*, Springer Verlag, 1991.

M. J. Pratt, Smooth parametric surface approximations to discrete data, *Computer Aided Geometric Design* **2** (1985) 165–171.

F. Preparata and M. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

R. C. Prim, Shortest connection matrix network and some generalizations, *Bell Systems Technical Journal* **36** (1957) 1389–1401.

W. T. Reeves, Particle systems—a technique for modeling a class of fuzzy objects, *Computer Graphics* **17** (1983) 359–376 (Proceedings of SIGGRAPH '83).

A. A. G. Requicha, Representations for rigid solids: theory, methods and systems, *ACM Computing Surveys* **12** (1980) 437–464.

A. Ricci, A constructive geometry for computer graphics, *The Computer Journal* **16** (1973) 157–160.

S. Rippa, Minimal roughness property of the Delaunay triangulation, *Computer Aided Geometric Design* **7** (1990) 489–497.

A. P. Rockwood and J. C. Owen, Blending surfaces in solid modeling, in: [Farin (1987)] 367–383.

L. L. Schumaker, Fitting surfaces to scattered data, in: G. G. Lorentz, C. K. Lui and L. L. Schumaker (ed.), *Approximation Theory II*, Academic Press, 1976, 203–268.

D. H. Schwartzmann and J. J. Vidal, An algorithm for determining the topological dimensionality of point clusters, *IEEE Transactions on Computers* **C-24** (1975) 1175–1182.

T. W. Sederberg, Algebraic geometry for surface and solid modeling, in: [Farin (1987)] 29–42.

T. W. Sederberg and A. K. Zundel, Scan line display of algebraic surfaces, *Computer Graphics* **23** (1989) 147–156 (Proceedings of SIGGRAPH '89).

R. Sedgewick, *Algorithms*, Addison-Wesley, 1988.

J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, 1982.

M. I. Shamos, Geometry and statistics: problems at the interface, in: J. F. Traub (ed.), *Algorithms and Complexity: New Directions and Recent Results*, Academic Press, 1976, 251–280.

M. Spivak, *Calculus on Manifolds*, Benjamin, 1965.

K. J. Supowit, The relative neighbourhood graph, with an application to minimum spanning trees, *Journal of the ACM* **30** (1983) 428–448.

J. L. Synge and B. A. Griffith, *Principles of Mechanics*, McGraw-Hill, 1949.

R. Szeliski and D. Tonnesen, *Surface modeling with particle systems*, DEC Cambridge Research Lab Memo, January 1991.

D. Terzopoulos and K. Fleischer, Deformable models, *The Visual Computer* **4** (1988) 306–331.

G. Toussaint, Pattern recognition and computational complexity, *Proceedings of the Fifth International Conference on Pattern Recognition*, December 1980, 1324–1347.

G. Toussaint (ed.), *Computational Geometry*, North-Holland, 1985.

G. Toussaint (ed.), *Computational Morphology*, North-Holland, 1988.

G. Turk, Generating textures on arbitrary surfaces using reaction-diffusion, *Computer Graphics* **25** (1991) 289–298 (Proceedings of SIGGRAPH '91).

L. C. Velho, Adaptive polygonization of implicit surfaces using simplicial decomposition and boundary constraints, *Proceedings of Eurographics '90*, September 1990, 125–136.

V. Vlassopoulos, Adaptive polygonization of parametric surfaces, *The Visual Computer* **6** (1990) 291–298.

W. N. Waggenspack and D. C. Anderson, Piecewise parametric approximations for algebraic curves, *Computer Aided Geometric Design* **6** (1989) 33–53.

Y. Yomdin, Sard's theorem and its improved versions in numerical analysis, in: E. L. Allgower and K. Georg (ed.), Computational Solution of Nonlinear Systems of Equations, *Lecture Notes in Applied Mathematics* **26** (1990) 701–706.

C. T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, *IEEE Transactions on Computers* **C-20** (1971) 68–86.

Index