Applied Mathematical Sciences

153

Stanley Osher
Ronald Fedkiw

Level Set Methods and Dynamic Implicit Surfaces

Springer

Stanley Osher Ronald Fedkiw

Level Set Methods and Dynamic Implicit Surfaces

With 99 Figures, Including 24 in Full Color

Springer

Stanley Osher
Department of Mathematics
University of California
at Los Angeles
Los Angeles, CA 90095-1555
USA
sjo@math.ucla.edu

Ronald Fedkiw
Department of Computer Science
Stanford University
Stanford, CA 94305-9020
USA
fedkiw@cs.stanford.edu

Cover photos: Top left and right, hand and rat brain — Duc Nguyen and Hong-Kai Zhao. Center campfire — Duc Nguyen and Nick Rasmussen and Industrial Light and Magic. Lower left and center, water glasses — Steve Marschner and Doug Enright.

Mathematics Subject Classification (2000): 65Mxx, 65C20, 65D17, 65-02, 65V10, 73V

Library of Congress Cataloging-in-Publication Data
Osher, Stanley.
Level set methods and dynamic implicit surfaces / Stanley Osher, Ronald Fedkiw
p. cm. – (Applied mathematical sciences ; 153)
Includes bibliographical references and index.
ISBN 0-387-95482-1 (alk. paper)
1. Level set methods. 2. Implicit functions. I. Fedkiw, Ronald P., 1968– II. Title.
III. Applied mathematical sciences (Springer-Verlag New York Inc.) ; v. 153
QA1.A647 vol. 153
[QC173.4]
510s–dc21
[515'.8]
2002020939

ISBN 0-387-95482-1 Printed on acid-free paper.

Printed in the United States of America.

9 8 7 6 5 4 3 2 SPIN 10920466

www.springer-ny.com

Springer-Verlag New York Berlin Heidelberg
A member of BertelsmannSpringer Science+Business Media GmbH

Dedicated with love to Katy, Brittany, and Bobbie

Preface

Scope, Aims, and Audiences

This book, *Level Set Methods and Dynamic Implicit Surfaces* is designed to serve two purposes:

Parts I and II introduce the reader to implicit surfaces and level set methods. We have used these chapters to teach introductory courses on the material to students with little more than a fundamental math background. No prior knowledge of partial differential equations or numerical analysis is required. These first eight chapters include enough detailed information to allow students to create working level set codes from scratch.

Parts III and IV of this book are based on a series of papers published by us and our colleagues. For the sake of brevity, a few details have been occasionally omitted. These chapters do include thorough explanations and enough of the significant details along with the appropriate references to allow the reader to get a firm grasp on the material.

This book is an introduction to the subject. We have given examples of the utility of the method to a diverse (but by no means complete) collection of application areas. We have also tried to give complete numerical recipes and a self-contained course in the appropriate numerical analysis. We believe that this book will enable users to apply the techniques presented here to real problems.

The level set method has been used in a rapidly growing number of areas, far too many to be represented here. These include epitaxial growth, optimal design, CAD, MEMS, optimal control, and others where the simulation

of moving interfaces plays a key role in the problem to be solved. A search of "level set methods" on the Google website (which gave over 2,700 responses as of May 2002) will give an interested reader some idea of the scope and utility of the method. In addition, some exciting advances in the technology have been made since we began writing this book. We hope to cover many of these topics in a future edition. In the meantime you can find some exciting animations and moving images as well as links to more relevant research papers via our personal web sites: `http://graphics.stanford.edu/~fedkiw` and `http://www.math.ucla.edu/~sjo/`.

Acknowledgments

Many people have helped us in this effort. We thank the following colleagues in particular: Steve Marschner, Paul Romburg, Gary Hewer, and Steve Ruuth for proofreading parts of the manuscript, Peter Smereka and Li-Tien Cheng for providing figures for the chapter on Codimension-Two Objects, Myungjoo Kang for providing figures for the chapter on Motion Involving Mean Curvature and Motion in the Normal Direction, Antonio Marquina and Frederic Gibou for help with the chapter on Image Restoration, Hong-Kai Zhao for help with chapter 13, Reconstruction of Surfaces from Unorganized Data Points, and Luminita Vese for help with the chapter on Snakes, Active Contours, and Segmentation. We particularly thank Barry Merriman for his extremely valuable collaboration on much of the research described here. Of course we have benefitted immensely from collaborations and discussions with far too many people to mention. We hope these colleagues and friends forgive us for omitting their names.

We would like to thank the following agencies for their support during this period: ONR, AFOSR, NSF, ARO, and DARPA. We are particularly grateful to Dr. Wen Masters of ONR for suggesting and believing in this project and for all of her encouragement during some of the more difficult times.

Finally, we thank our families and friends for putting up with us during this exciting, but stressful period.

Los Angeles, California — Stanley Osher
Stanford, California — Ronald Fedkiw

Contents

Part I
Implicit Surfaces

In the next two chapters we introduce implicit surfaces and illustrate a number of useful properties, focusing on those that will be of use to us later in the text. A good general review can be found in [16]. In the first chapter we discuss those properties that are true for a general implicit representation. In the second chapter we introduce the notion of a signed distance function with a Euclidean distance metric and a "±" sign used to indicate the inside and outside of the surface.

1
Implicit Functions

1.1 Points

In one spatial dimension, suppose we divide the real line into three distinct pieces using the points $x = -1$ and $x = 1$. That is, we define $(-\infty, -1)$, $(-1, 1)$, and $(1, \infty)$ as three separate subdomains of interest, although we regard the first and third as two disjoint pieces of the same region. We refer to $\Omega^- = (-1, 1)$ as the *inside* portion of the domain and $\Omega^+ = (-\infty, -1) \cup (1, \infty)$ as the *outside* portion of the domain. The border between the inside and the outside consists of the two points $\partial\Omega = \{-1, 1\}$ and is called the *interface*. In one spatial dimension, the inside and outside regions are one-dimensional objects, while the interface is less than one-dimensional. In fact, the points making up the interface are zero-dimensional. More generally, in $\Re^n$, subdomains are n-dimensional, while the interface has dimension $n - 1$. We say that the interface has codimension one.

In an *explicit* interface representation one explicitly writes down the points that belong to the interface as we did above when defining $\partial\Omega = \{-1, 1\}$. Alternatively, an *implicit* interface representation defines the interface as the isocontour of some function. For example, the zero isocontour of $\phi(x) = x^2 - 1$ is the set of all points where $\phi(x) = 0$; i.e., it is exactly $\partial\Omega = \{-1, 1\}$. This is shown in Figure 1.1. Note that the implicit function $\phi(x)$ is defined throughout the one-dimensional domain, while the isocontour defining the interface is one dimension lower. More generally, in $\Re^n$, the implicit function $\phi(\vec{x})$ is defined on all $\vec{x} \in \Re^n$, and its isocontour has dimension $n - 1$. Initially, the implicit representation might seem

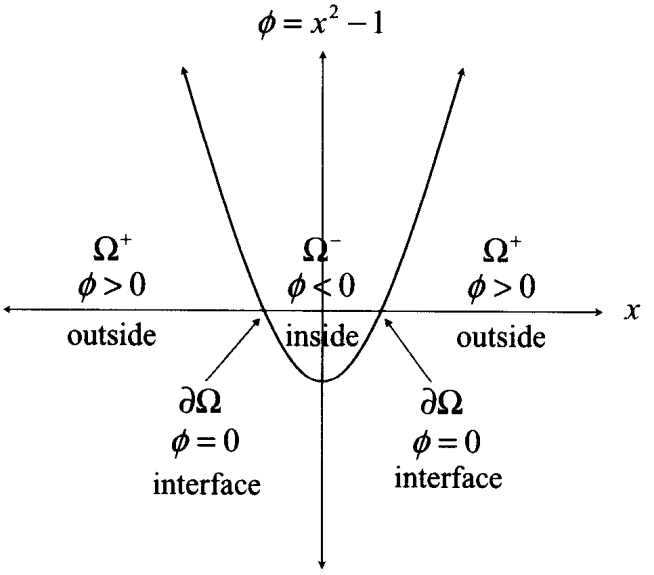

Figure 1.1. Implicit function $\phi(x) = x^2 - 1$ defining the regions Ω^- and Ω^+ as well as the boundary $\partial\Omega$

wasteful, since the implicit function $\phi(\vec{x})$ is defined on all of $\Re^n$, while the interface has only dimension $n-1$. However, we will see that a number of very powerful tools are readily available when we use this representation.

Above, we chose the $\phi(x) = 0$ isocontour to represent the lower-dimensional interface, but there is nothing special about the zero isocontour. For example, the $\hat{\phi}(x) = 1$ isocontour of $\hat{\phi}(x) = x^2$, defines the same interface, $\partial\Omega = \{-1, 1\}$. In general, for any function $\hat{\phi}(\vec{x})$ and an arbitrary isocontour $\hat{\phi}(\vec{x}) = a$ for some scalar $a \in \Re$, we can define $\phi(\vec{x}) = \hat{\phi}(\vec{x}) - a$, so that the $\phi(\vec{x}) = 0$ isocontour of ϕ is identical to the $\hat{\phi}(\vec{x}) = a$ isocontour of $\hat{\phi}$. In addition, the functions ϕ and $\hat{\phi}$ have identical properties up to a scalar translation a. Moreover, the partial derivatives of ϕ are the same as the partial derivatives of $\hat{\phi}$, since the scalar vanishes upon differentiation. Thus, throughout the text all of our implicit functions $\phi(\vec{x})$ will be defined so that the $\phi(\vec{x}) = 0$ isocontour represents the interface (unless otherwise specified).

1.2 Curves

In two spatial dimensions, our lower-dimensional interface is a curve that separates $\Re^2$ into separate subdomains with nonzero areas. Here we are limiting our interface curves to those that are *closed*, so that they have clearly defined interior and exterior regions. As an example, consider $\phi(\vec{x}) =$

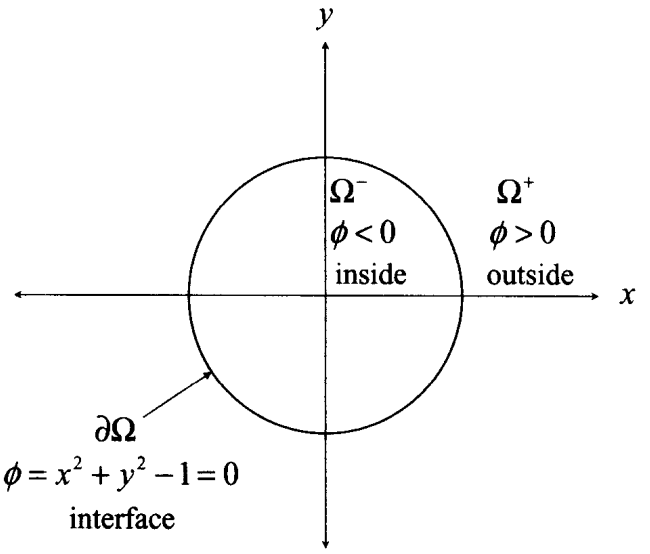

Figure 1.2. Implicit representation of the curve $x^2 + y^2 = 1$.

$x^2 + y^2 - 1$, where the interface defined by the $\phi(\vec{x}) = 0$ isocontour is the unit circle defined by $\partial\Omega = \{\vec{x} \mid |\vec{x}| = 1\}$. The interior region is the unit open disk $\Omega^- = \{\vec{x} \mid |\vec{x}| < 1\}$, and the exterior region is $\Omega^+ = \{\vec{x} \mid |\vec{x}| > 1\}$. These regions are depicted in Figure 1.2. The explicit representation of this interface is simply the unit circle defined by $\partial\Omega = \{\vec{x} \mid |\vec{x}| = 1\}$.

In two spatial dimensions, the explicit interface definition needs to specify all the points on a curve. While in this case it is easy to do, it can be somewhat more difficult for general curves. In general, one needs to parameterize the curve with a vector function $\vec{x}(s)$, where the parameter s is in $[s_o, s_f]$. The condition that the curve be closed implies that $\vec{x}(s_o) = \vec{x}(s_f)$.

While it is convenient to use analytical descriptions as we have done so far, complicated two-dimensional curves do not generally have such simple representations. A convenient way of approximating an explicit representation is to discretize the parameter s into a finite set of points $s_o < \cdots < s_{i-1} < s_i < s_{i+1} < \cdots < s_f$, where the subintervals $[s_i, s_{i+1}]$ are not necessarily of equal size. For each point s_i in parameter space, we then store the corresponding two-dimensional location of the curve denoted by $\vec{x}(s_i)$. As the number of points in the discretized parameter space is increased, so is the resolution (detail) of the two-dimensional curve.

The implicit representation can be stored with a discretization as well, except now one needs to discretize all of $\Re^2$, which is impractical, since it is unbounded. Instead, we discretize a bounded subdomain $D \subset \Re^2$. Within this domain, we choose a finite set of points (x_i, y_i) for $i = 1, \ldots, N$ to discretely approximate the implicit function ϕ. This illustrates a drawback of the implicit surface representation. Instead of resolving a one-dimensional

interval $[s_o, s_f]$, one needs to resolve a two-dimensional region D. More generally, in $\Re^n$, a discretization of an explicit representation needs to resolve only an $(n-1)$-dimensional set, while a discretization of an implicit representation needs to resolve an n-dimensional set. This can be avoided, in part, by placing all the points $\vec{x}$ very close to the interface, leaving the rest of D unresolved. Since only the $\phi(\vec{x}) = 0$ isocontour is important, only the points $\vec{x}$ near this isocontour are actually needed to accurately represent the interface. The rest of D is unimportant. Clustering points near the interface is a *local* approach to discretizing implicit representations. (We will give more details about local approaches later.) Once we have chosen the set of points that make up our discretization, we store the values of the implicit function $\phi(\vec{x})$ at each of these points.

Neither the explicit nor the implicit discretization tells us where the interface is located. Instead, they both give information at sample locations. In the explicit representation, we know the location of a finite set of points on the curve, but do not know the location of the remaining infinite set of points (on the curve). Usually, interpolation is used to approximate the location of points not represented in the discretization. For example, piecewise polynomial interpolation can be used to determine the shape of the interface between the data points. Splines are usually appropriate for this. Similarly, in the implicit representation we know the values of the implicit function ϕ at only a finite number of points and need to use interpolation to find the values of ϕ elsewhere. Even worse, here we may not know the location of any of the points on the interface, unless we have luckily chosen data points $\vec{x}$ where $\phi(\vec{x})$ is exactly equal to zero. In order to locate the interface, the $\phi(\vec{x}) = 0$ isocontour needs to be interpolated from the known values of ϕ at the data points. This is a rather standard procedure accomplished by a variety of contour plotting routines.

The set of data points where the implicit function ϕ is defined is called a *grid*. There are many ways of choosing the points in a grid, and these lead to a number of different types of grids, e.g., unstructured, adaptive, curvilinear. By far, the most popular grids, are Cartesian grids defined as $\{(x_i, y_j) \mid 1 \le i \le m, 1 \le j \le n\}$. The natural orderings of the x_i and y_j are usually used for convenience. That is, $x_1 < \cdots < x_{i-1} < x_i < x_{i+1} < \cdots < x_m$ and $y_1 < \cdots < y_{j-1} < y_j < y_{j+1} < \cdots < y_n$. In a *uniform* Cartesian grid, all the subintervals $[x_i, x_{i+1}]$ are equal in size, and we set $\triangle x = x_{i+1} - x_i$. Likewise, all the subintervals $[y_j, y_{j+1}]$ are equal in size, and we set $\triangle y = y_{j+1} - y_j$. Furthermore, it is usually convenient to choose $\triangle x = \triangle y$ so that the approximation errors are the same in the x-direction as they are in the y-direction. By definition, Cartesian grids imply the use of a rectangular domain $D = [x_1, x_m] \times [y_1, y_n]$. Again, since ϕ is important only near the interface, a local approach would indicate that many of the grid points are not needed, and the implicit representation can be optimized by storing only a subset of a uniform Cartesian grid. The Cartesian grid points that are not sufficiently near the interface can be discarded.