Spring 2011

Image based modeling of complex boundaries

Seth Ian Dillard
University of Iowa

Follow this and additional works at: https://ir.uiowa.edu/etd
Part of the Mechanical Engineering Commons

IMAGE BASED MODELING OF COMPLEX BOUNDARIES

by

Seth Ian Dillard

An Abstract

Of a thesis submitted in partial fulfillment
of the requirements for the Doctor of
Philosophy degree in Mechanical Engineering
in the Graduate College of
The University of Iowa

May 2011

Thesis Supervisors: Professor Uday Kumar
Assistant Professor James Buchholz

ABSTRACT

One outstanding challenge to understanding the behaviors of organisms and other complexities found in nature through the use of computational fluid dynamics simulations lies in the ability to accurately model the highly tortuous geometries and motions they generally exhibit. Descriptions must be created in a manner that is amenable to definition within some operative computational domain, while at the same time remaining fidelitous to the essence of what is desired to be understood. Typically models are created using functional approximations, so that complex objects are reduced to mathematically tractable representations. Such reductions can certainly lead to a great deal of insight, revealing trends by assigning parameterized motions and tracking their influence on a virtual surrounding environment. However, simplicity sometimes comes at the expense of fidelity; pared down to such a degree, simplified geometries evolving in prescribed fashions may fail to identify some of the essential physical mechanisms that make studying a system interesting to begin with. In this thesis, and alternative route to modeling complex geometries and behaviors is offered, basing its methodology on the coupling of image analysis and level set treatments. First a semi-Lagrangian method is explored, whereby images are utilized as a means for creating a set of surface points that describe a moving object. Later, points are dispensed with altogether, giving in the end a fully Eulerian representation of complex moving geometries that requires no surface meshing and that translates imaged objects directly to level sets without unnecessary tedium. The final framework outlined here represents a completely novel approach to modeling that combines image denoising, segmentation, optical flow, and morphing with level set- based embedded sharp interface methods to produce models that would be difficult to generate any other way.

Abstract Approved:	______________________
	Thesis Supervisor
	______________________
	Title and Department
	______________________
	Date
	______________________
	Thesis Supervisor
	______________________
	Title and Department
	______________________
	Date

IMAGE BASED MODELING OF COMPLEX BOUNDARIES

by

Seth Ian Dillard

A thesis submitted in partial fulfillment
of the requirements for the Doctor of
Philosophy degree in Mechanical Engineering
in the Graduate College of
The University of Iowa

May 2011

Thesis Supervisors: Professor Uday Kumar
Assistant Professor James Buchholz

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Seth Ian Dillard

has been approved by the Examining Committee
for the thesis requirement for the Doctor of Philosophy
degree in Mechanical Engineering at the May 2011 graduation.

Thesis Committee: ______________________________
Uday Kumar, Thesis Supervisor

____
James Buchholz, Thesis Supervisor

____
Krishnan Chandran

____
Jia Lu

____
Joseph Reinhardt

____
Sarah Vigmostad

To Annie and Estie

ACKNOWLEDGMENTS

I would like to thank my advisor Uday for sensing potential where some didn't, and for providing the patience, vision, and friendship it took to see it realized.

I would like to thank my co-advisor James for his guidance and insight, and his willingness to take me on as something of a project late in my present career and early in his.

I would like to thank my friends, who perennially gift the world with hilarity and knowledge, graciously and without pretense, simply because they like to.

I would like to thank my loving family, who have always supported my curious and creative endeavors. I'll keep trying to repay you.

Mostly I would like to thank Annie and Estie, for providing all of the things I've just mentioned and then, well, everything. I love you more than the world.

ABSTRACT

One outstanding challenge to understanding the behaviors of organisms and other complexities found in nature through the use of computational fluid dynamics simulations lies in the ability to accurately model the highly tortuous geometries and motions they generally exhibit. Descriptions must be created in a manner that is amenable to definition within some operative computational domain, while at the same time remaining fidelitous to the essence of what is desired to be understood. Typically models are created using functional approximations, so that complex objects are reduced to mathematically tractable representations. Such reductions can certainly lead to a great deal of insight, revealing trends by assigning parameterized motions and tracking their influence on a virtual surrounding environment. However, simplicity sometimes comes at the expense of fidelity; pared down to such a degree, simplified geometries evolving in prescribed fashions may fail to identify some of the essential physical mechanisms that make studying a system interesting to begin with. In this thesis, and alternative route to modeling complex geometries and behaviors is offered, basing its methodology on the coupling of image analysis and level set treatments. First a semi-Lagrangian method is explored, whereby images are utilized as a means for creating a set of surface points that describe a moving object. Later, points are dispensed with altogether, giving in the end a fully Eulerian representation of complex moving geometries that requires no surface meshing and that translates imaged objects directly to level sets without unnecessary tedium. The final framework outlined here represents a completely novel approach to modeling that combines image denoising, segmentation, optical flow, and morphing with level set- based embedded sharp interface methods to produce models that would be difficult to generate any other way.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION TO MODELING COMPLEX GEOMETRIES, MOTIONS, AND FLUID TRANSPORT MECHANISMS

1.1 Motivation

Something is said to be complex when it is difficult to analyze or disentangle [1]. Although it takes many forms, the quality of complexity is easily recognized when encountered; it is marked often with large numbers of interrelated components, actions that are not easily predicted, or other physical characteristics that cannot be described succinctly without a good deal of abstraction. Living organisms represent well the essence of complexity, and thus much of this work is directed toward developing simple and effective methods for modeling the time-dependent geometries and fluid transport mechanisms found in biological systems.

Fluid motion, specifically that generated by the motility of animals' internal and external surfaces for the purposes of locomotion through a fluid environment, transporting heat and mass in solution or suspension, or producing and hearing sound, merits investigation because such processes are necessary for sustaining life. More than 10^9 years of evolved organic forms have lead to the earth's current population of species; whether viewed from a teleological or a mechanistic vantage, these species represent iterated designs that have remained successful for a very long time, in a system that often rewards efficiency with longevity [2-3]. Thus, life is established as something which imparts a great deal of engineering perspective in the very fact that it exists.

Locomotion through fluids is, for the most part, an external flow problem. When animals swim or fly, actuation of their external surfaces and appendages imparts momentum onto surrounding fluid material, which in turn generates the reactionary forces necessary for propulsion and control. In aqueous environments, undulatory motions of an elongated body or tail provide a fundamental means of propulsion that may be found in a large number of living examples, from primitive single-cell protozoa to the spermatozoa of higher species and the anguilliform swimming motion of the common eel [2]. The prevalence of this mode of transport among aquatic life suggests a level of optimality that warrants detailed study, and will be investigated as part of this work using the American eel (*Anguilla rostrata*) as a prototype.

Equally important to sustaining life’s processes is the motility of internal surfaces, which produce fluid motions that transport heat and mass within living systems. The respiratory system, the cardiovascular system, and the gastrointestinal tract are all clear examples, and each is marked by the presence of unsteady or pulsatile flows through geometrically elaborate, deformable containing vessels over a wide range of Reynolds numbers [2]. Detailed studies of such complex systems have lead to enhanced insights that have proven useful in the advancement of biological sciences and engineering designs: Clinicians now have a greater understanding of the distribution of wall shear stresses throughout the circulatory system, and its role in the advancement of lethal and common disease states such as atherosclerosis, stenoses, and aneurisms [4]; peristaltic pumps transport intravenous fluids and dialyzed blood in the same way that the veins or small intestine might, with predictable regularity and in a manner which minimizes damage to delicate constitutive structures, such as red blood cells and platelets; and peristaltic mixing throughout the gastrointestinal tract is suspected as one of the key mechanisms responsible for rapid nutrient absorption and subsequent reactions that enable energetically efficient movement [5].

Adding another layer of complexity is the fact that biological fluids are themselves complex systems. Chyme passing from the stomach mixes with pancreatico-biliary secretions to create a multiphase chemical-laden slurry with interactions occurring across a wide range of scales, from molecular to visible [5]. Blood, too, is made up of many disparate structures related through a multitude of coordinated interactions, all with the same singular purpose as all other biofluids: maintaining life. Of course, being alive is not a prerequisite for complexity; there are countless complex processes involving inanimate objects. Flows through porous media, multiphase transport phenomena, fluid-structure interactions, etc. are all sufficiently difficult to analyze and solve to earn the adjective "complex," and so describing and modeling them is not a straightforward undertaking. As such, the work herein attempts to make a contribution in the way of offering a route to modeling complex objects and behaviors in a general sense, with a wide range of applications. However, biological examples are often favored here as bases for methodological development due to the interest they hold outside numerical treatment.

1.2 The Conventional Route to Numerical Biofluid Calculations

Because biological systems change shape continually during movement, and because they generally do not come in shapes that are easy to define functionally, the tasks of describing their surfaces and interaction with fluids and computationally meshing them for CFD analysis have proven challenging. Current simulation techniques reported in the literature concerning animal locomotion primarily involve defining simplified model geometries based upon measurements taken of animals directly, followed by prescribing some sort of motion in order to replicate observed behaviors, e.g. modeling a fish as an ellipsoid propelled by an oscillating flat plate [6-7]. This makes mathematical representation of material reality more tractable. One readily apparent drawback to these current techniques is the intensity of labor involved in creating such models; for instance, animals must be measured in several places to allow for geometric reconstruction to proceed, and a surface mesh must be manually generated so that the model can be imported into a flow solver. The tedious nature of surface generation and kinematic analysis performed in this way may compromise resolution of finer animal body structures and motions in favor of a more qualitative description that is geometrically easier to deal with.

An alternative approach employs image analysis to directly define the shapes of complex objects in a computational flow domain based on their visual appearance. This allows for a more accurate representation of geometry and motion, and is an ongoing area of active research, particularly in the fields of medical imagery and computer vision [8-17]. However, conventional CFD modeling approaches still require surface mesh generation (or volume mesh generation, in 3D) in order to define complex and/or moving boundaries, making for tedious implementation (Figure 1-1). It is therefore desirable to develop a modeling framework that bypasses mesh generation altogether. To this end, we propose a completely Eulerian method which maps imaged objects directly onto a Cartesian mesh for CFD treatment using level set representations.

1.3 Introduction to Image Analysis

Digital images are made up of square pixels (or cubic *voxels* in 3D imaging), each possessing a single intensity or color value. Intensity values range from zero (no light intensity, or black) to 255 (white) with 254 shades of grey in between. Color values follow the same pattern, but with $(\mathrm{red}, \mathrm{green}, \mathrm{blue})$ vectors replacing scalar intensity levels. E.g. black has a vector representation of $(0, 0, 0)$, while medium green has a representation of $(0, 125, 0)$, etc. A color image has $256 \times 256 \times 256$, or about 16.8 million, possible pixel hues. Groups of pixels give an image its meaning, concertedly transforming it from an array of numbers or vectors to a picture that can be visually related to something real. More pixels describing a scene yield finer detail, or better resolution, and lend greater fidelity to objects being depicted.

1.3.1 Image Segmentation

Image segmentation is the process of delineating a digital image into different parts the way humans might during visualization, based upon relationships between groups of pixels that make up the image [13, 18]. More simply put, segmentation is the act of recognizing the outlines of shapes and distinguishing them from one another. Modern segmentation methods in the literature treat an image and its pixel values as a topological field of sorts, on which segmentation can be accomplished by computing pixel gradients or by minimizing a functional over the pixel field in a manner analogous to energy minimization [13, 18-19]. In general the more complex an image is according to our visual system, the more complicated a segmentation algorithm must be to adequately separate the scene into distinct objects.

The proposed modeling framework is based entirely on imaging, and so segmentation is one of the primary steps necessary for the transformation from imaged object to modeled surface to take place. Thus, quality segmentation is a key factor in the method's overall success, and warrants thorough investigation; the process is discussed in detail in Chapters 3 and 4.

1.3.2 Image Denoising

All real images have some kind of intrinsic corruption that is introduced during the acquisition process, whether it is the photoelectric noise possessed by digital images or the film-grain noise seen in photographs [20]. Thus, denoising makes up a large part of image processing. Here, we are primarily interested in using images to generate models via segmentation; a good model needs a good segment, and a good segment in turn needs a good noise-free image to start from. Unfortunately, even if noise has been minimized during the acquisition process to the fullest extent possible, circumstances typically dictate the need for some smoothing before an image is ready for conversion to a CFD model.

Image denoising algorithms are numerous, but most stem from the same diffusion-based methodology, a description of which comprises part of Chapter 4. Some methods emerging in the literature more recently involve a multi-resolution treatment provided by wavelet analysis using the discrete wavelet transform (DWT). In fact, the

DWT has become pervasive in several aspects of image processing, such as compression and feature identification, and so shall be given due introduction next for completeness.

1.3.3 The Discrete Wavelet Transform (DWT)

Like Fourier analysis, wavelet analysis is a means of quantifying a signal $s(t)$ in terms of a mathematical decomposition, though there are certainly some important differences between the two. The Fourier transform (Equation 1.1) is multi-frequency in nature, so that a signal may be represented as a collection of sinusoids possessing different frequencies and amplitudes:

$$\mathcal{F}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} s(t)\mathrm{e}^{-i\omega t}\,\mathrm{dt}, \qquad \textbf{(1.1)}$$

where

$$\mathrm{e}^{-i\omega t} = \cos\omega t - i\sin\omega t. \qquad \textbf{(1.2)}$$

Unfortunately, signals cast in Fourier space suffer from a loss of localized information due to the fact that signals $s(t)$ are being described with waves that extend infinitely for all time t from $-\infty$ to $+\infty$. Thus singular changes in a signal will affect its Fourier representation *everywhere* in Fourier space, rendering it impossible to know the location of a unique signal feature; or in fact whether it is unique at all, rather than a collection of periodic contributions.

In practice, an exercise will never be performed in which an infinitely populous signal is analyzed; that would take far too long – infinitely long, actually. Real signals come in the form of discretely sampled data sets, and so a discrete version of the Fourier transform is used to represent them in terms of their discrete spectra:

$$\mathcal{F}_k = \frac{1}{N}\sum_{j=0}^{N-1} s_j \mathrm{e}^{-i\left(\frac{2\pi kj}{N}\right)}. \quad \textbf{(1.3)}$$

This of course runs into the same difficulty as its continuous counterpart, namely a loss of localized signal features accompanying the switch to Fourier space.

In addition, while a continuous signal can be brought back from Fourier space with fidelity using the inverse Fourier transform (Equation 1.4)

$$s(t) = \int_{-\infty}^{\infty} \mathcal{F}(\omega)\mathrm{e}^{i\omega t}\,\mathrm{d}\omega, \quad \textbf{(1.4)}$$

a discrete signal with local discontinuities suffers from “Gibbs phenomena” when transformed back from Fourier space to physical space using the inverse discrete Fourier transform (Equation 1.5)

$$s_j = \frac{1}{N}\sum_{k=0}^{N-1} \mathcal{F}_k \mathrm{e}^{i\left(\frac{2\pi kj}{N}\right)}, \quad \textbf{(1.5)}$$

whereby large oscillations arise near signal discontinuities in the process of trying to represent them using continuous waves acting on a finite number of points.

In 1984, the wavelet transform was devised as a new method to overcome some of these difficulties inherent to Fourier analysis, prompted initially by a desire to better interpret seismic records [21]. Wavelet functions exist within a finite region (that is to say, they possess compact support), and so they can be scaled and shifted in ways that allow for signal location and frequency information to be preserved, rather than attempting to represent an entire signal using frequencies alone. This imparts wavelets with useful features like “shrinkage,” or compression, and adaptive multi-scale resolution capabilities, and it has led to their extensive use in speech analysis and image processing and transmission [22].

Wavelet decomposition is performed using a pair of functions – a *scaling* function (Equation 1.6) and a *wavelet* function (Equation 1.7) – operating on a signal as a filter bank that separates it into frequency bands [23].

$$\varphi(x) = \sum_{k=0}^{N-1} c_k \varphi(2x - k) \tag{1.6}$$

$$W(x) = \sum_{k=0}^{N-1} (-1)^k c_k \varphi(2x + k - N + 1) \tag{1.7}$$

Here x denotes a discrete location within the signal $s(x)$, and N is the number of coefficients describing the scaling and wavelet functions of choice.

In 1988, the mathematician Ingrid Daubechies discovered a family of filter functions possessing the desirable characteristics of orthogonality and maximum flatness at their beginning and end positions $\omega = 0$ and $\omega = \pi$ – traits which are especially amenable to discrete signal analysis [23]. Their utility has led to a certain pervasiveness of the Daubechies family of wavelets, to the point of becoming nearly a standard; all of the wavelet-based research publications referenced in this thesis employ Daubechies wavelets in some manner.

All Daubechies wavelets possess even numbers of coefficients, and satisfy three specific conditions [21]:

1. The scaling function is unique and maintains unit area (signal "energy" is conserved).

$$\sum_{k=0}^{N-1} c_k = 2 \tag{1.8}$$

2. Accuracy of the signal representation is maximized (to order $N/2$).

$$\sum_{k=0}^{N-1} (-1)^k k^m c_k = 0\,;\, m = 1,2, \dots, \frac{N}{2} - 1 \tag{1.9}$$

3. The wavelet system is orthogonal to allow for maximum compression of data.

$$\sum_{k=0}^{N-1} c_k c_{k+2m} = 0\,;\, m \neq 0 \tag{1.10}$$

$$\sum_{k=0}^{N-1} c_k^2 = 2 \tag{1.11}$$

For an in-depth treatment of these conditions, the reader is referred to [21, 23].

Of the Daubechies wavelets, the Daubechies-4 (often written Daub4 or D4 for short) is commonly used in signal processing due to its compact support (the number 4 denotes the fact that it possesses four coefficients), and as an instructional prototype due to its relative simplicity. Setting $N = 4$ in the three wavelet conditions just listed leads to the system of equations

$$c_0 + c_1 + c_2 + c_3 = 2 \tag{1.12}$$

$$c_0 - c_1 + c_2 - c_3 = 0 \tag{1.13}$$

$$-c_1 + 2c_2 - 3c_3 = 0 \tag{1.14}$$

$$c_0c_2 + c_1c_3 = 0 \tag{1.15}$$

$$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 2, \tag{1.16}$$

which have the solution

$$c_0 = \frac{1+\sqrt{3}}{4} \tag{1.17}$$

$$c_1 = \frac{3+\sqrt{3}}{4} \tag{1.18}$$

$$c_2 = \frac{3-\sqrt{3}}{4} \tag{1.19}$$

$$c_3 = \frac{1-\sqrt{3}}{4}. \tag{1.20}$$

Coefficients c_0 through c_3 comprise the Daub4 filter coefficients, and together give the Daub4 scaling function (Equation 1.21)

$$\varphi(x) = c_0\varphi(2x) + c_1\varphi(2x-1) + c_2\varphi(2x-2) + c_3\varphi(2x-3) \tag{1.21}$$

and Daub4 wavelet function (Equation 1.22)

$$W(x) = c_0\varphi(2x-3) - c_1\varphi(2x-2) + c_2\varphi(2x-1) - c_3\varphi(2x). \tag{1.22}$$

Interestingly, the Daubechies functions cannot be expressed in closed form, but can only be constructed through iteration. The process of generating Daubechies 4 scaling and wavelet functions may be illustrated following Newland [21]: Starting with a unit box situated on the interval $0 \leq x \leq 1$, the box is represented as ordinate 1 located at $x = 0$. A single application of the scaling function (Equation 1.21) $\varphi(x)$ to the signal gives four new ordinates $c_0, c_1, c_2,$ and c_3 located at $x = 0, 0.5, 1,$ and 1.5. Applying $\varphi(x)$ once again to the resulting signal gives eight new ordinates; c_0 at $x = 0$ contributes $c_0^2, c_0c_1, c_0c_2,$ and c_0c_3 at $x = 0, 0.25, 0.5,$ and 0.75, c_1 at $x = 0.5$ contributes $c_1c_0, c_1^2, c_1c_2,$ and c_1c_3 at $x = 0.5, 0.75, 1,$ and 1.25, etcetera, so that the eight new ordinates take the values $c_0^2, c_0c_1, c_0c_2 + c_1c_0, c_0c_3 + c_1^2, c_1c_2 + c_2c_0, c_1c_3 + c_2c_1, c_2^2 + c_3c_0, c_2c_3 + c_3c_1, c_3c_2,$ and c_3^2 at $x = 0, 0.25, 0.5, 0.75, 1, \dots, 2.25$ following the matrix scheme

$$[\varphi_2] = \begin{bmatrix} c_0 & 0 & 0 & 0 \\ c_1 & 0 & 0 & 0 \\ c_2 & c_0 & 0 & 0 \\ c_3 & c_1 & 0 & 0 \\ 0 & c_2 & c_0 & 0 \\ 0 & c_3 & c_1 & 0 \\ 0 & 0 & c_2 & c_0 \\ 0 & 0 & c_3 & c_1 \\ 0 & 0 & 0 & c_2 \\ 0 & 0 & 0 & c_3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} [1]. \qquad \textbf{(1.23)}$$

Continuing to iterate in this manner (as illustrated through six iterative steps in Figure 1-2) eventually leads to the fractal Daubechies-4 scaling function shown in Figure 1-3. The Daubechies-4 wavelet function is generated in a similar fashion (Figure 1-4 and Figure 1-5) following the matrix scheme

$$[W_2] = \begin{bmatrix} -c_3 & 0 & 0 & 0 \\ 0 & -c_3 & 0 & 0 \\ c_2 & 0 & -c_3 & 0 \\ 0 & c_2 & 0 & -c_3 \\ -c_1 & 0 & c_2 & 0 \\ 0 & -c_1 & 0 & c_2 \\ c_0 & 0 & -c_1 & 0 \\ 0 & c_0 & 0 & -c_1 \\ 0 & 0 & c_0 & 0 \\ 0 & 0 & 0 & c_0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} [1]. \qquad \textbf{(1.24)}$$

The scaling function is also known as a lowpass filter, and it acts as a moving average over the data being analyzed and thus serves to give a smooth representation of the original signal. The wavelet function, on the other hand, is a highpass filter, which acts as a moving difference function that picks out “bumps” in a signal [23]. The lowpass and highpass filters together make up an invertible system that allows a signal $s(x)$ to be separated into distinct frequency bands.

During application of the discrete wavelet transform, a multi-level representation is achieved by decomposing a signal into a series of scaling and wavelet coefficients representing increasingly large (doubling) signal intervals. The first level of the DWT decomposes the signal into level-1 scaling and wavelet coefficients φ_1 and W_1, then the second level decomposes the smooth signal representation φ_1 into a subset of level-2 scaling and wavelet coefficients φ_2 and W_2, and so on, doubling the signal interval size until the largest scale possible has been reached (Figure 1-6).

Since two linear operations are being performed separately on $s(x)$ to get the lowpass and highpass filter coefficients $\varphi(x)$ and $W(x)$, the result would be a data set that doubles in size with each level of transformation if all points in the signal were represented in wavelet space throughout the process. Moreover, this would require that the scaling and wavelet functions continuously increase in support, doubling their number of coefficients with each level of the transform. This is obviously undesirable from the standpoints of information storage and computational complexity. In practice, both of these problems are eliminated through *signal decimation*; that is, downsampling the signal population by half with each level of the DWT (Figure 1-7). As an example, if a signal is represented by, say, 128 points, the level-1 DWT will decompose the signal into 64 scaling coefficients and 64 wavelet coefficients; the level-2 DWT further decomposes the 64 level-1 scaling coefficients into 32 scaling coefficients and 32 wavelet coefficients, and so on. If the original signal is a power of 2 in length, this process may be continued (with signal wrap-around) until the final wavelet space representation consists of 2 scaling coefficients and 126 wavelet coefficients.

Figure 1-8 illustrates this process with a top hat signal represented by 128 points. Each level of the DWT gives a set of wavelet coefficients, and a set of scaling coefficients that represent a shifted and dilated version of the original signal. Most notable is the fact that a progression through multiple levels gives a *multiscale representation of discontinuities in the signal*, which is an important feature for coherent structure identification at scales of interest. This allows for a distinction to be made between signal noise, which tends to occur only at the smallest scales, and true features of interest, which have a signature across all scales, and will become pertinent in later chapters when image denoising is required.

As a final note, it should be pointed out that decomposing a signal in multiple dimensions with the DWT is a straightforward matter. In 2-D, this is accomplished by taking the DWT of the signal in one direction, X or Y, then taking the DWT of that result in the other direction. Order doesn’t matter, as the result will be the same regardless. The process is shown in Figure 1-9 using a 2-D top hat signal, and can be seen to be completely analogous to the 1-D transform, but with discontinuity detection now taking place in the horizontal (X), vertical (Y), and diagonal (XY) directions. 3-D transformation follows similarly.

1.4 The Level Set Method

The purpose of image segmentation within the image-based modeling framework being outlined here is to generate the solid surfaces that will be used to supply boundary conditions during fluid flow calculations. Present work in our Computational Thermofluids Group employs a level set [24-25] based Cartesian grid method to model moving boundaries. Methods in which interface effects are included in the discrete spatial operators acting in a continuum domain setting are called sharp interface methods, and the level set method falls into this category. Critical issues that arise in developing sharp interface Cartesian grid methods for moving boundary problems are:

i) Accurate representation of embedded interfaces- Explicit surface tracking (involving surface meshing and re-meshing) can be challenging for complex moving boundaries, particularly in the presence of sharp edges, cusps, instabilities and topological changes in the boundaries; geometric details such as intersections between the triangulated surface mesh and the underlying flow solver mesh need to be computed repeatedly.

   The level set technique presents a solution to these problems, because it is an implicit interface representation with built-in regularization due to the entropy-satisfying solutions obtained from level set advection. In this approach, the signed normal distance to an interface is stored at each mesh point in a narrow band field surrounding the interface. The interface is implicitly contained in this information as the zero-level isocontour of the level set field, and can be deduced to desired accuracy with a combination of a dense Cartesian mesh and higher order interface tracking schemes. Representation of all interfaces (whether solid-fluid or fluid-fluid) using level sets facilitates simple discretization at computational points adjacent to interfaces.

ii) Computing flows around immersed boundaries- Since interfaces are allowed to pass through the Cartesian mesh in an arbitrary fashion using the level set method, finite volume discretization would require reshaping of cells cut by implicit surfaces. Here this is avoided by employing a finite-difference discretization of the strong form of the governing equations.

When adopting a finite-difference approach, the accuracy and conservation properties of the flow solver generally become a concern. However, for problems constructed on a Cartesian mesh, the deviation of the finite-difference approach from the finite-volume approach appears only at grid points adjoining an immersed boundary. At these embedded interfaces, mesh refinement is employed in order to minimize losses in mass that can result from maintaining grid cell constancy and employing finite differencing.

A second-order accurate Cartesian grid based finite-difference scheme is used to discretize the incompressible Navier-Stokes equations. The discretization essentially depends on tempering the differential operators with the level set distance function field. As a result, the present algorithm handles embedded solid-fluid interfaces (using the sharp-interface method detailed in this chapter) and fluid-fluid interfaces (using the Ghost Fluid method) [26] in a unified fashion.

1.4.1 Transport Equations

The governing equations for incompressible flow are:

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0 \qquad \textbf{(1.25)}$$

and

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} = -\boldsymbol{\nabla} p + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u}. \qquad \textbf{(1.26)}$$

In Equation 1.26, $\mathrm{Re} = \rho_o U_o L / \mu$ is the Reynolds number, where $\rho_o, U_o, L,$ and μ are the bulk fluid density, characteristic velocity, characteristic length and dynamic viscosity respectively. Scalar (temperature, species concentration) transport equations take the general form

$$\beta_t \frac{\partial \xi}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{u} \xi = \beta_d \nabla^2 \xi, \qquad \textbf{(1.27)}$$

where β_t and β_d are material constants associated with the time-dependent and diffusive terms respectively.

1.4.2 The Flow Solver

A cell-centered collocated arrangement of the flow variables is used to discretize the governing equations. A two-step fractional step method [27-28] is used to advance the solution in time. The first step evaluates an intermediate velocity by solving an unsteady advection-diffusion equation,

$$\frac{\boldsymbol{u}^{*}-\boldsymbol{u}^{n}}{\Delta t}=-\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}+\frac{1}{\mathrm{Re}}\nabla^{2}\boldsymbol{u}, \qquad \textbf{(1.28)}$$

where the intermediate velocity $\boldsymbol{u}^{*}$ is evaluated with central difference discretization schemes for convection and diffusion terms. The convective terms are treated explicitly and discretized using a second-order accurate Adams-Bashforth method:

$$\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}=\frac{1}{2}\left(3\boldsymbol{u}^{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}^{n}-\boldsymbol{u}^{n-1}\cdot\boldsymbol{\nabla}\boldsymbol{u}^{n-1}\right). \qquad \textbf{(1.29)}$$

The diffusion terms are treated semi-implicitly using a Crank-Nicholson scheme:

$$\frac{1}{\mathrm{Re}}\nabla^{2}\boldsymbol{u}=\frac{1}{2\mathrm{Re}}\left(\nabla^{2}\boldsymbol{u}^{*}+\nabla^{2}\boldsymbol{u}^{n}\right). \qquad \textbf{(1.30)}$$

The second fractional step involves the correction of the intermediate velocity field $\boldsymbol{u}^{*}$ to enforce mass conservation,

$$\frac{\boldsymbol{u}^{n+1}-\boldsymbol{u}^{*}}{\Delta t}=-\boldsymbol{\nabla}\pi, \qquad \textbf{(1.31)}$$

where a pseudo-pressure field π is evaluated to impose a divergence-free velocity field at time step $n+1$. This is done by taking the divergence of Equation (1.31) to obtain a Poisson equation:

$$\nabla^{2}\pi=\frac{\boldsymbol{\nabla}\cdot\boldsymbol{u}}{\Delta t}. \qquad \textbf{(1.32)}$$

The final semi-discrete form of the equations including each of the above discretization schemes is then

$$\frac{\boldsymbol{u}^{*}}{\Delta t}-\frac{1}{2\mathrm{Re}}\nabla^{2}\boldsymbol{u}^{*}=\frac{\boldsymbol{u}^{n}}{\Delta t}+\frac{\rho\boldsymbol{g}}{\mathrm{Fr}^{2}}+\frac{1}{2\mathrm{Re}}\nabla^{2}\boldsymbol{u}^{n}$$
$$-\frac{1}{2}\left(3\boldsymbol{u}^{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}^{n}-\boldsymbol{u}^{n-1}\cdot\boldsymbol{\nabla}\boldsymbol{u}^{n-1}\right) \qquad \textbf{(1.33)}$$

and

$$\nabla^2 \pi^{n+1} = \frac{\nabla \cdot \boldsymbol{u}^*}{\Delta t}, \tag{1.34}$$

where $\mathrm{Fr} = u_o / \sqrt{g_o L_o}$ is the Froude number, used if buoyancy effects are considered. The intermediate velocity is then corrected to obtain the final divergence-free velocity field:

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t \boldsymbol{\nabla} \pi. \tag{1.35}$$

The advection-diffusion equations for scalar (heat and species) transport are discretized in a similar manner, i.e.

$$\beta_t \frac{\xi^{n+1}}{\Delta t} - \frac{\beta_d}{2} \nabla^2 \xi^{n+1} = \beta_t \frac{\xi^n}{\Delta t} + \frac{\beta_d}{2} \nabla^2 \xi^n$$
$$-\frac{1}{2}\left(3\boldsymbol{u}^n \cdot \boldsymbol{\nabla} \xi^n - \boldsymbol{u}^{n-1} \cdot \boldsymbol{\nabla} \xi^{n-1}\right). \tag{1.36}$$

1.4.3 Implicit Interface Representation using Level Sets

Embedded surfaces are represented implicitly on the Cartesian mesh using a standard level set approach [24, 29-31]. The level set method defines a scalar field φ_l, where subscript l denotes the l^{th} interface embedded within the computational domain. The value of φ_l at any point is the signed normal distance from the l^{th} interface, with $\varphi_l < 0$ inside the immersed boundaries and $\varphi_l > 0$ outside. The interface location is thus implicitly embedded in the φ_l field, since the $\varphi_l = 0$ contour represents the l^{th} immersed boundary surface.

In the case of moving interfaces, boundary motion is tracked by advecting the level set using

$$(\varphi_l)_t + \boldsymbol{V}_l \cdot \boldsymbol{\nabla} \varphi_l = 0. \tag{1.37}$$

In Equation 1.37, $\boldsymbol{V}_l$ is the l^{th} level set velocity field which is derived directly from the physics of the problem. A fourth-order ENO scheme in space and fourth-order Runge-Kutta integration in time are used for the evolution of the level set field. Since $\boldsymbol{V}_l$ is prescribed only by the physics on the interface (i.e. on the zero-level set), the velocity value at grid points that lie in the narrow band around the zero-level set needs to be obtained. This is done by extension of the interfacial velocity [30] away from the front using

$$\psi_\tau + \boldsymbol{V}_{ext} \cdot \nabla\psi = 0, \qquad \textbf{(1.38)}$$

where ψ is the quantity (i.e. the interface velocity component $V_{l,x}$ or $V_{l,y}$) that needs to be extended away from the interface. A natural choice for the extension velocity is $\boldsymbol{V}_{ext} = \text{sign}(\varphi_l)\frac{\nabla\varphi_l}{|\nabla\varphi_l|}$, which extends the velocity in a normal direction outward from an interface; this populates the narrow band around each interface in the time $\tau = O(\Delta x)$ with the necessary level set velocity information to impose dynamic boundary conditions on surrounding fluid regions. A reinitialization procedure [32-33] is carried out after level set advection to return the φ-field to a signed distance function, i.e. to satisfy $|\nabla\varphi_l| = 1$. Defining $(\varphi_l)_o$ as the level set field prior to re-initialization, the following equation is solved to steady state, in order to re-initialize the level set field for the next advection step:

$$(\varphi_l)_\tau + \boldsymbol{w} \cdot \nabla\varphi_l = \text{sign}(\varphi_l) \qquad \textbf{(1.39)}$$

$$\boldsymbol{w} = \text{sign}((\varphi_l)_o)\frac{\nabla(\varphi_l)_o}{|\nabla(\varphi_l)_o|}. \qquad \textbf{(1.40)}$$

In Equation 1.40, $\text{sign}((\varphi_l)_o) = \frac{(\varphi_l)_o}{\sqrt{(\varphi_l)_o^2+(\Delta x)^2}}$, with the initial condition $\varphi_l(\boldsymbol{x},0) = (\varphi_l)_o(\boldsymbol{x})$. Calculating the normal $\boldsymbol{n}$ and curvature κ of an interface using the level set field information is simple within this framework:

$$\boldsymbol{n} = \frac{\nabla\varphi_l}{|\nabla\varphi_l|} \qquad \textbf{(1.41)}$$

and

$$\kappa = -\nabla \cdot \boldsymbol{n}. \qquad \textbf{(1.42)}$$

1.4.4 Moving Boundaries

In the Eulerian sharp interface framework, a solid boundary moving across a grid point will cause the phase of the point to change from liquid to solid, or vice versa. Different approaches have been employed to handle that particular situation, including Ghost-Fluid and immersed boundary methods [26, 34-35] and fictitious domain methods [36], in which flow fields are computed within, as well as outside, the immersed solid object. In this way, when the boundary crosses over a grid point and changes the state from solid to fluid, the newly emerged fluid point simply takes on the flow field variables that were available at that point in the previous time step.

In the sharp interface method [37-38], as well as immersed interface methods [39-40], where the flow is computed separately in each sub-domain (fluid and solid) separated by an interface (and no ghost flow field exists in the solid), a scheme must be devised to obtain the flow field variables at newly emerged fluid points. Note that the converse case in which a grid point that was in the fluid phase transitions into the solid phase presents no issues, since the flow field is not computed in the solid phase. A newly emerged fluid grid point is defined by the condition $(\varphi_l)_{i,j}^{n+1}(\varphi_l)_{i,j}^{n} < 0$. Since the point was previously in the solid phase ($(\varphi_l)_{i,j}^{n} < 0$), it has no history in the fluid phase ($(\varphi_l)_{i,j}^{n+1} > 0$), and so $\boldsymbol{u}^n$ (and also ξ^n) does not exist for such a point. Therefore, these points must be evolved to the next time level $n+1$ in a special fashion. Note that since the pressure Poisson equation does not have a time-dependent term, the pressure in such a cell can be evaluated as usual once a $\boldsymbol{u}^*$ value is available after solving the momentum equation. The method to obtain $\boldsymbol{u}^n$ (and ξ^n) for such points follows along the lines detailed in [37-38], and is analogous to the approach taken in moving grid formulations when a fresh grid point is inserted following mesh refinement. The value at such points is obtained by interpolation from known values in the surrounding grid cells and on the moving boundary (where boundary conditions are specified).

For a particular time step in which a grid point changes phase from solid to fluid, the value taken by $\boldsymbol{u}^n$ at that point is found using a linear interpolation operator spanning neighboring points in the fluid and on the interface. The interpolation points that are picked depend on the orientation of the interface in the cell as illustrated in Figure 1-10. For the particular case in Figure 1-10, the value at the freshly cleared cell (i,j) is calculated as $\psi_{i,j}^{n} = \left(\chi_{-y}\psi_{i,j+1}^{n} + \psi_{I_{-y}}^{n}\right)/\left(1+\chi_{-y}\right)$, where χ_{-y} is the distance between the grid point (i,j) and the interfacial point I_{-y}. Interpolation points are chosen depending on the direction of the normal vector at I_{-y} $\left(\boldsymbol{n} = \left(n_x, n_y\right)\right)$ and the ratio n_y/n_x. For instance, in the above expression for the case in Figure 1-10, points I_{-y} and $(i,j+1)$ are chosen since $n_y > 0$ and $n_y > n_x$. Consistent with the differencing scheme at interface-adjacent grid points, the treatment at the newly emerged fluid points is first-order accurate. Note that this procedure is equivalent, in analogy with purely Lagrangian (moving grid) methods, to interpolating the values of variables to a newly inserted point after mesh refinement occurs from values at the old mesh points at the previous step (i.e. before refinement). In diffuse interface Eulerian methods (where interfaces may be captured using Volume of Fluid, level set, phase field, etc), the interfacial forces are spread over the mesh [41-42] and this issue of cross-over does not arise, since there is no clear-cut interface location and all properties are taken to vary smoothly over a few mesh points.

For a complete description of discretized equations used to solve for a fluid flow field domain embedded with dynamic sharp interfaces, the reader is referred to [29, 43-44].

1.5 Toward Image-Based Modeling without Surface Meshing

The dominant theme of this work involves simplifying the difficult task of modeling complex geometries and movements found in biological systems by dispensing with the need for a priori mathematical descriptions. Such descriptions, based upon assumptions or direct measurements, can be useful for elucidating the isolated effects of various parameters under scrutiny, as will be shown in Chapter 2. However, it is also sometimes desirable to understand the effects of a number of separate physical mechanisms that are unknown individually, but that concertedly produce a unique action that is interesting by virtue of its prevalence in nature. Progress has been made to this end through the course of image based modeling developments that will be described in the coming chapters.

A preliminary Lagrangian image-based modeling method was designed and implemented as the first step in this move away from purely mathematical descriptions, and will be the primary focus of chapter 3. It is shown to be effective in computationally capturing phenomena that would be difficult to reproduce without the benefit of imaging techniques, but, as it stands, still requires Lagrangian surface meshing for a complete description of the boundaries of interest interacting with fluids in the flow calculations. While dispensing with the need for functional approximations, this Lagrangian formulation of imaged objects comes with its own set of problems. Besides proving to be highly tedious to implement, the necessity of maintaining point correspondence between image frames in order to track boundary velocities and positions required surface meshes with equal numbers of evenly spaced points, which was found to lead to the introduction of unphysical boundary conditions. In addition, the need to smooth surface points once they were established often led to undesirable deformations in the modeled geometry.

Proposed is a much simpler image-based approach, in which image files are used directly as geometric and dynamic templates for building computational surfaces, using level sets, in a purely Eulerian setting (Figure 1-11), without the need to employ any surface meshing whatsoever. Through image segmentation, a level set field can be generated on an image mesh and then later mapped directly onto a flow solver mesh by interpolation.

Without the ability to generate a set of surface points that can be smoothed in space, one of the keys to the success of the proposed Eulerian method lies in generating high-quality segmentation contours directly on the image domain. This is not always straightforward on real images, which tend to contain noise or imperfect boundary representations that can lead to unusable segments. Thus, the first step toward a truly Eulerian approach was to develop a denoising facility to pre-process images and ready them for segmentation. Chapter 4 gives a background of denoising techniques, and compares several of the state-of-the-art methods currently used in image processing. The methods were tested thoroughly on synthetic images corrupted with noise, as well as on several real images, giving insight into their relative strengths and weaknesses, and ultimately leading to a selection for use in further developing the proposed modeling framework. Chapter 4 also includes a detailed discussion of image segmentation, and the construction of level set-based moving interfaces from segments as a means for obviating the tedious task of Lagrangian point treatment.

A motion tracking technique known in computer vision as *optical flow* is the subject of a literature review outlined in Chapter 5. Optical flow is simply a means for constructing an Eulerian vector field which possesses the same density as the set of images it is derived from, based on the assumption that brightness is a conserved quantity that is invariant to positioning within an image frame. Each pixel in an image is assigned a velocity vector, and the resultant velocity field quantifies the motion taking place in a set of images from one frame to the next. Because of the assumption of brightness constancy, as well as other assumptions that will be discussed in Chapter 5, such as brightness gradient constancy and image smoothness, optical flow solutions are fairly sensitive to noise present in the ordered image pairs on which they are obtained, reinforcing the necessity of the work described in Chapter 4.

Both linear and nonlinear optical flow methods were thoroughly tested on a variety of synthetic and real images, the results and discussion of which form the topic of Chapter 6. Nonlinear optical flow was found to perform well in most situations, lending the ability to describe boundary conditions in terms of a level set field, and thus completing the physical description necessary to construct computational models in a purely Eulerian setting. Applications of optical flow to particle flows and particle image velocimetry (PIV) analysis are also discussed.

Image sequence frame rates are much too slow to match the small time steps required for high-resolution CFD simulations, and so intermediate representations of object locations are required to fill in the missing information between image frames. It was initially hoped that optical flow would act as the primary mechanism by which this missing level set information could be obtained, but advecting surfaces using optical flow field information alone turned out to be insufficient. Thus, Chapter 7 opens with an introduction to another computer vision technique, *image morphing*, designed to fulfill precisely the purpose sought in completing the Eulerian modeling framework we set out to build – supplying the missing information between image frames so that the relatively low temporal resolution of image sequences no longer precludes modeling behavior on much finer time scales.

The discussion of morphing methods and their application to this work is followed by a final simulation that brings together all of the ideas outlined in this thesis, modeling a denoised and segmented video sequence of a swimming American eel as an embedded level set interface on a CFD mesh, complete with boundary conditions supplied via optical flow and motion supplied via morphing, demonstrating a complete unification of image processing and CFD analysis.

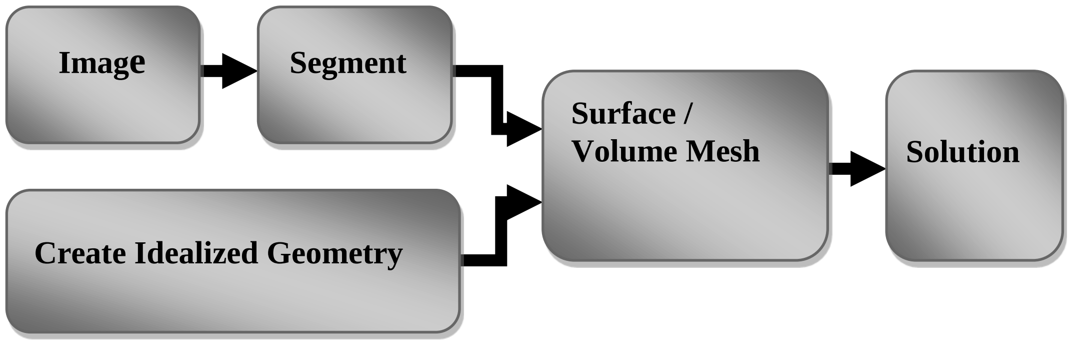


Figure 1-1. The conventional CFD routes to modeling complex geometries.

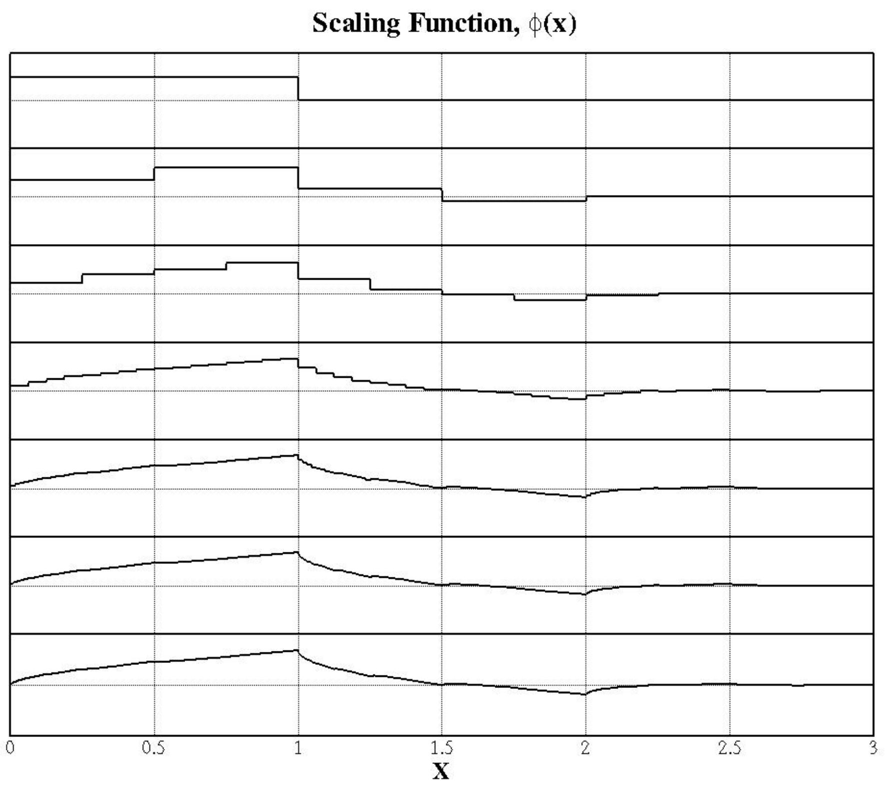


Figure 1-2. The Daubechies-4 scaling function, approximated by six convolutions of the four scaling coefficients with a unit box.

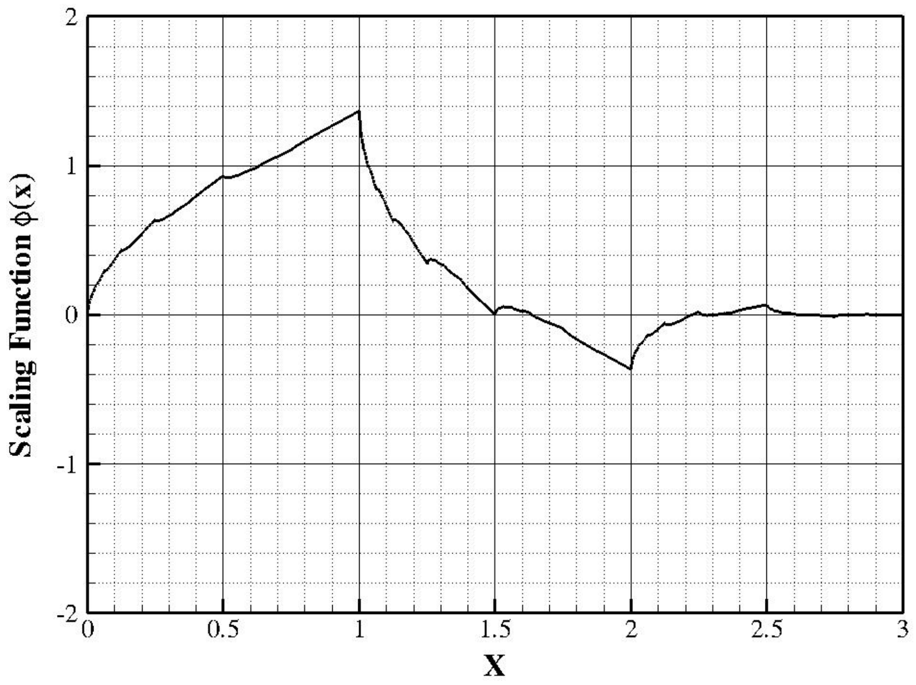


Figure 1-3. The Daubechies-4 scaling function.

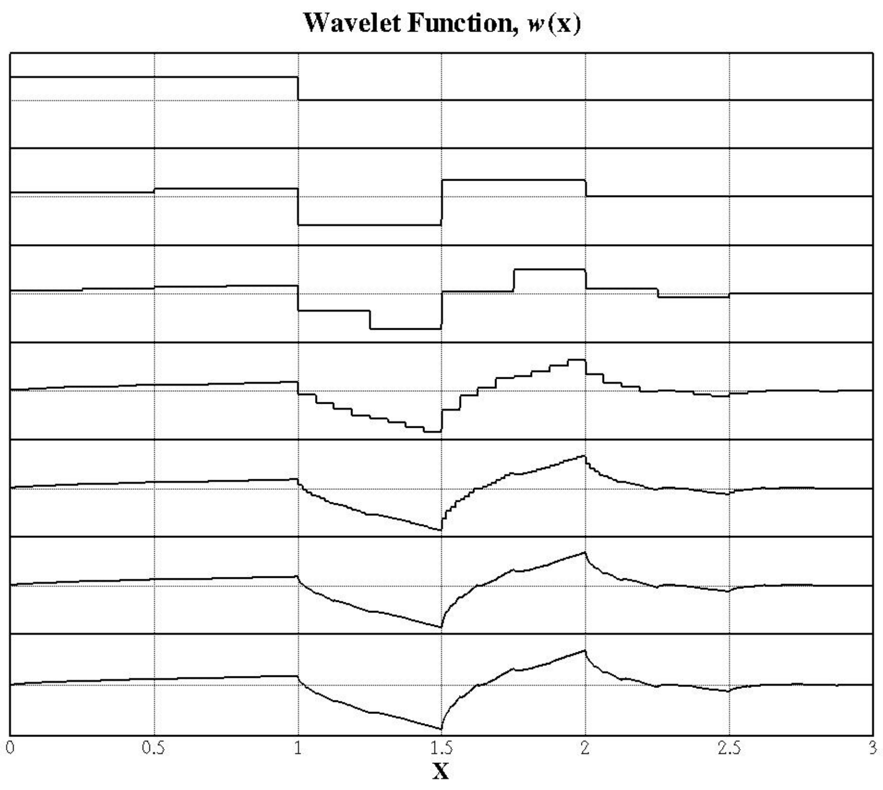


Figure 1-4. The Daubechies-4 wavelet function, approximated by six convolutions of the four wavelet coefficients with a unit box.

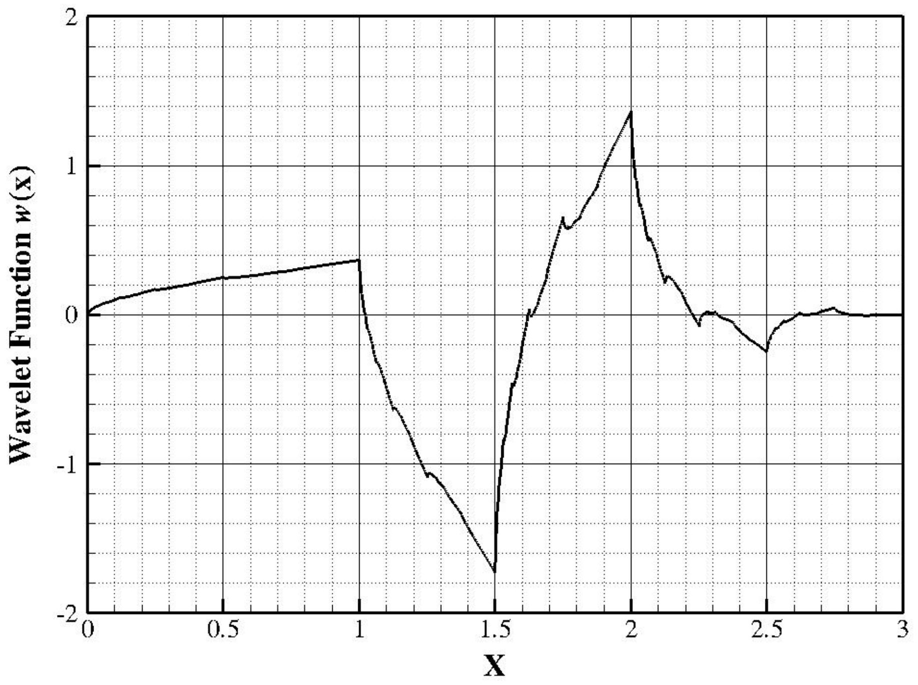


Figure 1-5. The Daubechies-4 wavelet function.

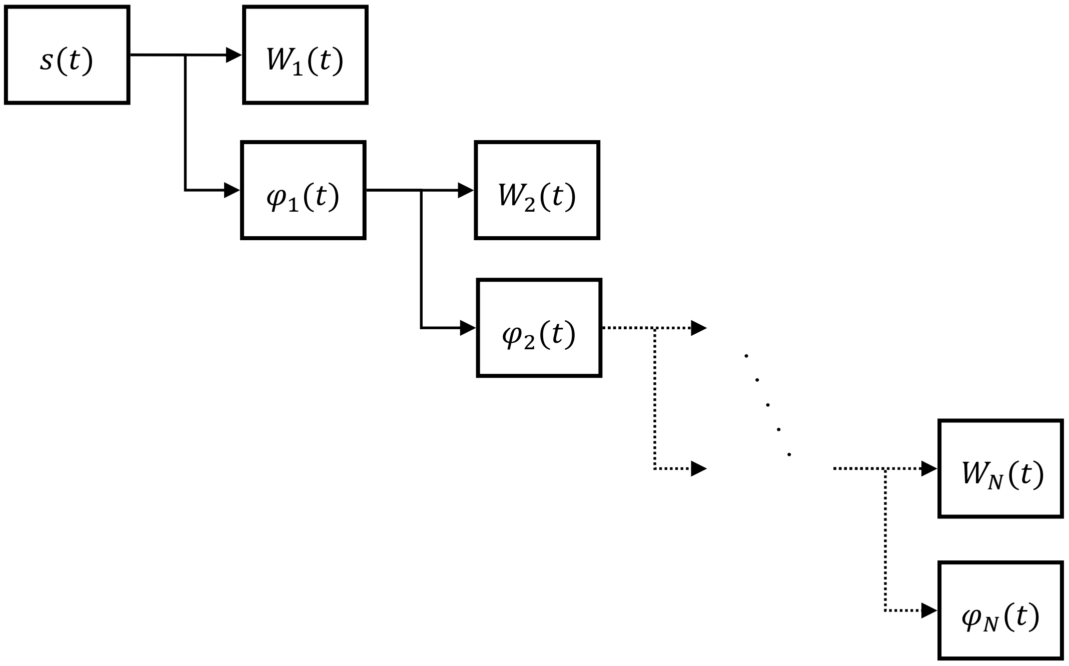


Figure 1-6. Wavelet decomposition takes place recursively, with scaling coefficients at each level being decomposed into a new set of scaling and wavelet coefficients until the signal is evaluated on the coarsest scale possible.

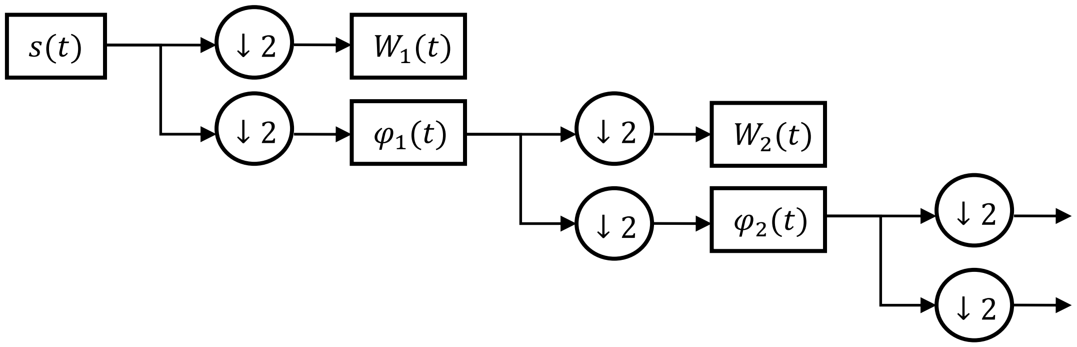


Figure 1-7. The signal is downsampled to half its original size with each level of wavelet decomposition.

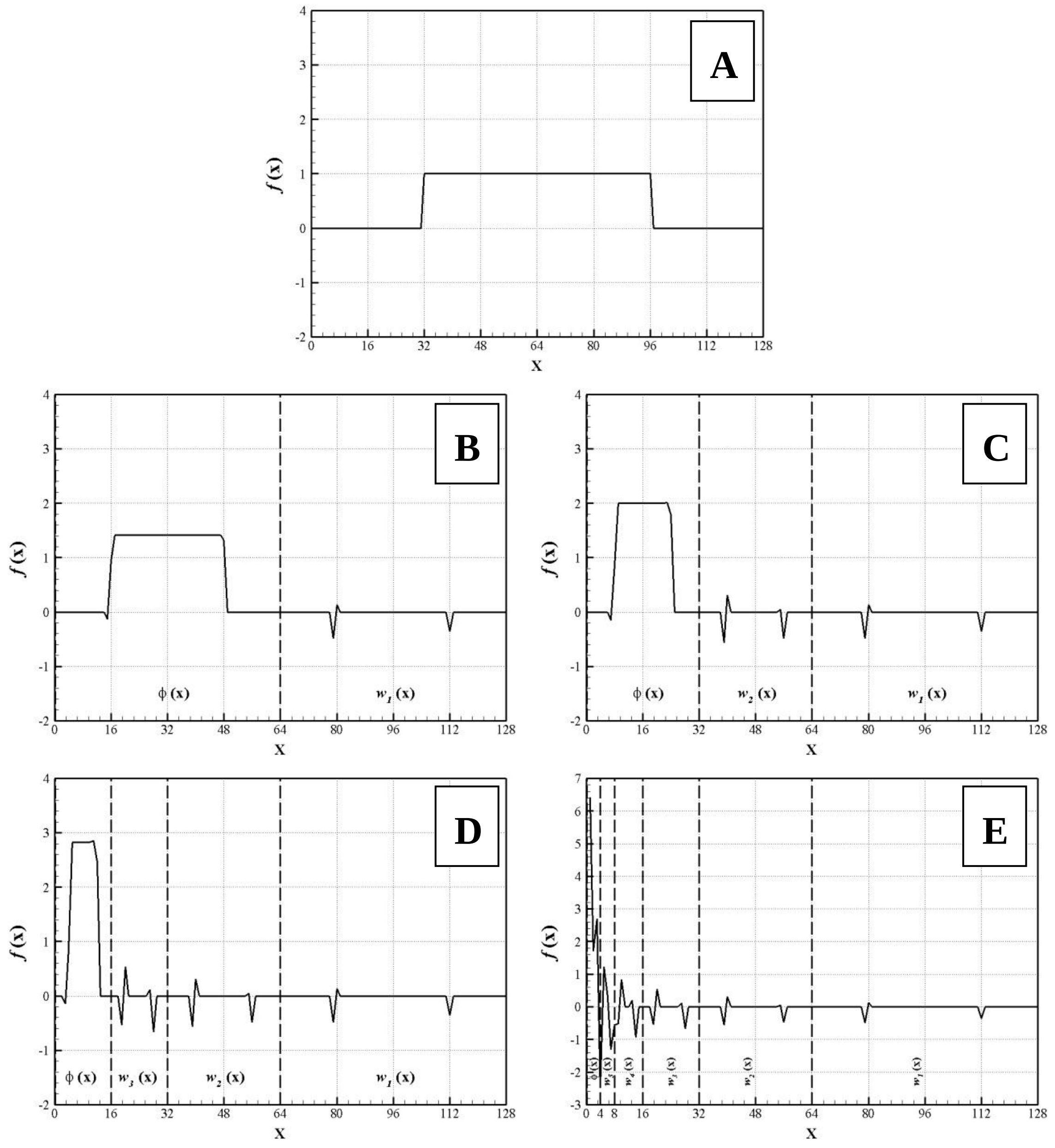


Figure 1-8. Decomposing a 1-D signal with the discrete wavelet transform using Daub4 wavelets: (A) Illustrative top-hat signal; (B) 1st level DWT; (C) 2nd level; (D) 3rd level; (E) signal after applying the DWT on 5 levels.

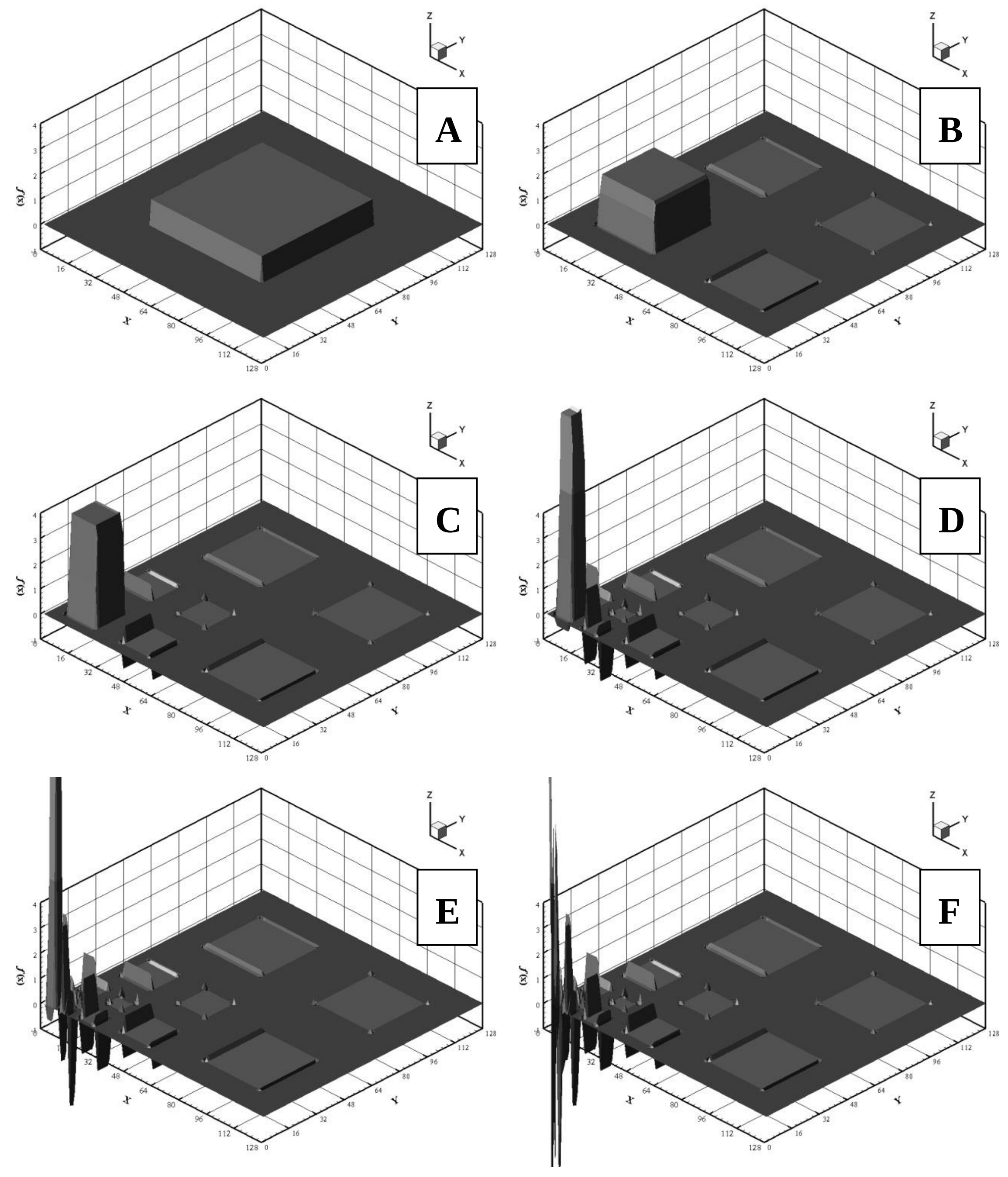


Figure 1-9. Decomposing a 2-D signal with the discrete wavelet transform using Daub4 wavelets: (A) Illustrative top-hat signal; (B) 1st level DWT; (C) 2nd level; (D) 3rd level; (E) 4th level; (F) signal after applying the DWT on 5 levels.

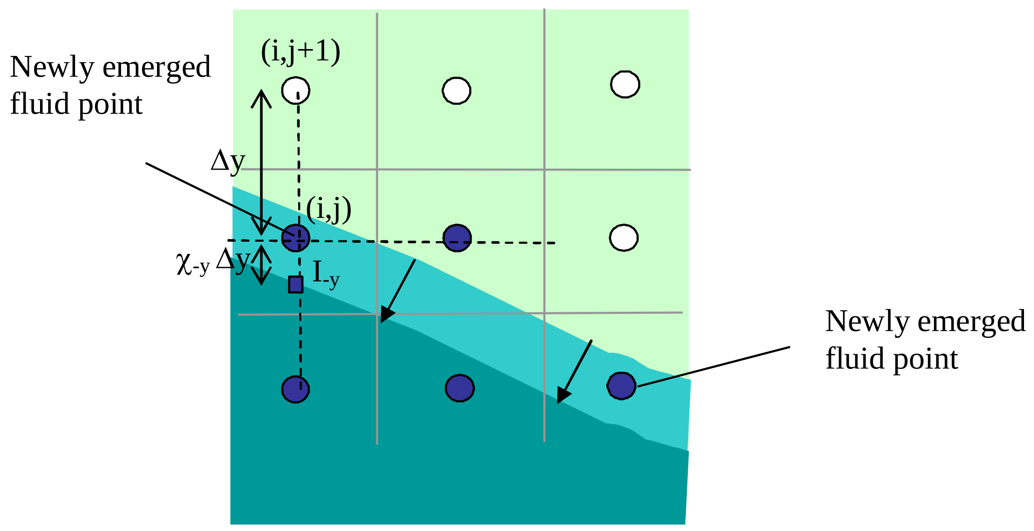


Figure 1-10. The emergence of a point from the solid phase to the fluid phase as the sharp solid/fluid interface traverses the mesh.

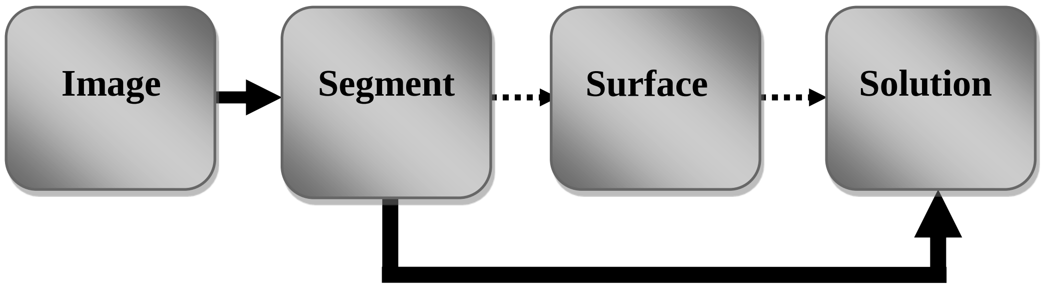


Figure 1-11. The aim of the present work is to develop a method that bypasses the tedious steps of mesh generation.

CHAPTER 2

IDEALIZED GEOMETRIC MODELING: MIXING AND TRANSPORT IN THE PROXIMAL GI TRACT AT THE ANTRO-DUODENAL JUNCTION

2.1 Introduction

While this thesis seeks to develop a modeling technique that unifies computer vision and CFD using level set methods, initial research work in the present course of study was performed using idealized geometries and motions that were mathematically prescribed based upon a priori assumptions. This is the standard modeling route generally taken, as it allows for a neat, tractable representation of complex physical phenomena. The task was to computationally model fluid flow through the pylorus (a regulator of sorts situated between the stomach and the small intestine), and was motivated by gastrointestinal research performed by Dr. Konrad Schulze of the Department of Internal Medicine, University of Iowa Health Care. Knowledge of the pylorus' action was primarily qualitative in nature, and so its functional representation was necessarily simple. Thus, a contrast is established between the level of detail offered by this type of idealization, and that of the image-based approaches being targeted for implementation during the remainder of this thesis project—a contrast which will become apparent in the following chapters.

The anatomical features and interesting properties of the pylorus that make it worth understanding from both clinical and physical perspectives are discussed in the forthcoming section.

2.2 Description of the Anatomy and Physiology

The gastric outlet to the small intestine, or antro-duodenal junction, is thought to have an impact on the effectiveness of mixing and subsequent absorption of nutrients in the gastrointestinal tract [5, 45-46]. Following ingestion, the stomach secretes gastric juice containing protein- denaturing hydrochloric acid and pepsinogen, and effectively grinds the mixture with a series of strong muscular contractions, producing a suspension of smaller solid particles and gastric juice known as chyme. Peristaltic contractions in the sinus of the stomach serve to transport chyme distally into the small intestine, where further enzymatic catalysis, transport, and digestion continue [5, 45-47]. The objective of this study is to determine what role, if any, the morphology of the antral-duodenal junction (Figure 2-1) plays in effecting transport and dynamic mixing of nutrients in the proximal duodenum. For this purpose, the flow of gastric effluent is computed *in silico*, modeling a set of experiments that were conducted on the cat gut in vitro [48].

The pylorus is comprised of a collection of tissue structures that connect the antrum to the duodenum. Its luminal diameter is controlled by a sphincter muscle complex that sets the resistance to bulk gastric effluent by regulating the tone of the pyloric orifice. During gastric digestion the proximal and distal pyloric muscle loops occlude the pyloric lumen, preventing premature discharge of unprocessed material to the duodenum. Once the stomach has completed its task of breaking down large solid agglomerates into smaller particles, the pylorus relaxes and peristaltic contractions in the antrum begin to force chyme distally. Antral contraction waves approach the pyloric orifice and, along with the sphincter complex and mucosal folds, cause steady constriction of the pyloric lumen. Chyme continues to be forcibly transported through this lumen until it is fully occluded, a process thought to potentiate an effluent jet into the superior duodenum [5, 45-46]. The geometry and contractile activity of the pylorus are thought to potentially affect gastric outflow and mixing effectiveness in the first part of the duodenum, as mixing of chyme with duodenal contents (in particular pancreatico-biliary secretions) is essential for digestion and absorption to proceed [5]. Thus, the ultimate aim of this study is to examine, through idealized geometric modeling and simulation, the effects of morphologic structure and pulsatility on the mechanics of flow and mixing in the antral-duodenal junction.

2.3 Computational Modeling of Transport and Mixing in the Antro-Duodenal Region

In vivo and in vitro laboratory experiments provide a great deal of information regarding the function of various components of the GI tract. However, better understanding the system's mechanics requires development of quantitative models that allow precise and isolated effects of these components to be more thoroughly investigated. Such models include the essential physical mechanisms pertinent to the physiology of the overall system. Clearly, inclusion of all the known mechanisms acting in the GI tract leads to a highly complex dynamical system. While a 3-dimensional transient model of the entire system would best reproduce actual physical behavior, and remains a long-term goal of this thesis work, it is kept in mind that there is still much physical insight to be revealed in idealized models that allow for examination of specific features over a range of flow and geometric parameters. This is the approach evaluated presently, with emphasis placed on morphology and dynamic properties of the pylorus.

In these preliminary calculations, basic aspects of fluid mechanics across the gastric outlet in simplified models of its geometry are examined. Several cases were constructed and analyzed in order to assess the importance of individual aspects of anatomical structure and kinematics. Flow calculations were performed in a series of channel-like domains (Figure 2-2) representing the distal antrum and superior duodenum, separated by various configurations of the pylorus. Of specific interest are the following aspects:

1. How does the pulsatile nature of gastric outflow affect the dynamics of nutrient transport and mixing?
2. What are the geometric features of the antral-duodenal junction that enhance or suppress transport/mixing in that region?
3. To what extent does wall motion in concert with pulsatile gastric outflow (aspect 1) through different geometric configurations (aspect 2) lead to enhanced transport and mixing?

To examine the aspects 1-3 listed above, flow fields were computed in several configurations: A) steady flow through a relaxed (symmetric) pylorus; B) pulsatile flow through a relaxed pylorus; C) pulsatile flow through a closing pylorus; D) pulsatile flow through a static pylorus, with asymmetry produced by tonicity of the proximal and distal pyloric muscle loops along with the pyloric torus; and E) pulsatile flow through a similarly asymmetrical, but closing, pylorus. F) In addition, some of these cases were varied over a range of gastric outflow rates (or effluent viscosities, alternately) to determine the effect of the pylorus on homogenization of chyme with different consistencies (or, more precisely, different viscosities).

Computational methods for solving equations of flow around moving boundaries in the manner chosen here were briefly outlined in Chapter 1, and have been thoroughly validated in several papers [29, 43-44, 49]. Following fluid mechanical conventions, the governing equations were "non-dimensionalized," or normalized, by dividing dimensional quantities by their corresponding representative scales in order to reveal parametric relationships known to govern physical behavior. In the cases presented here, length L^* is normalized by the characteristic unit length of the focal region (i.e. nominal diameter of the duodenum), velocity U^* by the maximum inflow velocity in the gastric pulse U_{max}, and time t^* by L/U_{max}. This normalization leads to an important non-dimensional quantity known as the Reynolds number, $\mathrm{Re} = \frac{LU}{\nu}$, where ν is the kinematic viscosity of the fluid. Physically, when the Reynolds number is very small, viscous effects dominate over fluid inertia. The Reynolds number for all cases studied in this investigation was varied from Re = 1 (low) to Re = 333 (moderate) — the upper-bound value obtained from data on duodenal dimensions and flow rates of normal saline introduced into the cat gut *in vitro* [48]. In moderate Reynolds number flows (Re ~ 100 to Re ~ 1000), the presence of relatively strong laminar vortical fluid patterns is expected. Such vortices can lead to rapid fluid mixing, particularly if they possess temporal and spatial variations, as will be shown [50-52].

The flow field equations solved in these models include the standard Navier-Stokes (continuity and momentum) equations (Equations 2.1 and 2.2), where $\boldsymbol{u}^*$ is a nondimensional fluid velocity vector with u^*, v^* (x- and y- velocity) components in 2D:

$$\nabla \cdot \boldsymbol{u} = 0, \qquad \textbf{(2.1)}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u}. \qquad \textbf{(2.2)}$$

Based on available data and non-dimensionalization techniques described above, the channel height in the antral and duodenal regions was set to 1.0, with the overall channel length from inlet to exit given 6.0 units length. The 2D computational domain for all cases was discretized into square grid elements measuring 0.05, 0.025, or 0.0125 units per side depending on the level of mesh refinement demanded by localized fluid motion to obtain a suitable solution [53].

For each model involving temporal variation of fluid velocity as seen in digestive systems, a pulsatile inlet velocity condition was imposed to represent rudimentary flow characteristics generated by periodic antral contractions (Figure 2-3). The nondimensional inlet velocity U^*_{inlet} was varied in a sinusoidal manner between 0.0 and $U^*_{inlet,max} = 1.0$, and the frequency of oscillation was set to $f = 0.4$ per time unit so that one pulse would have a total duration of 2.5 time units (corresponding to *in vitro* experimental data on cats [48]). To complete the cycle, each pulse was separated by a quiescent period ($U^*_{inlet} = 0.0$) of 1.0 time unit, giving a total cycle duration of $t^* = 3.5$. The quiescent phase was principally designed to represent the refractory period in which pyloric relaxation and return to an open position occurs, but it was retained for all cases, with or without dynamic wall events, to make results more directly comparable.

In addition to the standard flow variables (velocities and pressure), a conserved species transport equation was solved as a measure of spatial gastric effluent homogenization with duodenal contents. Mixing effectiveness was quantified in each case by calculating the scalar variance [54] of an evolving marker species (marked fluid region) that was initialized as a block measuring 1.0 unit in length and 0.5 unit height, with an initial concentration of 1.0. The species was given a small diffusivity of $D^* =$

10^{-5} in order to ensure advection-dominant transport, and placed vertically centered about the domain's longitudinal axis with the front (right) side of the block given the same horizontal coordinates as the distal terminus of the antrum. These scalar species blocks were evolved in each of the flow fields by solving the transport equation [55]:

$$\frac{\partial}{\partial t}\int_V \xi\, dV + \oint \xi(\boldsymbol{u}\cdot\boldsymbol{n})\, ds = \frac{1}{\mathrm{Pe}}\oint \boldsymbol{\nabla}\xi\cdot\boldsymbol{n}\, ds. \qquad \textbf{(2.3)}$$

In Equation 2.3, ξ represents scalar species concentration, $\mathrm{Pe} = (U_o L)/D$ is the Peclet number (the ratio of convective to diffusive transport in the fluid continuum), with U_o representing the characteristic bulk fluid velocity (ranging from 0.0 to $U^*_{inlet,max}$). Clearly, a small diffusivity coefficient results in very little diffusive transport, implying that the scalar species simply advects along fluid streamlines. The scalar species variance calculated for each case is defined by

$$\mathrm{var}(\xi) \equiv \langle \xi^2 \rangle = \int_V \left(\xi - \bar{\xi}\right)^2 dV \qquad \textbf{(2.4)}$$

In Equation 4, *V* is the volume of the computational domain and $\bar{\xi}$ is the volume averaged concentration of the initial species block, i.e.

$$\bar{\xi} = \frac{\int_V \xi\, dV}{\int_V dV} \qquad \textbf{(2.5)}$$

The concept of scalar variance is clarified by considering a rectangular domain of species concentration zero, within which is placed a scalar species block of concentration 1.0 (Figure 2-5). The mean concentration $\bar{\xi}$ of the field in this state is simply

$$\bar{\xi} = \frac{0.0\times(0.75\times0.25)+1.0\times(0.25\times0.25)}{(0.75\times0.25)+(0.25\times0.25)} = \frac{1}{4}, \qquad \textbf{(2.6)}$$

which gives a variance of

$$\langle \xi^2 \rangle = (0.0 - 0.25)^2 \times (0.75 \times 0.25)$$
$$+(1.0 - 0.25)^2 \times (0.25 \times 0.25) = \frac{3}{64}. \qquad \textbf{(2.7)}$$

In a field that that has become twice as "well mixed" (Figure 2-5), variance decreases accordingly with the increased level of species homogenization:

$$\langle \xi^2 \rangle = (0.0 - 0.25)^2 \times (0.5 \times 0.25)$$
$$+(0.5 - 0.25)^2 \times (0.5 \times 0.25) = \frac{1}{64}. \quad \textbf{(2.8)}$$

Scalar variance approaches zero as perfect mixing is more nearly achieved.

2.4 Computational Results

2.4.1 Steady Flow across the Gastric Outlet through a Relaxed Pylorus

Volumetrically, a large portion of gastric emptying of aqueous material is believed to occur across an initially relaxed pylorus [5, 45-47]. In its relaxed state, the torus and distal muscle loop of the human pylorus form a lumen of about 1 cm diameter between the antrum and duodenum, whose diameters are several times larger [5]. To isolate the effect of such a localized channel narrowing on flow, steady flow behavior (at Re = 333) was computed through the channel illustrated in Figure 2-2. This calculation was designed to establish a baseline for comparison with forthcoming cases exhibiting inflow pulsatility. *Note that in all of the results presented here the flow is from left to right, i.e. the antrum (inlet) lies to the left of the pylorus and the duodenum to the right*. A constant velocity of $U^*_{inlet} = 1.0$ was imposed at the channel inlet, with the resultant flow given 10.0 time units to propagate through the domain in order to achieve steady-state conditions. After this initial period, the scalar species marker was introduced as described previously and allowed to pass through the channel domain to the exit.

The dominant flow field feature in this first case consists of a pair of recirculation zones distal to the pylorus (Figure 2-7; only one is shown with streamlines due to symmetry.) While the marker fluid is distorted, it remains essentially unmixed as it passes into the superior duodenum. This is primarily due to the lack of dynamic vortical activity in the flow. Vorticity is limited to the separation region distal to the pylorus, segregated from the rest of the flow by a *separatrix* [50, 56-57], which in this case is the streamline separating flow in the zone of separation from that which passes clear through the channel. The separatrix thus blocks passage of bulk advecting fluid (along with the scalar species) into the recirculation zone, and vice versa; the ability to effectively mix different regions of fluid is severely limited unless some mechanism for disrupting the manifold (i.e. the separatrix) between the recirculating zone and the core flow is put into action. This is provided by pulsatility (intermittent gastric outflow) as shown in the next section.

2.4.2 Pulsatile Flow across the Gastric Outlet through a Relaxed Pylorus

Pulsatile inflow conditions were imposed next (Figure 2-3) on a channel that was identical to the one evaluated in the steady flow case. Examining Figure 2-7, pulsatility in the flow has apparently increased mixing a great deal over the steady flow solution, and in a relatively small spatial region. Streamlines illustrate a pattern of dynamic vortex behavior that was absent in the previous case. In the pulsatile flow, jet development through the pylorus is coincident with vortex growth and strengthening in the distal recirculation zones; the marked fluid is entrained by vortices as they gain momentum and push the separatrix further into the jet region. As the inlet flow decelerates, mean flow through the channel decelerates as well. This leads to "shedding" of the vortices, i.e. the vortices detach from the wall distal to the pylorus and are carried away downstream, and a new pair of vortices forms at the next flow pulse, leading to a periodic vortical flow field in the superior duodenum. Vortex formation and shedding from the divergent distal aspect of the pylorus provides a key mechanism for drawing the scalar species into the sheet shown in the figure; the separatrix between regions of large-scale advection and recirculation has been disrupted during the deceleration phase of the cycle, allowing the marked species to become entrained in the vortical flow.

2.4.3 Pulsatile Flow across the Gastric Outlet through a Closing Pylorus

Transport of gastric effluent induced by periodic antral contractions occurs in concert with the closure of pyloric orifice such that fluid is forced through a narrowing lumen. To isolate the contributions made by this narrowing to the mixing process, a model was constructed identically to that described in the previous section, but with the inclusion of pyloric closure. In this model, temporal motion of the solid boundary defining the pyloric geometry was varied quasi-sinusoidally such that the pylorus was in its fully open position at the beginning of a pulse cycle, and closed by the time the inlet velocity returned to zero at the end of the active part of the cycle. Temporally sinusoidal reopening of the pylorus was set to take place during the 1.0 time unit refractory period marking the end of the cycle.

Enhanced mixing clearly is achieved by closure of the pyloric lumen in concert with flow pulsatility (Figure 2-8). The jet flow effected by the narrowing lumen gives locally advecting fluid a great deal of axial momentum, while the recirculation zones grow concurrently with the increasing obstruction. This effectively leads to large regions of vorticity, which act to stretch marked fluid along the domain's axis in addition to rolling it into sheets within each vortex. Prominent secondary vortices are also observed in this case, possessing a rotational sense opposite to the primary vortices. This leads to further stretching and folding of the scalar field.

2.4.4 Pulsatile Flow across the Gastric Outlet through a Static Asymmetric Pylorus

In the previous three sections, three ingredients were examined that concertedly lead to increases in the degree of fluid mixing seen in a channel flow; namely, some sort of partial obstruction leading to convergence and subsequent divergence of the flow, pulsatility, and wall motion. So far, however, only highly simplified, symmetric geometries have been covered to make this point. In reality, with increased tonicity, the positioning of the proximal and distal pyloric muscle loops lead to a "notched" pyloric configuration when viewed in two-dimensional longitudinal sections (Figure 2-9 and Figure 2-10). The next case is designed to capture the effect of this geometric feature of the pylorus on flow into the proximal duodenum. Figure 2-11 illustrates the static asymmetric pylorus increasing mixedness of the scalar field over its symmetric counterpart. As fluid is accelerated through the pylorus in this configuration, the resultant efflux is directed away from the centerline due to geometric asymmetry. As the jet evolves downstream and diverges further from the axial midline, it collides with the upper wall of the channel and is subsequently deflected away toward the lower wall. This results in a meandering pathway of bulk advection, demarcated by vortices which are now staggered in the channel and free to occupy the whole of the channel width rather than being mirrored symmetrically. The resulting path of the fluid jet (and hence of the advected scalar) is highly tortuous, leading to larger residence times and stretching of fluid elements, and therefore to enhanced mixing.

2.4.5 Pulsatile Flow across the Gastric Outlet through a Closing Asymmetric Pylorus

Asymmetry is seen to be a significant contributor to mixing effectiveness. Next, all of the mechanisms (pulsatility, asymmetry, pyloric closure) examined heretofore are involved by examining the effect of closure of an asymmetric pylorus.

Immediately evident is the fact that the mechanisms of pulsatility, asymmetry and pyloric closure concertedly lead to significant enhancement of mixing. After a single gastric pulse, the scalar marker species has become much more nearly homogenized than in any of the previous cases (Figure 2-12). It is noteworthy that this homogenization not only occurs rapidly, but takes place within a small spatial region as well; scaling with the human GI system while maintaining the same Reynolds number, the level of mixing seen in this case would occur within a 4 or 5 cm segment of the proximal duodenum [5] — even neglecting the presence of prominent geometric features such as the duodenal cap and mucosal folds in the channel.

2.4.6 Mixing Effectiveness attributed to Geometric and Dynamic Properties

Commonly employed scalar variance measures were obtained during each flow calculation as a method of quantifying scalar mixing. For each of the cases computed, normalized scalar variance (Equation 4) was plotted for one complete pulse cycle to directly compare each flow's mixing effectiveness. In Figure 2-13, the scalar variance measure of each of the 5 cases is plotted. Dynamic behavior of the pylorus, combined with pulsatility, clearly leads to effective mixing. Imperfections in the form of channel asymmetry enhance this phenomenon even more; the physical structure of the pylorus appears to have a significant effect on mixing patterns, at least in the Reynolds number (i.e. the fluid viscosity) regime corresponding to saline in the cat gut examined in vitro [48].

2.4.7 Mixing Effectiveness attributed to Reynolds Number

The effective viscosity of chyme can vary considerably depending on its solid fraction [5, 45-46, 48], with more viscous solutions yielding lower Reynolds numbers. Thus, the Reynolds number was varied over several orders of magnitude to study the effects of viscosity on mixing. For brevity, results are limited to Re variations in the fifth and final case involving a closing, asymmetric pylorus. As seen from the scalar variance plot in Figure 2-15 the flows with Reynolds numbers of order 1 and order 10 lack the momentum necessary to overcome viscous stresses in the fluid and generate vorticity, leaving the initial scalar species blocks largely intact as they progress through the channel. Advection begins to dominate as chyme viscosity decreases, resulting in a large degree of fluid stretching, vortex formation and hence, mixing. Scalar variance plots help to quantify this shift in behavior, and leave us to conclude that pyloric structure may not play a large role in the homogenization of dense slurries without adequate dilution.

2.5 Conclusions

The lumen of the gastroduodenal junction has a complex geometry which changes with the contractile activity of gastroduodenal musculature in response to the pH, osmolarity, caloric density and mechanical properties of the luminal contents [5, 45-46]. The focus here was on the contribution likely to be made by several specific anatomical parameters (like the manifold structure of the pyloric lumen produced by its muscle loops), and by some specific functional parameters (steady versus intermittent gastric outflow). A comparison of results between aqueous and more viscous luminal contents was made, as well. The results indicate that intermittent gastric outflow in combination with the complex geometry and motion of the pyloric lumen is likely to enhance duodenal mixing of aqueous fluids, facilitating rapid chemical digestion and subsequent absorption of nutrients in the duodenum. More viscous fluids or slurries may remain unmixed for a longer period of time, and will necessarily involve contractions of the duodenum to provide any significant homogenization.

For this study, no image data or other means for accurately reproducing pyloric morphology were available; the bases for modeling the pylorus were simply a qualitative description of its approximate geometry and motion, and knowledge of the relationship between pyloric luminal diameter and the rate of chyme being discharged from the stomach [48]. Thus, the complexity of the models was necessarily limited. Austere geometric representations such as these can lend general physical insight, as has been shown, but capturing intricate motions and shapes of the type found in biological organisms with specificity requires a more detailed description; improving representative fidelity will be a dominant focus for the remainder of this work.

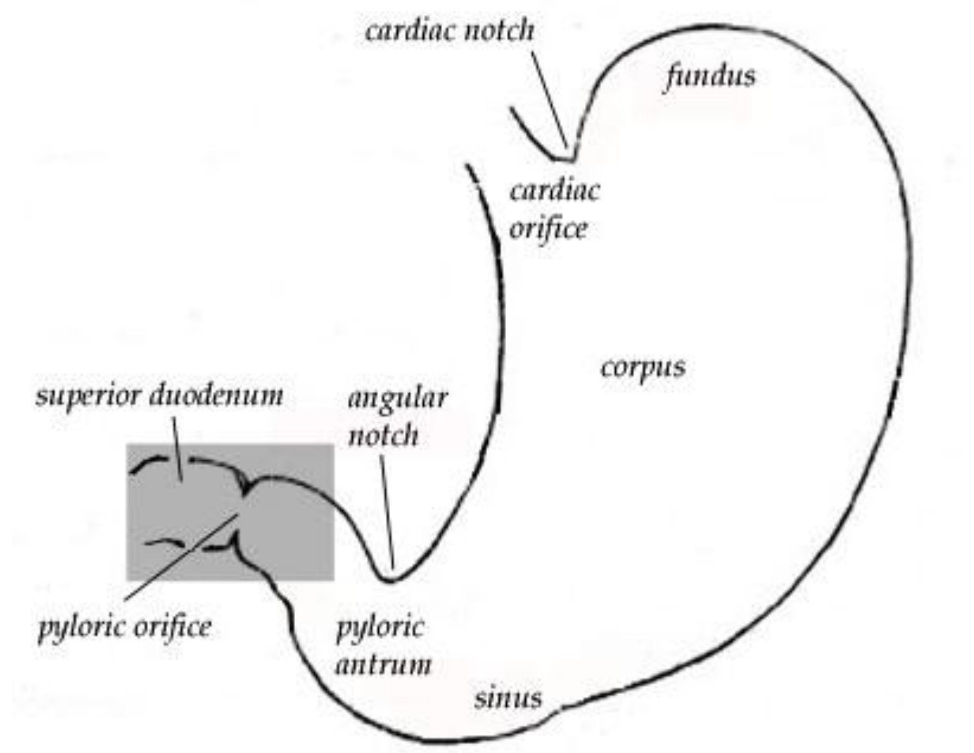
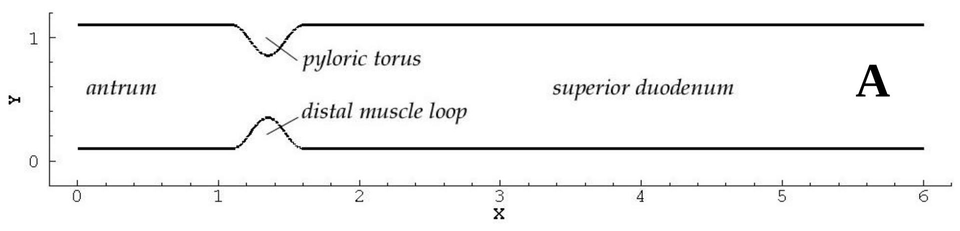


Figure 2-1. The region of interest (highlighted in grey) includes the gastric outlet, or antrum, pylorus, and superior duodenum.



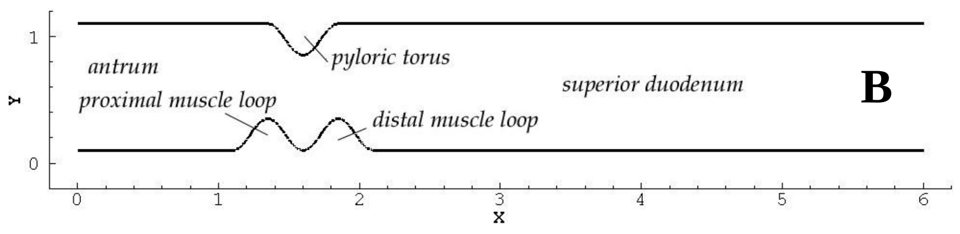


Figure 2-2. Two representative channel domains used in flow calculations: (A) relaxed pylorus; (B) "notched" configuration resulting from tonicity of the pyloric torus and both muscle loops.

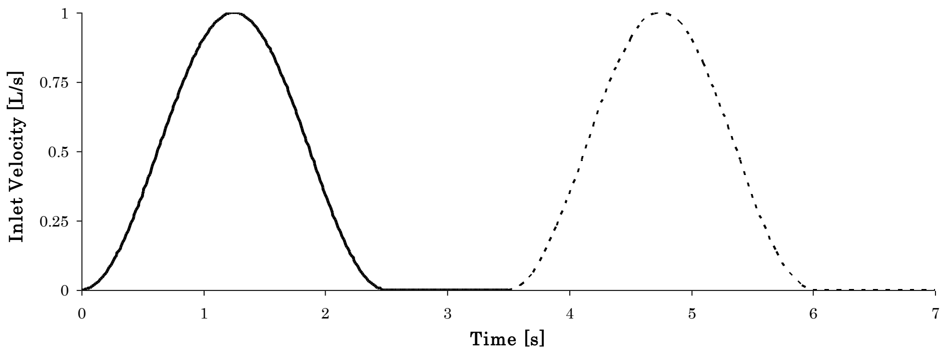


Figure 2-3. Temporal variation of inlet velocity.

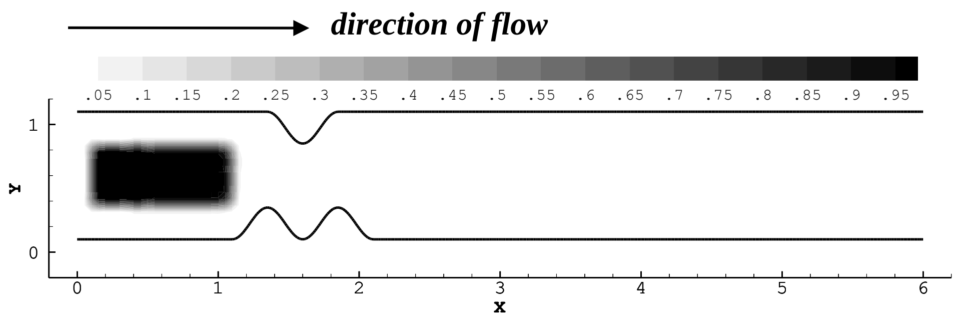


Figure 2-4. The scalar species block was initialized identically for each case evaluated.

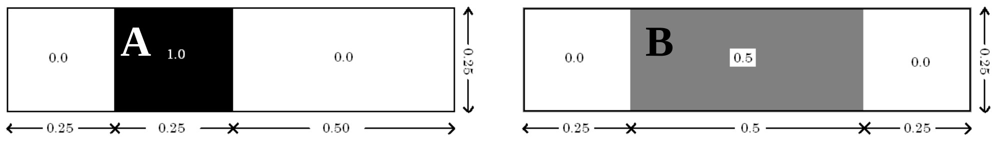


Figure 2-5. Illustrative example of scalar variance calculation: (a) initial and (b) final states shown, each with a mean concentration of 0.25.

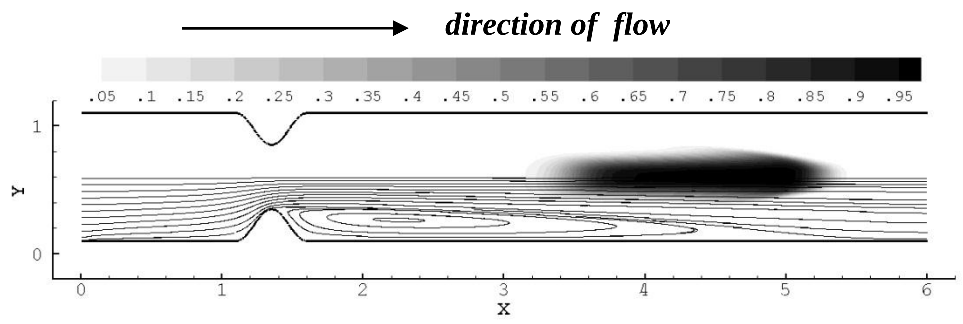


Figure 2-6. Passage of a passive scalar marker through a relaxed pylorus under steady flow conditions, with inlet velocity set to $U^*_{inlet} = 1.0$ and Re = 333. In this image, 2.0 time units have passed, with the marked fluid exhibiting vertical compression toward and horizontal stretching along the symmetry axis in the weak jet created by the pyloric constriction. Aside from an increase in surface area due to longitudinal stretching, no mixing is seen in this case.

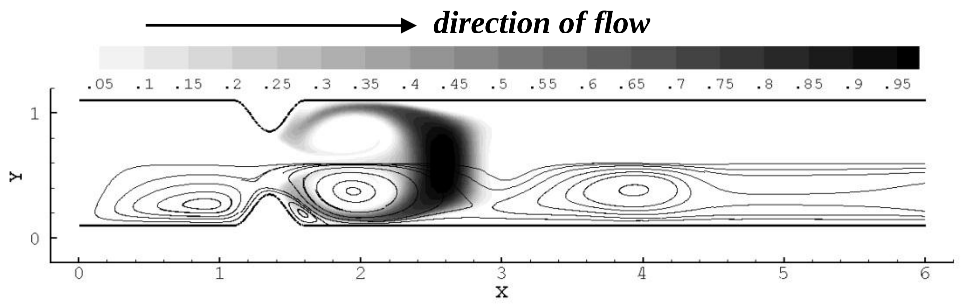


Figure 2-7. Scalar species distribution following the active phase of one inlet pulse (t = 2.5 time units, Re = 333), illustrating vortical stretching and resultant increase in interfacial area between regions of concentration 1.0 and concentration 0.0.

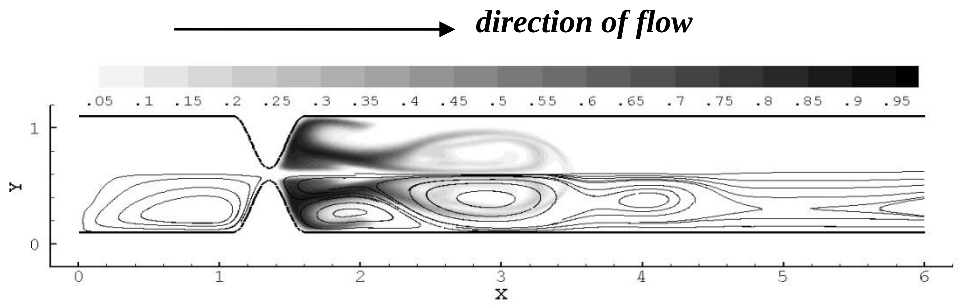


Figure 2-8. Scalar species passage through a narrowing pylorus (Re = 333): t = 2.5 time units; the pyloric lumen is occluded and inlet velocity has returned to 0.0. The thin, high velocity jet effected by the narrow lumen has led to rapid growth of a strong vortex pair, which sheds and propagates downstream as the bulk flow is halted.

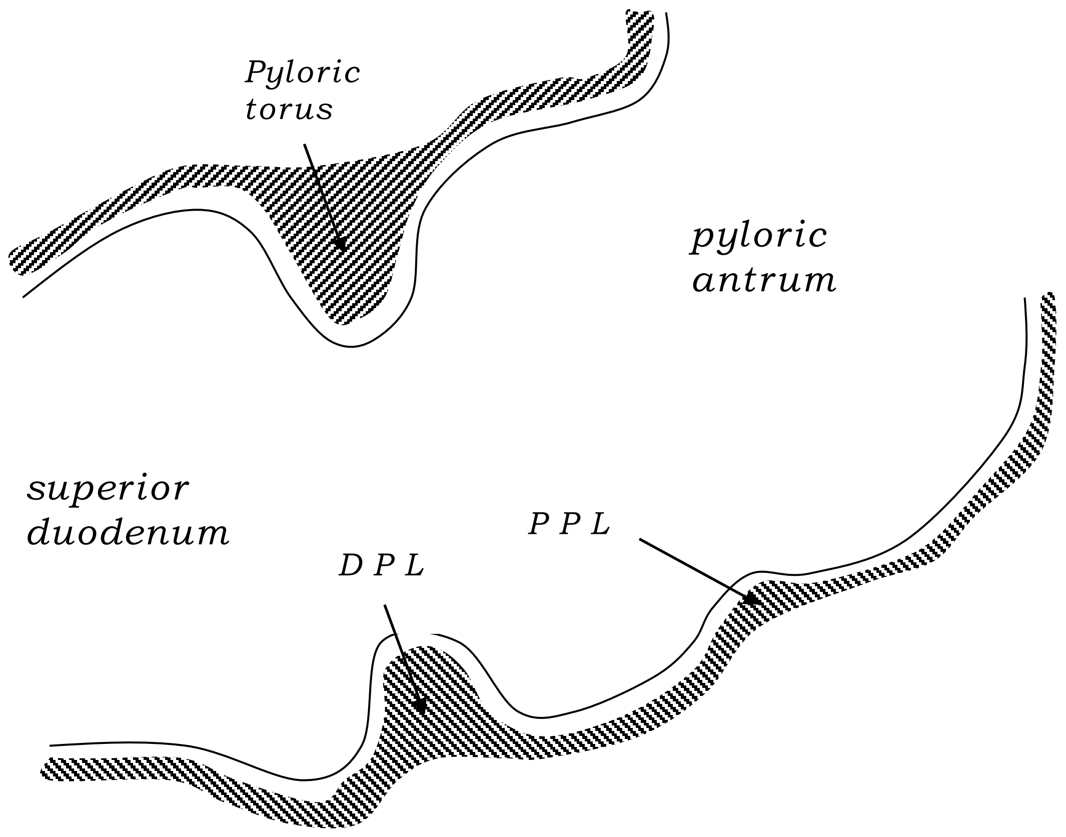


Figure 2-9. Schematic of the relaxed pylorus.

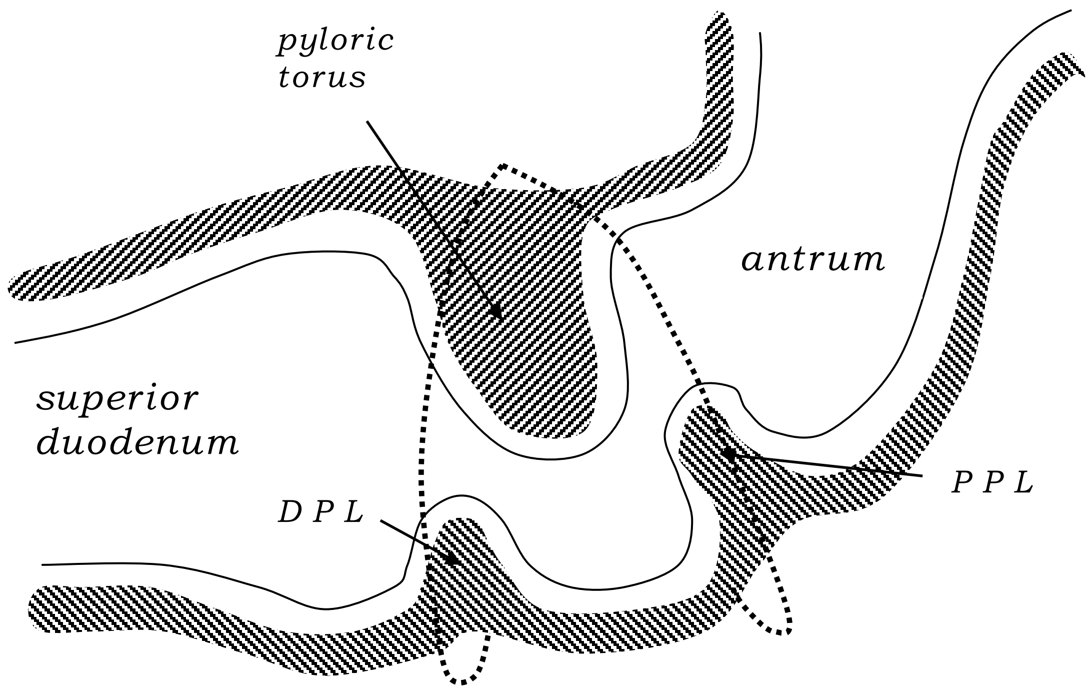


Figure 2-10. 2-D asymmetry is produced by tonicity of both pyloric muscle loops in the contracted state.

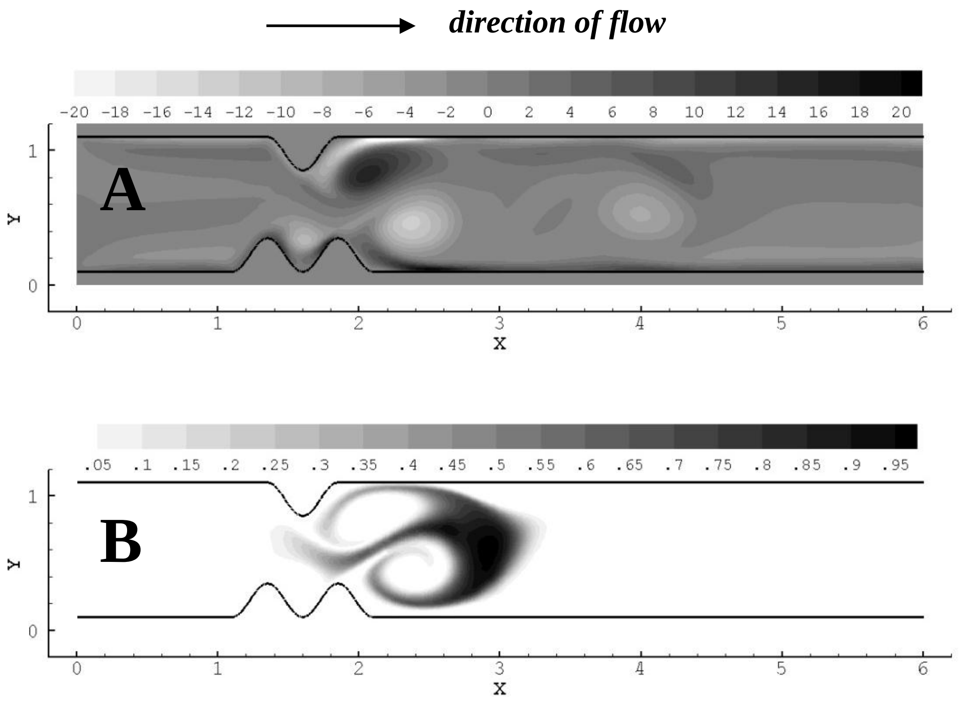


Figure 2-11. Vorticity (A) and scalar field (B) after 2.5 time units at Re = 333. Inlet velocity has returned to 0.0 and vortices have shed from divergent surfaces distal to the lumen.

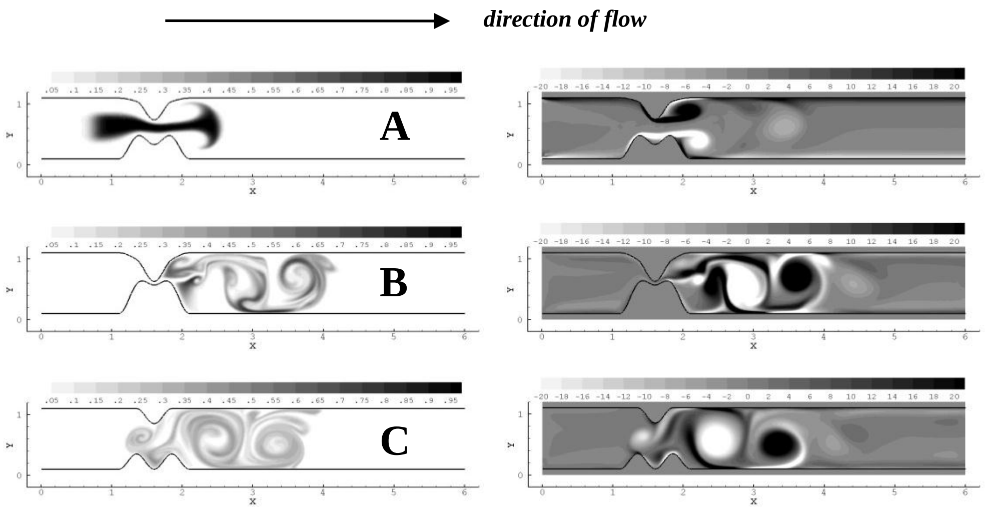


Figure 2-12. A temporal progression of scalar species (left) and vorticity (right) during one complete inlet pulse cycle (3.5 time units) through the closing asymmetric pylorus: (A) t = 1.25 time units, with the vortical jet becoming apparent as inlet velocity is maximized and luminal narrowing occurs; (B) t = 2.5 time units. The lumen is fully closed, and the inlet velocity has returned to 0.0; (C) t = 3.5 time units. The pylorus has returned to its original open state during the 1.0 time unit refractory period. Strong, stable regions of vorticity remain, entraining weaker unstable vortices and further stretching the species.

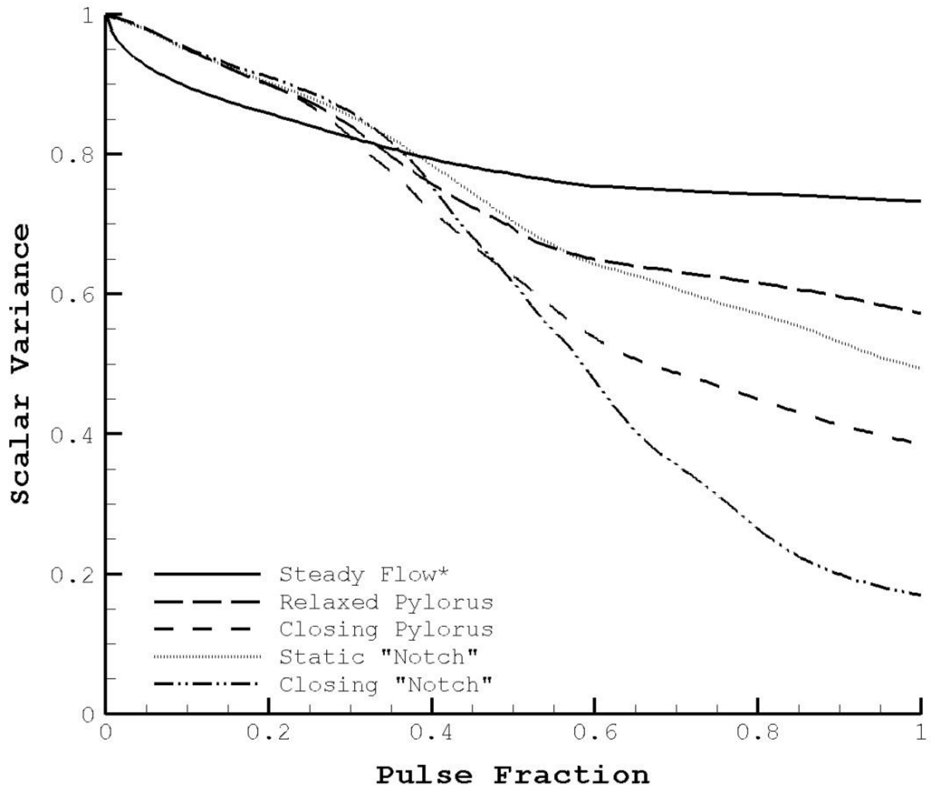


Figure 2-13. Scalar variance plots of each case through one complete pulse cycle; Re = 333.

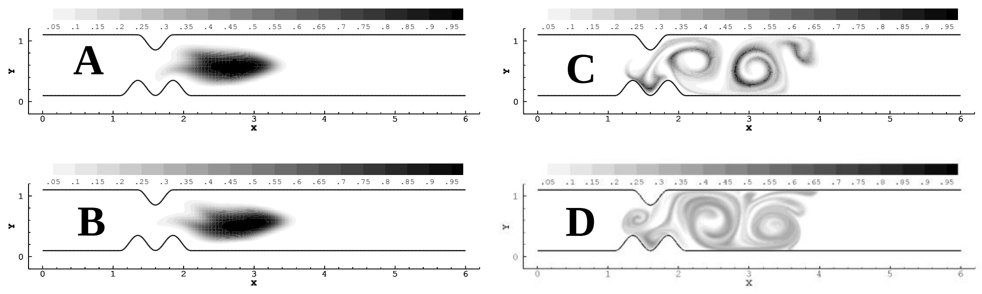


Figure 2-14. Scalar species plots following one pulse cycle through the closing, asymmetric pylorus at four different Reynolds numbers: (A) Re = 1; (B) Re = 10; (C) Re = 100; (D) Re = 333.

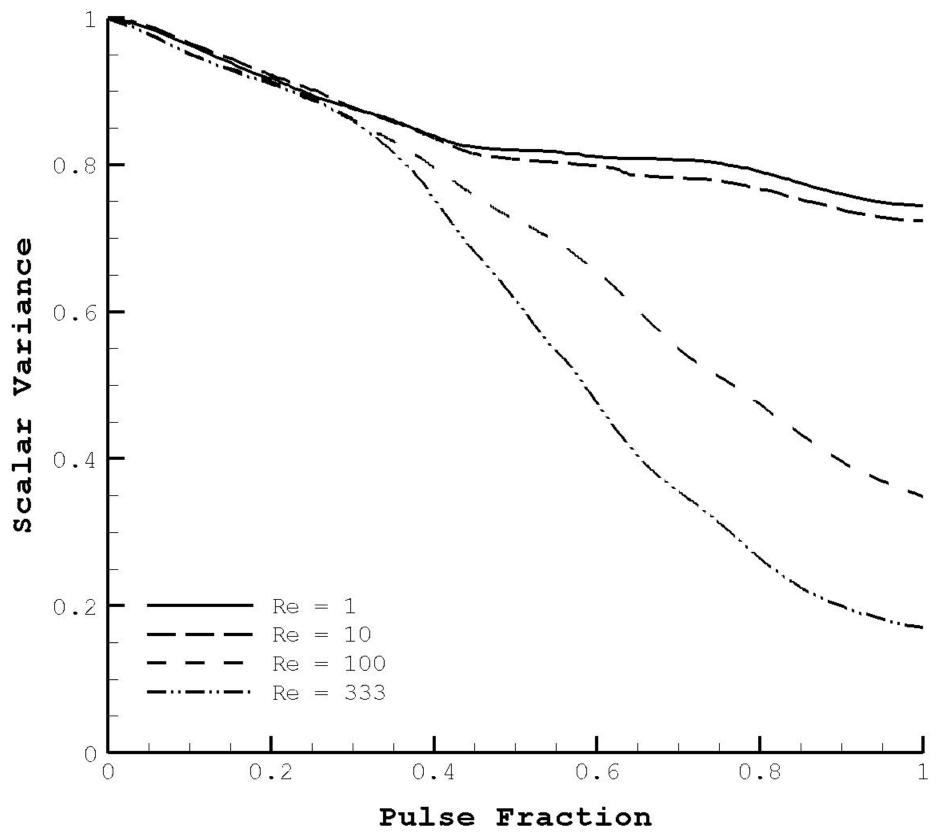


Figure 2-15. Scalar variance of the asymmetric pylorus model at four different Reynolds numbers.

CHAPTER 3
IMAGE BASED MODELING IN TWO DIMENSIONS: PHYSICAL ANALYSIS OF LOCOMOTION AND TRANSPORT WITHOUT IDEALIZED GEOMETRIES

3.1 Introduction

In chapter 2, a set of idealized geometric models of the pylorus was introduced, and it was shown that such representations can be instructive for the purpose of elucidating physical trends in different types of computed flow fields. However, for more complex shapes interacting with fluids, particularly those exhibiting motion that is difficult to describe functionally, idealized geometries evolving in prescribed fashions may fail to reveal some of the essential physical mechanisms that make studying the system interesting to begin with. Alternately, generating surfaces and effecting their temporal evolution with fidelity may introduce a level of tedium that produces a strong desire for a simpler way of doing things. In this chapter, a first attempt at an image-based approach designed to obviate some of these difficulties inherent to modeling complex phenomena such as animal locomotion and fluid transport through organ systems is developed, in which image files are used directly as the geometric descriptions upon which to build computational surfaces for CFD simulations.

Two different systems were chosen for developing the methodology herein, for reasons of biological interest as well as amenability to image segmentation. The first is an example of animal locomotion through an aqueous environment, modeled using a video file of an American eel (*Anguilla rostrata*) swimming in an experimental water tunnel apparatus. The video was provided by Dr. Eric Tytell, and was originally created to supplement experimental PIV results published by Tytell and Lauder [58] (Figure 3-1). The second system involves internal fluid transport through a guinea pig (*Cavia porcellus*) duodenum in vitro, the motion of which was captured on video during an experiment set up by Dr. Konrad Schulze at the University of Iowa Health Care center's department of internal medicine (Figure 3-1). Each video was segmented frame-by-frame, and the resulting segmented surfaces were converted to sets of Lagrangian points, mapped onto a Cartesian mesh as zero level set contours, and imposed as boundaries embedded in their respective Eulerian flow fields.

3.2 Image Segmentation

In the introductory chapter, it was mentioned that the more complex an image is according to our visual system, the more advanced a segmentation algorithm must be to adequately separate the scene into distinct objects. Fortunately, the image files chosen for this development project are particularly amenable to simple segmentation techniques in that there are only two primary regions in each picture – the object of interest and its surroundings – and that the two regions are highly contrasted. The eel is nearly black in color (low in intensity), while the background is much lighter (higher intensity); the duodenal images reverse the color scheme but maintain the disparate intensity values between object and background. This allowed for early development of image-based modeling ideas to proceed without too much emphasis yet needed in the way of image quality. As such, the rapid k-means level set segmentation approach outlined by Gibou and Fedkiw [18] was chosen for its simplicity in implementation as a means to produce an initial survey of the overall viability of generating models from images.

The k-means approach for segmenting an image entails initializing the image's pixel (x) intensity field domain Ω with an arbitrary closed curve C, so that $\Omega = \{\boldsymbol{x} | \boldsymbol{x} \in \Omega_1 \cup \Omega_2\}$, with C enclosing some arbitrary region Ω_1 of pixels possessing an average intensity level of I^-. The remaining pixels in the image lie in region Ω_2, outside the closed curve where the average pixel intensity is I^+ (Figure 3-2). Curve C may be regarded as an isosurface (zero level set) of level set field φ, which is iteratively evolved according to pixel values falling inside and outside of the curve:

$$\frac{d\varphi}{dt} = -\lambda_1(I_0 - I^-)^2 + \lambda_2(I_0 - I^+)^2 \quad \textbf{(3.1)}$$

In Equation 3.1, λ_1 and λ_2 are weighting coefficients (which are normally tuned so that curve evolution is biased toward either the inner or outer region depending on the final segmentation aims, but are both set to unity here due to the high level of image contrast exhibited in both cases), and I_o is the value of a single pixel in the image. Each pixel x, then, has its own associated level set velocity $\frac{d\varphi}{dt}$ (which will henceforth be denoted $\boldsymbol{v}$). The idea is to minimize $\boldsymbol{v}$ throughout the image domain, at which point segmentation is achieved. Gibou and Fedkiw note that image segmentation is not concerned with the nature of a curve's evolution, but only with the final result [18], and therefore reason that large time steps may be taken in reaching a final solution to Equation 3.1. They further argue that this may be equivalently accomplished by regarding a segmented image as a binary field, with a single (negative) value given to all pixels "inside" the curve describing an object, and another (positive) value given to all pixels "outside." Thus, for segmentation calculations performed in this work, each pixel is given a segmentation field magnitude of unity, with a sign equal to that of $\boldsymbol{v}$ at the corresponding pixel location.

Starting with an arbitrary initial curve such as that illustrated in Figure 3-2, curve C is iteratively evolved until $\Delta \boldsymbol{v}$ vanishes, typically taking 4 or 5 iterations (Figure 3-2). The result is a binary field of value ± 1 closely describing the shape of the object being segmented (Figure 3-3 and Figure 3-4), albeit in an unsmooth, pixel-wise manner. This process is repeated for each image frame in the video file, with the exception of the arbitrary initialization step; after the first frame, each subsequent frame uses the already existing segmentation field from the frame before it as an initial state (or "shape prior") in order to speed up calculations.

The video files used in this investigation, and perhaps most meso-scale video files in existence, are spatially and temporally too coarse to be used "as is" for CFD calculations; computational fluid simulations require higher fidelity than what is given by the pixel density and frame capture rate of the average video camera. For this reason, segmented objects must be mapped onto flow meshes containing more grid points than there are image pixels, and intermediate geometric configurations must be reconstructed between image frames to avoid jumps in position that are incompatible with fluid response length and time scales.

3.3 Preprocessing

Matlab 2007A was used to convert the supplied AVI video files of the eel and the duodenal segment into raw data files, and was concurrently used to crop the duodenum video frames in order to isolate the region of interest during the process (Figure 3-5). The coarse nature of each of the image files serving as model bases required some preprocessing before segmentation could proceed. In regions of large gradient between each of the objects (the eel and the duodenum) and their surroundings, there existed a quantized “checkerboard” pattern where different intensity values are meant to blend with each other and create the soft outline generated when a rounded object reflects light in a smooth, continuous fashion (Figure 3-6). This becomes problematic for segmentation techniques that rely on relative intensity values of neighboring pixels as a means of detecting objects. In extreme cases, a checkerboard pattern featuring alternating light and dark grey values that are close in intensity levels to the object of interest and its surroundings, respectively, may be segmented into a number of different regions that contain only a few pixels.

To avoid generation of such tortuous segmented surfaces, each of the image frames used in this work was preprocessed with wavelet filtering techniques in order to smooth boundaries and provide a clean segment. A 2-D discrete wavelet transform (DWT) algorithm written in Fortran 90 roughly following [59-60] was utilized to map each image frame into wavelet space, where relative pixel intensity values could be compared within neighborhoods of varying size. Described in some detail in Chapter 1, the DWT decomposes an image pixel intensity field of dimension 2^N by 2^N (or an image that has been expanded to those dimensions by padding with extra pixels), where N is an integer value, into a set of 2^{N-1} wavelet coefficients and 2^{N-1} scaling coefficients which describe the image in terms of a wavelet function. This operation is performed again on the resultant scaling coefficients, giving a new set of 2^{N-2} wavelet coefficients and 2^{N-2} scaling coefficients, in addition to the 2^{N-1} wavelet coefficients left over from the previous step. The operation continues recursively until the image is comprised of 2 scaling coefficients and $2^{2N} - 2$ wavelet coefficients organized in an octree structure, and can be reduced no further (Figure 3-7 and Figure 3-8.) [21, 23, 61-63].

Wavelets can be used to filter and smooth images by thresholding in several different ways. One method involves removing wavelet coefficients at different scales, generally from the smallest scale to the largest scale. Undesirable noise in an image signal is usually a small-scale phenomenon, so scale-based filtering can be an effective way to remove it. Other methods for image filtering regard the wavelet coefficients as a measure of energy, and remove coefficients that are either below a predetermined energy threshold themselves, or remove a group of coefficients whose sum is less than some percentage of the total energy in the transformed image [61, 63]. These thresholding operations can also be combined, so that coefficient clipping only occurs at certain levels in wavelet space. Figure 3-9, Figure 3-10, Figure 3-11, and Figure 3-12 illustrate the results of these filtering techniques using four different wavelet shapes—a Haar wavelet (a step function with 2 wavelet coefficients that adheres to the same conditions as Daubechies wavelets), and three Daubechies wavelets (the continuous function having a minimum of 4 wavelet coefficients as described in Chapter 1; 4, 12, and 20 coefficients were tested for this work). After testing a range of scale and energy filtering levels for each of these wavelets, it was found that the most reliable segmentation results were achieved using a Daubechies-20 DWT with 0.001% of the total wavelet energy removed. Thus, 99.99% of the original image information was retained, giving a high level of fidelity, with just enough energy removed to smooth out some of the especially noisy "trouble" regions. (This also testifies to the ideal nature of the selected test images, as little preprocessing was required to make them useful.)

Even with wavelet smoothing performed on the images, and a high level of contrast overall, it was found that the low pixel resolution of the eel images, combined with the narrowness of the animal’s tail, made segmentation of that region particularly difficult (Figure 3-13). The segmentation results produced in this region had jumps in them from one frame to another, particularly in the eel’s axial direction (Figure 3-14), so it was necessary to truncate the eel’s body at $x = 245$ to prevent this unphysical motion from carrying over into the CFD model.

3.4 Computational surface meshing

Reconstruction of an object’s morphology through arbitrary time step sizes Δt requires knowledge of its surface velocity between image frames (which have a known image, or *fiducial*, time step size ΔT). The binary field generated by the k-means segmentation approach simply provides a scalar value for each pixel in the image domain; one more constraint would be required for the construction of a 2-D velocity field. Thus, acquisition of surface velocity was accomplished here by converting the binary fields produced by segmenting the eel and duodenum from their respective image frames into Lagrangian surface points describing the objects of interest, so that individual surface points could be tracked spatially and temporally, and used to supply boundary conditions to the surrounding flow field during CFD simulations. This was performed in several steps:

1. Interface location search- Initially, an algorithm was written with the intent of being able to locate segmented interfaces in a general fashion, regardless of geometry. The segmentation field value, X and Y locations, and information

regarding whether a pixel was adjacent to another pixel possessing a different field value from itself (i.e. an interfacial pixel) were all stored in a data structure for each pixel in the image domain. Starting with the first pixel at the lower left corner of the image, rows of pixels were swept until one was found possessing a different field value from those swept before it (Figure 3-15). Once this first object pixel was detected, each of its 4 neighbors was probed in an anti-clockwise direction (in the order east, north, west, and south) to find whether it was (a) an interfacial pixel and (b) a pixel with a different field value from the current one (Figure 3-16). Whenever both of these conditions were met, a point was constructed on the interface and its address in the domain stored. Once probing was finished, the pixel was tagged as "complete." The next pixel to be visited by the algorithm was either the last one found possessing a different field value from the current pixel, or the last interfacial cell found that was not marked "complete." In this way, the search algorithm would march back and forth across the interface, storing point locations along the way (Figure 3-17).

However, it was quickly discovered that this general searching and point placement algorithm was not satisfactorily robust, and could be made to fail relatively easily if a segmented field contained protrusions that were 1 or 2 pixels in width. An example of this type of situation is illustrated in Figure 3-18 and Figure 3-19.

Due to the fact that both the eel and duodenum image data sets featured slender bodies oriented horizontally with respect to their dominant length, and following failure of the original surface point placement algorithm, it was decided to instead sweep each segmented image frame from bottom to top along pixel columns, one column at a time from left to right (Figure 3-20), to determine surface locations. A surface was regarded as existing between two segmentation field values of opposite sign, i.e. a change of value from -1 to +1 requires crossing an interface:

$$\mathbf{sign}(\varphi_j \cdot \varphi_{j+1}) = \begin{cases} -, if\ interface\ present \\ +, no\ interface\ present \end{cases}. \qquad \mathbf{(3.2)}$$

Whenever a negative value occurred, the Y-location $\left(j + \frac{1}{2}\right)$ of the corresponding surface was stored, giving 2 Y-locations per segmentation field column possessing object pixels.

2. Centerline generation- For the eel, the two stored Y-locations in each column were averaged to give a centerline location (Figure 3-21):

$$Y_{centerline} = \frac{Y_{upper\ surface} + Y_{lower\ surface}}{2}. \qquad \mathbf{(3.3)}$$

Then, the Y-locations were updated to describe their positions above and below the centerline, i.e.

$$Y^+ = Y_{upper\ surface} - Y_{centerline}, \qquad \mathbf{(3.4a)}$$

$$Y^- = Y_{centerline} - Y_{lower\ surface}. \qquad \mathbf{(3.4b)}$$

This operation was performed in order to ensure that the eel's body would maintain symmetry during subsequent smoothing operations, i.e. the upper and lower surface contours could not cross each other in an unphysical manner or develop any gross displacements to one side or the other. The duodenum did not require this step, being an asymmetric channel with two distinct surfaces that do not require matching for surface closure.

3. Centerline and duodenal wall smoothing- The centerline points of the eel and each channel wall of the duodenum were smoothed using a Savitzky-Golay filter of window size 30 (Figure 3-22); the upper and lower surface locations of the eel were updated with respect to its smooth centerline locations.
4. Lagrangian point placement- Lagrangian points were defined on each surface at the interface crossing locations stored earlier between segmented pixels. Each point was made part of an array structure containing information about each point's X- and Y-locations, and velocity components.

   All aspects of the flow solution process take place within an Eulerian framework on a Cartesian mesh, so the natural question arises: Why is there a need for Lagrangian point placement and tracking if this is the case? In the present context, Lagrangian information is used solely for the purpose of supplying embedded interfaces with velocity information, which is in turn extended into the neighboring flow field to generate the corresponding physical fluid response. However, work aimed at obviating these steps has since been completed, and shall be the primary focus of Chapters 5 and 6.
5. Secondary surface smoothing- The eel's surface points were smoothed using a 1-D discrete wavelet transform algorithm written as part of this thesis work and based on [59-60]. The periodic nature of wavelets makes them ideal for smoothing a closed surface such as that of the eel, which must begin and end in the same place when circumscribed just as a periodic signal does. A Daubechies 20 wavelet was chosen for its favorable de-noising capabilities (its larger support

gives a smoother signal representation than smaller wavelets, which are more apt to sharply pick out discontinuities); 4 wavenumbers were removed for smoothing.

6. Point spacing- One of the drawbacks to smoothing closed surface points in a manner analogous to 1-D signal de-noising is the fact that point spacing is not necessarily maintained; the points are treated as being equally spaced according to the DWT algorithm even though they are clearly not, based upon the searching and placement method used to produce them. Points were originally placed on the eel in an anti-clockwise direction, with the first and last points lying collocated on the eel's nose, or snout (furthest location to the left in the figures). The first point location was preserved by way of its role as a reference location during smoothing operations, however, it was found that a gap was produced between the first and last points after smoothing was performed, affecting closure of the surface. This was remedied by redistributing the points along the eel's surface so that they became evenly spaced with uniform segment lengths between them.

    To accomplish this, the total surface arc length was computed and divided into uniform segment lengths based upon the number of points on the surface. Starting with the first fixed point on the eel's nose, an iterative placement routine was written to compute the distance between one point x_i and its next neighbor x_{i+1}, traveling around the surface in an anti-clockwise direction. If the distance was found to be greater than the target uniform segment length, the difference was subtracted from the point spacing and point x_{i+1} shifted closer to x_i along the vector between them. Otherwise, the difference was added, shifting point x_{i+1}

further from point x_i along the vector between x_{i+1} and x_{i+2}. The process was repeated along the entire surface until the first fixed point at the eel's snout was reached once again, completing the operation and closing the surface.

7. Point population- Evenly spaced Lagrangian points were made denser by way of linearly populating intervals between them with more points in order to ensure that an adequate surface description would be interpolated onto the Cartesian mesh during flow computations. The number of points placed on each interval was thus decided by the ratio of interval length Δs to minimum grid spacing Δx_{min} on the flow mesh:

$$N_{interval\ points} = \mathrm{int}\left(\frac{\Delta s}{\Delta x_{min}}\right). \qquad \textbf{(3.5)}$$

8. Scaling and shifting- Surface points in each case were scaled and shifted to produce physically meaningful length scales and for convenient positioning within the computational flow domain, rather than being relegated to operating on dimensions relating to pixel count. Figure 3-23 shows the final set of points generated using this algorithm, for the first image frame of the swimming eel video. The points are overlaid on a scaled and shifted version of the original image for illustration.

9. Surface velocity calculation- In order to compute surface velocities, a three-frame strategy was utilized to avoid step-wise jumps in (constant) interfacial velocity magnitudes between fiducial time steps ΔT. Upon passing a time value $T_0 = n\Delta T; n = 1,2,3 \ldots$ where the position corresponds to a new image frame, surface velocities were computed at two fiducial time intervals — the updated current image time, and the subsequent image time. This was performed using interfacial

position data from the images at times T_0, $T_0 + \Delta T$, and $T_0 + 2\Delta T$, to form velocities as

$$\boldsymbol{v}_1 = \frac{\boldsymbol{x}_2 - \boldsymbol{x}_1}{\Delta T} \qquad \textbf{(3.6a)}$$

and

$$\boldsymbol{v}_2 = \frac{\boldsymbol{x}_3 - \boldsymbol{x}_2}{\Delta T}. \qquad \textbf{(3.6b)}$$

Here, $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ are smoothed, scaled, and shifted Lagrangian surface coordinates obtained from the set of three images frames using steps 1 through 8 outlined above. The position of the interface is then updated for current time *t* as

$$\boldsymbol{x}^n = \boldsymbol{x}^{n-1} + (t - T_0)\boldsymbol{v}_1, \qquad \textbf{(3.7)}$$

and the velocity is updated by interpolating at the current time using $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. For all subsequent intermediate time steps between T_0 and $T_0 + \Delta T$, the position is updated using (3.7) with the velocity replaced by an updated interpolated velocity; the velocity is then re-interpolated and updated at the new current position. The end result is a linearly changing, rather than constant, velocity profile between any two fiducial time steps ΔT; acceleration is not continuous, but the results seem to be stable for the simulations run thus far.

Clearly, using the Lagrangian method to compute surface velocities required an equal number of surface points $\boldsymbol{x}$ to be placed on the modeled objects in each of the three frames, as one-to-one point correspondence over the entire surface is necessary for a complete description of velocities and positions through the fiducial time steps. This condition was enforced by segmenting and smoothing each of the three frames, and placing a first estimate of the surface points for each frame based on current total arc length calculated along the entire surface being modeled. The total number of surface points was then tallied on each frame, and the maximum of these was taken as the number of points to place on the surface for each of the three frames.

3.5 Level Sets: Sharp Interfaces Embedded in an Eulerian Flow Field

In this body of work, all CFD simulations are computed on a fixed Cartesian mesh, with local mesh refinement capabilities providing enhanced resolution of interesting physical flow features and regions of fluid interaction with immersed boundaries [53]. Interfaces are tracked using a level set field, φ, embedded in the computational flow domain, with zero level set isosurfaces representing fluid-solid interfaces.

The level set field represents a signed normal distance from an Eulerian grid point in the flow domain to the nearest zero level set contour. The field is constructed using what is known as the Fast Marching Method, a method which marches contours from a set of points with known field value (generally on the zero level set isosurface) outward by imposing the Eikonal condition [24].

$$|\nabla\varphi| = 1 \qquad \textbf{(3.8)}$$

The position and motion of a zero level set contour supplies the necessary boundary condition information for fluid motion around solid structures being modeled within the Eulerian flow field. Because level set field information is not necessary throughout the entire flow domain, but rather is only needed near solid boundaries that affect flow patterns, a narrow-band approach is used in which level set field information is stored in a small layer of grid cells adjacent to zero level set contours (of width $6\Delta x$ in each direction normal to the interface) in the interest of computational efficiency. The narrow-band level set field describing the eel's surface on its first image frame, generated using the smoothed Lagrangian points marking the segmentation contour as outlined above, is illustrated in Figure 3-24 and Figure 3-25.

3.6 External Flow: Locomotion of the American Eel through its Aqueous Environment

The first simulation case run for this development project was that of the swimming American eel, modeled using video footage supplied by Dr. Eric Tytell. The eel was filmed swimming in a water tunnel apparatus, for the purpose of better understanding the hydrodynamic consequences of its undulatory, or *anguilliform* swimming motion. The eel's dimensions and boundary conditions for the experimental setup are supplied in [58]. For the current investigation, 36 video frames containing one complete tail beat cycle of the eel's motion were selected for segmentation, and then copied 9 more times in sequence to produce a total of 360 video frames containing 10 identical tail beats.

In addition to serving as a demonstration of the image-based modeling techniques under development, the eel's anguilliform swimming motion holds biological interest by way of its prevalence as a mode of locomotion seen in a large number of species residing in aqueous environments. It was therefore decided to examine several different combinations of Reynolds number and Strouhal number regimes in order to study the effect on wake structures, thrust production, and other quantities of physical interest.

In Tytell's work, the eel being studied was measured to be 20 cm in length ($L = 20\ cm$), and was placed in a water tunnel in which it swam at a steady-state rate of $1.4\ L\ s^{-1}$, or $28\ cm\ s^{-1}$. Thus, the Reynolds number of the swimming eel, based on its body length, is roughly 56,000. The Strouhal number of the swimming eel ($\mathrm{St} = \frac{fA}{U}$, where f is vortex shedding frequency, A is the characteristic length of the problem—the peak-to-peak amplitude, or distance traversed by the eel's tail in this case—and U is the free-stream velocity of the fluid environment) was reported to be ~0.3, based upon tail beat amplitude of approximately 7% of the eel's body length ($\sim 0.07\ L$) and a frequency f of 3.1 Hz. The DNS nature of the flow solver used for these simulations limited cases to Reynolds numbers that were more than an order of magnitude lower than what was produced by the swimming eel in the experimental setup; Reynolds numbers of 2500 and 5000 were studied here. Three different Strouhal numbers, St = 0.3, 0.5, and 0.7, were simulated for each Reynolds number regime and examined for variations in wake structure and thrust magnitude.

For each of the simulations run, the model eel was initialized in a domain of dimension 5 units length by 2 units height, with a coarse grid spacing of $\Delta x = 0.025$ and three levels of mesh refinement, giving a minimum grid spacing of $\Delta x_{min} = 3.125 \times 10^{-3}$ (Figure 3-26, Figure 3-27, and Figure 3-28). Open flow conditions were simulated by initializing the velocity field with a horizontal (u-velocity) magnitude of 1.0, imposing inlet conditions of $u = 1.0$ on the west side of the domain, specifying the east as an outlet, and setting Dirichlet conditions with u-velocity of 1.0 on the north and south boundaries. In each case, the eel's position was fixed for 1000 time steps in order to allow for flow field maturation before imposing swimming motion. Fluid density was specified as $\rho = 1.0$, so that the Reynolds number could be varied simply by changing the dynamic viscosity value μ. Strouhal number variation was accomplished by specifying the fiducial time step size ΔT between image frames.

Results of the 6 different cases simulated are encouraging in that they appear to share some of the physical trends revealed on review of Tytell's experimental observations, particularly at the lower Strouhal numbers (keeping in mind that we are still limited to a 2-D representation here). In their analysis of wake structures created by the steadily swimming eel using particle image velocimetry (PIV), Tytell et al. report finding a wake dominated by lateral jets, formed by vortex pairs of opposite sign aligned parallel with the direction of flow. They argue that each reversal of the eel's tail direction during swimming begins a process of shedding boluses of fluid that are "sucked" into the troughs of waves traveling along the eel's body and accumulate in the boundary layer. The continuous lateral sweeping of the tail results in vorticity being advected downstream from the tail's tip in the form of an elongated vortical shear layer, which, once shed from the rear of the body, is unstable in its geometric configuration and rolls up into two vortices—a primary and a secondary vortex—of the same rotational sense. As the tail changes direction, the same thing happens on the opposite side of the body to give another primary and secondary vortex pair, each possessing a sense opposite to the previous vortex pair. In this way, two sets of vortex pairs are formed during each complete tail beat, resulting in two lateral jets of opposite direction alternately staggered in the eel's wake.

Figure 3-29 and Figure 3-30 show contour plots of vorticity for one complete tail beat cycle at Strouhal numbers of St = 0.3 and St = 0.5, respectively, and at a Reynolds number of $\mathrm{Re} = 5000$. Each of these sets of plots reveal vortical wake structures that qualitatively support Tytell's findings; in snapshot (A), the position of the eel's tail to its right side (top side in the figures) leads to a thicker boundary layer on the left side of the tail, presumably generated by the body's curvature. As the tail progresses from right to left (top to bottom in the figures), this "pocket" of fluid is advected downstream while the sweeping tail elongates it into a stretched shear layer (B and C). In (C), instabilities are beginning to form in this high-aspect-ratio shear layer once the tail has traversed over its maximum lateral distance, and the shear layer pinches off into two vortices (frames D and E) of the same sign—a primary vortex and a secondary vortex—that continue to separate as they are advected further into the wake. This result indicates that a line of counter-rotating vortices on each side of the wake is just beginning to form, in a manner similar to what is predicted by Tytell's PIV experiments, and may continue to match those experiments more closely as physical Reynolds numbers are approached.

One notable difference between the $\mathrm{St} = 0.3$ and $\mathrm{St} = 0.5$ cases, aside from the decreased streamwise spacing caused by increasing the vortex shedding frequency, is that the overall width of the wake has decreased with increasing Strouhal number; vortices of negative (clockwise) sense being shed from the right side of the eel's tail have moved toward the left side, and vice-versa, so that the vortices are now arranged closer to the wake's centerline. This is likely due to the fact that the transverse velocity has increased with respect to separation zone residence time, and so the tail has moved closer to its centerline position by the time a vortex leaves its surface on each pass. Examining Figure 3-31shows that this trend has continued into the higher Strouhal number $\mathrm{St} = 0.7$ case, so that vortices in the wake have actually been pulled across its centerline to form the beginnings of a reverse Kármán vortex street. Thus, the wake has transitioned from one exhibiting drag signatures to one of thrust production as the Strouhal number has increased.

Figure 3-32 and Figure 3-33 show all three Strouhal number regimes together, at Reynolds numbers of $\mathrm{Re} = 2500$ and $\mathrm{Re} = 5000$, respectively, so that they may be more easily compared visually. Both Reynolds number regimes are qualitatively similar, but increased diffusion in the lower Reynolds number case has led to vorticity patterns that are more stable and thus do not exhibit the same behavior of a high aspect ratio shear layer separating into two distinct vortical regions the way those in the $\mathrm{Re} = 5000$ case do. For this reason, vortices advecting downstream in the wake also diminish more quickly as they are diffused into the surrounding flow. Because the higher Reynolds number case offers a better approximation of material reality, it is the focus of consideration from here on out.

Contours of lateral velocity (plotted for all 3 Strouhal numbers in Figure 3-34) confirm, at least qualitatively, that the counter-rotating vortices produced by the eel's tail motion result in fluid flowing laterally from the wake, alternating between left and right sides as dictated by the tail's continuously changing direction. The lateral jets that are predicted by Tytell's experiments are not present here per se, but the low Reynolds numbers and relatively coarse resolution of these preliminary cases, along with restriction of motion to two dimensions, are likely contributors to this lack of agreement.

Figure 3-35 shows streamline plots in the immediate vicinity of the eel's body through one complete tail beat cycle for St = 0.7, Re = 5000. (It should be noted here that streamline plots were generated for all of the Strouhal and Reynolds number regimes evaluated, and all were qualitatively identical.) The most notable feature of this set of plots is the fact that it illustrates boluses of fluid traveling along the body lengthwise in the troughs of the waveform produced by the eel's swimming motion. Again, this seems to match well with Tytell's experimental observations.

Contours of axial velocity are plotted in Figure 3-36, further illustrating the earlier observation that the American eel lacks the signatures of a thrust-generating wake at Strouhal numbers of St = 0.3 and St = 0.5, but that thrust appears once the Strouhal number is increased to St = 0.7. Indeed, at a Strouhal number of St ≈ 0.3, corresponding to steady-state swimming, Tytell's results indicate similar behavior, raising questions about the eel's swimming efficiency; much energy appears to be expended on lateral fluid displacement that does not contribute to forward motion. Yet, anguilliform swimming is found to be prevalent in nature, and ocean-dwelling eels migrate over extreme distances [2, 58]. Lighthill's elongated body theory predicts that anguilliform swimming *should* be efficient in the absence of thrust production, though it is an inviscid model that does not suffer realisms like drag. Carangiform swimmers, such as tuna and other fish that move using strong, localized tail beats, do produce a wake that has a large axial velocity component—even during steady-state swimming—that is typically generated by shedding linked vortex rings [64-66]. Thus, their mechanisms for forward propulsion are relatively straightforward and obvious. The steadily swimming eel, however, created a wake possessing a relative axial velocity deficit and stronger lateral components in these simulations.

One possible explanation for the eel's ability to maintain forward locomotion in the absence of obvious thrust signatures in the wake lies a little further upstream, in the boundary layer next to its body. Carangiform swimmers are essentially composed of a nearly rigid body being propelled by a flapping tail; most of the fish's body contributes little to reduce drag in the swimming process, and that drag must be balanced by thrust generation in the tail in order to maintain steady motion. However, the eel's anguilliform motion possibly helps to decrease drag production by transporting fluid boluses in troughs along the length of its body [67]. To help lend further insight, lift and drag forces were computed on interfacial fluid cells and numerically integrated to give total body lift and drag coefficients:

$$C_L \equiv \frac{\ell}{({}^{1}/_{2})\rho U^2 L} \qquad \textbf{(3.9)}$$

$$C_D \equiv \frac{d}{({}^{1}/_{2})\rho U^2 L} \qquad \textbf{(3.10)}$$

In Equations 3.9 and 3.10, ℓ and d represent 2-dimensional lift and drag force components (force per unit length), ρ is fluid density, U is bulk fluid velocity, and L is the characteristic length scale being considered for lift and drag production. The swimming eel is a slender body, so its characteristic length—that which dominates the flow physics—is the distance from its nose to its tail.

These lift and drag coefficients are illustrated in Figure 3-37 and Figure 3-38, which show their temporal evolutions through one complete tail beat cycle during steady-state swimming. The instantaneous lift coefficients are notable in that they are roughly an order of magnitude larger than their corresponding drag coefficients, suggesting the possibility of high instantaneous energy costs expended on lateral excursions for the benefit of an average positive thrust. It should be remembered, however, that the eel is laterally stationary overall, and so is constantly accelerating fluid in the lateral direction between Y-velocity magnitudes of $\pm 1.0\ cm\ s^{-1}$ along portions of its body (Figure 3-34).

Maintaining steady-state swimming in a forward direction, on the other hand, requires only that fluid is axially accelerated to a velocity sufficient to overcome any momentum removed from the fluid in front of the eel's body [58]. If the eel were accelerating from rest to its steady-state swimming speed of $1.0\ cm\ s^{-1}$, drag may more closely match lift in magnitude.

One prominent feature of the instantaneous drag coefficient plot is that the vast majority of thrust production is seen to occur at the beginning of each tail beat cycle, when the tail is situated in its right (top) extreme position and beginning to traverse to the left. In each of the cases evaluated here, this was the initial position assigned to the eel's tail from which swimming motion was programmed to commence. Interestingly, this behavior has also been observed in experiments and numerical simulations involving flapping plates; maximum thrust was found to occur periodically whenever the plate returned to its starting configuration [68-69]. It is suspected that this is caused by a slight asymmetry in the wake that depends on initial position and direction of motion, and can result in the wake deviating from its expected trajectory. Careful examination of the eel's wake plots reveals that the wake is indeed deflected to the right (upward) slightly, rather than taking a perfectly symmetrical and horizontal path. Of course, this could also be a result of the eel not being perfectly aligned in the images that it is being modeled from, so it would be interesting to run some cases in which the tail is initialized on the left (bottom) to see whether the wake is similarly deflected to the opposite side. (It could also be a different matter altogether: perhaps the eel being filmed was swimming in the presence of a cross-flow component and so actually *was* producing an asymmetric thrust signature.)

Because drag forces pertain more directly to thrust generation, they will occupy the primary focus for the remainder of this analysis. Figure 3-42 is a plot of drag forces averaged over one complete tail beat, for each of the Strouhal number regimes evaluated in these simulations. At a Strouhal number of St = 0.3, the drag coefficient is small—notable because this is approximately the same Strouhal number of the steadily swimming eel that these CFD simulations are fashioned after. “Steadily swimming” is really the key term here; if an animal is swimming steadily, then it cannot be generating net thrust because it is not accelerating (assuming that it is sufficiently buoyant as to not be constantly working to overcome gravity, i.e. its density very closely matches the density of its environment). Thus, a balance of thrust and drag along the body through one tail beat at this Strouhal number is precisely the result anticipated.

Surprisingly, increasing the Strouhal number to St = 0.5 in these simulations did not result in an increase in thrust (a decrease in the drag coefficient) as would be expected; in fact thrust did not appear at all until the Strouhal number was increased to St = 0.7. While the wake plots in Figure 3-33, Figure 3-34, and Figure 3-36 anticipate this result somewhat, the change is expected to occur motonically [68].

In order to check these results, a control volume analysis was performed on the eel simulations in addition to the surface force calculations. For each case, a control volume spanning from 0.5 units length upstream of the eel to 1.0 unit length downstream in the x-direction, and spanning the domain (from 0.0 to 2.0 units length) in the y-direction, was defined in which to calculate a momentum balance (Figure 3-40). Since the north and south edges of the control volume lay along the north and south edges of the domain, which were assigned Dirichlet boundary conditions, all momentum flux was assumed to occur across the east and west faces. Drag force per unit length experienced by the eel's body was then calculated by integrating over the east (outlet, boundary 2) and west (inlet, boundary 1) edges of the control volume:

$$d = \rho \int_0^h \left(u_1^2(y) - u_2^2(y)\right)dy + \int_0^h \left(p_1(y) - p_2(y)\right)dy, \qquad \textbf{(3.11)}$$

where h is the span of the control volume in the y-direction, $u(y)$ are x-velocity components along the control volume edges, and $p(y)$ are pressure values. The results of this control volume analysis are given as instantaneous drag coefficient measurements in Figure 3-41 and average drag coefficients for each Strouhal number regime in Figure 3-42 (plotted along with the average surface drag coefficient measurements for comparison). It can be seen that instantaneous wake drag coefficients are similar to those found by surface analysis, though shifted temporally on the tail beat fraction axis. This is confirmed by the average thrust coefficient plots in Figure 3-42, which bear resemblance to each other. Curiously, the St = 0.3 case exhibits a small thrust value when analyzed using the control volume method, but this might simply be attributable to error introduced by neglecting viscous terms in the analysis. In any case, what is clear is that an actual eel, unconstrained by the boundary conditions imposed in these models, would be accelerating by beating its tail at higher frequencies in the St = 0.7 range.

In considering the foregoing physical analyses, it is important to remember that their validity is completely dependent upon the fidelity with which the geometry and motion of the swimming eel has been represented. Converting the eel from a set of (coarse) images in a video file to a set of manipulated Lagrangian surface points, and then to a moving zero level set contour embedded in an Eulerian flow mesh, has caused the loss of some features that may be pertinent to the flow physics involved in the eel's locomotion. Figure 3-43 illustrates one particularly troublesome example: truncation of the eel's tail and surface point smoothing has resulted in a modeled tail shape that deforms in an unrealistic manner during motion. When viewed from above, an eel's tail should in reality taper nearly to a point, resulting in a sharply acute edge where the two sides of its body meet. The sharpness of the tail has been lost in these models, resulting in more of a bluff-body situation with regard to vortex shedding into the wake behind. In addition, the width of the tail changes during motion, and symmetry is not maintained at the point where the left and right halves of the eel's body meet. The vector fields plotted in Figure 3-43 reveal flow patterns around the eel's tail that would not necessarily be expected if it tapered nearly linearly to a point as it should, which may cause a disparity between actual and modeled vortical transport and its resultant wake morphology and thrust generation.

Perhaps most importantly, it should be borne in mind that maintaining one-to-one correspondence of Lagrangian surface points between sets of frames, along with surface closure and connectivity to the Cartesian fluid mesh, required much manipulation of points in a direction tangent to the surface they were supposed to represent. Such tangential motions caused by point population and subsequent shifting to give equal intervals have nothing to do with motion in material reality, and so contributed undesirably to the surface velocity calculations used to set boundary conditions on the flow mesh during CFD simulation. While these spurious velocity contributions should be small relative to bulk fluid motion, owing to the close spacing of Lagrangian surface points, they nonetheless arise in an unfortuitous place; namely, the surface responsible for generating all of the shear responsible for vorticity generation and propulsion.

Despite these limitations present in the current method, qualitative results were still felt to be sufficiently encouraging to justify further development of this paradigm of generating CFD models from image files. With a purely Eulerian system and higher resolution video images, high-fidelity complex simulations were seen to present themselves within the realm of near-future possibilities.

3.7 Internal Flow: Mass Transport and Mixing in the Small Intestine

The swimming eel described in the previous section represents an external flow problem; fluid transport through internal systems represents the other major component of fluid mechanics research, and shall be the present topic of discussion.

The gastrointestinal tract is a uniquely interesting internal flow system, in that it is able to transport a wide variety of fluids and particulate slurries at any point throughout its length—some sufficiently viscous and particle-laden as to barely qualify as fluids in the classical sense—simply through the organized actuation of its surfaces, without the assistance of any upstream pumping mechanism. All of this occurs for the singular purpose of extracting the chemical energy required to keep its owner alive (while removing what is not needed), and it happens with sufficient efficiency to leave plenty of energy left over for all of the intrinsic and extrinsic processes that make life what it is. Dr. Konrad Schulze, a physician with the University of Iowa's Department of Internal Medicine, has been researching gastrointestinal morphology and flow for the purpose of providing a greater level of biological and clinical insight to the field. A video file of an excised guinea pig (*Cavia porcellus*) duodenal segment contracting in a fluid tank was used to create the simulations discussed in this section.

The duodenum is the first part of the small intestine, where gastric chyme is deposited from the stomach through the pylorus (the "gatekeeper" described in Chapter 2). It is thought that the primary role of the duodenum is one of mixing; pancreatico-biliary secretions are introduced to the chyme from the stomach here by way of the common bile and pancreatic ducts [70], and the fluids are mixed via peristaltic contractions while being slowly transported distally [5]. In Schulze's experimental setup, a duodenal segment of approximately 10 cm in length and ½ cm in diameter was excised from a male guinea pig and placed into a fluid tank containing an isotonic solution (Krebs), spanning between two stopcocks attached to sections of tubing that each led to a small fluid reservoir cup. The duodenum was primed, and the cups filled, with Krebs solution, so that peristaltic motion by the duodenal segment would produce flow from one cup (referenced as the inlet, or proximal, cup) to the other (the outlet, or distal, cup), and vice-versa.

Just as with the American eel simulations described in the previous section, the AVI video file provided by Dr. Schulze was converted to a set of raw data files using Matlab 2007A, and those were segmented using the simplified k-means approach outlined earlier. A set of 300 frames was chosen, each frame separated by a physical time of 0.33 seconds, with the whole set corresponding to a sequence of 7 primary duodenal contractions occurring over a period of 99 seconds. Correlating pertinent data for the study were recorded during image acquisition and supplied by Dr. Schulze. The provided data included proximal and distal cup volumes, obtained by weighing each cup continuously throughout the experiment, pressure through the proximal and distal stopcocks, and duodenal diameter measured at 32 evenly spaced locations along its length using an open source program called Motility Mapping (Figure 3-44).

In order to facilitate segmentation and limit it to regions of interest, the duodenal images were cropped between the two stopcocks to which the duodenum was attached. After pre-processing the image segments using all of the steps outlined early on in this chapter, fixed inlet and exit regions were added to the channel flow domain. This was done for several reasons: 1) inlet and exit segments of the computational flow domain must remain a constant width to enforce boundary conditions and maintain mass flow conservation; 2) the experimental setup featured fixed inlet and exit tubing sections, so it made sense to provide the same conditions computationally; and 3) axial extension of the domain was desirable to prevent scalar species blocks (first described in Chapter 2), used to measure flow mixing rates, from exiting the domain and thus ceasing to fulfill their purpose. All lengths were normalized based upon the fixed inlet diameter for convenient Reynolds number calculation. The result was a domain of 30 units length by 3 units height, with a coarse grid spacing of $\Delta x = 0.15$. Local mesh refinement was used to reduce this spacing up to 2 more levels to $\Delta x_{min} = 0.0375$ where necessary to resolve boundaries, flow features, and regions containing scalar species (Figure 3-45 and Figure 3-46).

Using the data provided by Dr. Schulze, a maximum Reynolds number at the inlet was calculated so that flow conditions could be matched in the simulations. In the absence of directly provided inlet velocities, the Reynolds number was computed using proximal cup volume data (Figure 3-47). The proximal cup volume was given in ml (or

cm^3), and it was known that the tubing to which the duodenum was attached had an internal diameter of approximately ½ cm (giving a cross-sectional area of roughly 0.196 cm^2).

$$A_c = \frac{\pi}{4}(0.5\ cm)^2 \approx 0.196\ cm^2 \qquad \textbf{(3.12)}$$

The maximum cup volume change was averaged over a 5 second period, to eliminate the effects of noise in measured values, resulting in a maximum volumetric flow rate of 0.08 $cm^3\ s^{-1}$.

$$Q_{max} = \frac{5.3\ ml - 4.9\ ml}{5\ s} = 0.08\ \frac{cm^3}{s} \qquad \textbf{(3.13)}$$

The maximum volumetric flow rate was used in turn to give a maximum inlet velocity:

$$U_{max,inlet} = \frac{Q_{max}}{A_c} \approx 0.41\ \frac{m}{s} \qquad \textbf{(3.14)}$$

A kinematic viscosity of $\nu = 10^{-6}\ m^2\ s^{-1}$ (0.01 $cm^2\ s^{-1}$) was used to approximate a Reynolds number of 20.5, assuming Krebs solution to have properties similar to liquid water:

$$Re_{D,max} = \frac{U_{max,inlet} D_{inlet}}{\nu} \approx 20.5 \qquad \textbf{(3.15)}$$

After calculating the maximum inlet Reynolds number to set boundary conditions, it was discovered that a key piece of information, required for image-based modeling of this experiment to proceed, was missing: it was not known precisely which image frames in the video file were used to generate the data provided. Thus, there was no correlation between what was happening in the selected image frames and what was happening with the inlet reservoir cup. Assuming that the imaged frames are essentially nothing but 2-D projections of the duodenum's channel area, it was decided to investigate whether the channel diameters recorded by the Motility Mapping program could be used for generating the necessary boundary conditions. 31 trapezoidal panels of approximate width 1/3 cm were used to estimate the total channel area of the imaged duodenal segment; each panel was constructed using 2 adjacent point pairs. For each panel, the left point pair was defined as being separated by distance A, and the right was defined as separated by distance B. The panel width of 1/3 cm can be denoted as C. Each panel's area was then calculated as:

$$Area_{panel} = \frac{C}{2}(A+B) = \frac{A+B}{6} \quad \textbf{(3.16)}$$

Summing the panel areas gave an approximation of the total 2-D channel area for each time step *n* in the provided data set (Figure 3-48):

$$A^{n}_{channel} = \sum_{i=1}^{31} A^{n}_{i} \quad \textbf{(3.17)}$$

Examination of the data in this manner revealed a strong correlation between flow to and from the proximal cup, and changes in the inverse projected area of the contracting duodenal segment (Figure 3-49). This was particularly true in the period of time between 40 s and 80 s, where the volumetric flow rate appears to have reached a kind of quasi-steady-state behavior. Thus, it was decided to calculate channel area during image segmentation and construction of the geometric model and store it, for each fiducial time step, for use in calculating inlet velocity boundary conditions for the flow domain. The maximum change in inverse area throughout the image frame set was found to be approximately 7.62×10^{-4}, or 1/1312. It was desired to generate a maximum inlet velocity of $U_{inlet,max} = 1.0$ for convenient maximum Reynolds number evaluation by way of simply altering the fluid's dynamic viscosity, so the inlet velocity was set as 1312 times the change in inverse channel area between any two image frames. With a fiducial time step size of $\Delta T = 0.33\ s$, the inlet velocity for any image time step n could then be written as:

$$U_{inlet}^{n} = \frac{433}{\Delta T}\left(\frac{1}{A^{n}} - \frac{1}{A^{n+1}}\right) \qquad \textbf{(3.18)}$$

Inlet velocities were linearly interpolated for intermediate time steps Δt in the same manner as interfacial velocities as described earlier in this chapter.

Small oscillations in the channel walls between fiducial time steps resulted in a noisy temporal inlet velocity profile that led to difficulties converging pressure in the flow solver. Therefore, the entire inlet velocity profile was smoothed using a Savitzky-Golay filter of window size 20, chosen for its ability to remove noise while maintaining the original curve's overall behavior (Figure 3-50).

With boundary conditions established and coupled with the morphology of the imaged duodenum, it was decided to run simulations at three different Reynolds numbers. The first simulation was run at $Re = 20$, closely matching that of the actual guinea pig duodenum being modeled. The second and third runs were performed at $Re = 100$ and $Re = 500$, respectively; the human duodenum is on the order of 2.5 cm in diameter [personal communication with Dr. Konrad Schulze], which is 5 times the diameter of the guinea pig duodenum. Thus, a Reynolds number of 100 would be produced if the velocity were held constant with the geometry increased to human scale. Concurrently increasing inlet velocity to match the velocity/diameter ratio of the guinea pig leads to a Reynolds number of 500.

The physical experiments were performed with Krebs solution, which is similar to water in viscosity. It is likely that chyme will, in actuality, have different characteristics if anything but water or a similar liquid is ingested, so it is hoped that the Reynolds numbers studied here will provide insight for a range of mixing possibilities. It was not possible to run a maximum inlet Reynolds number higher than 500, as the resultant jet through narrowing portions of the duodenum produced localized Reynolds numbers that were too high for direct numerical simulation (DNS) of flow on such a course mesh without some sort of turbulence modeling capabilities.

Figure 3-51, Figure 3-52, and Figure 3-53 illustrate contour plots of axial velocity within the flexible portion of the channel (representing the duodenal segment itself) for each of the three Reynolds numbers simulated. This set of images shows 6 instants during the first peristaltic contraction and subsequent expansion, taking place over a period of 13.2 s. Frame (A) (14.52 s into the simulation) marks the very beginning of the peristaltic cycle, effecting primarily axial transport from the proximal end to the distal end of the duodenum. Contraction continues through frame (B), where distal axial velocity has grown considerably in magnitude due to the squeezing of the channel near its center as peristaltic wave amplitude approaches its maximum. In frame (C), the channel width has reached its minimum, and is beginning to expand back toward its resting state, starting the process of flow reversal. Frame (D) shows that the channel has widened considerably in the single second that has passed since frame (C), producing a strong backflow toward the proximal end of the duodenum. Expansion decelerates through frame (E) until the resting configuration shown in frame (F) is reached.

Axial velocity magnitudes are similar in each of the Reynolds number regimes, because Reynolds number was determined by setting the dynamic viscosity of the fluid while keeping the inlet boundary conditions the same between cases. However, flow field stability is clearly different between the cases; $Re = 20$ gives a fairly smooth axial velocity profile considering the complexity of the waveform effecting it, while $Re = 500$ shows clear signatures of flow field instability—increasingly so as the channel expands and flow reverses.

Contours of vorticity (Figure 3-54, Figure 3-55, and Figure 3-56) and streamline plots (Figure 3-57, Figure 3-58, and Figure 3-59) for the same instants in time further illustrate the complicated flow features generated by increasing the Reynolds number. For the low Reynolds number case ($\mathrm{Re} = 20$), a thick boundary layer is formed and pushed distally as fluid is squeezed through the contracting channel, but remains stably attached to the channel walls throughout the peristaltic cycle. Streamlines indicate that fluid is initially pushed radially until the axial velocity generated by the traveling peristaltic wave becomes sufficient to orient the streamlines parallel to the channel walls. As axial velocity slows and then reverses while the channel opens up, streamlines move once again toward an orientation perpendicular to channel walls.

Increasing the Reynolds number to $\mathrm{Re} = 100$ results in a thinner boundary layer that is more sensitive to flow reversal and channel expansion; Figure 3-55 (C) indicates that the boundary layer formed during forward transport of fluid from proximal to distal is lifted from the channel walls as the flow reverses and the channel widens. The separated boundary layer has an aspect ratio that is too high to maintain naturally, and so rolls up into a series of weak vortices. However, the Reynolds number is still of an order in which viscosity exhibits a strong presence, and the vortices quickly dissipate into the surrounding flow field as a result.

The high Reynolds number of 500 produces the most interesting flow field topology of the three cases. The thin boundary layer generated by the squeezing action of the duodenum is highly unstable when the flow reverses and the channel opens (Figure 3-56 and Figure 3-59 (C) and (D)), and quickly rolls up into a series of vortices which are now slower to dissipate due to their relatively higher momentum (Figure 3-56 and Figure 3-59 (E)). The final result when the channel returns to its resting state is a series of counter-rotating vortices, alternating in rotational sense, covering roughly 1/3 of the channel's length (Figure 3-56 and Figure 3-59 (F)).

Just as in the idealized geometric representation of the pylorus in Chapter 2, passive scalar species blocks were placed in the duodenal domain for each of these cases, as a way of quantifying mixing effectiveness. Three species blocks ($\varphi_i; i = 1,2,3$) were initialized in each simulation, with a concentration value of 1.0 and an area of 0.5 square unit length (1.0 unit length by 0.5 unit height). The blocks were spaced evenly in the flexible portion of the channel, centered about $L = 8.0$, $L = 12.0$, and $L = 16.0$, respectively, and allowed to advect passively with the flow according to the transport equation:

$$\frac{\partial}{\partial t}\iiint \varphi dV + \oint \varphi(\boldsymbol{u}\cdot\boldsymbol{n})dS = \frac{1}{\mathrm{Pe}}\oint \boldsymbol{\nabla}\varphi\cdot\boldsymbol{n}dS \tag{3.19}$$

The Peclet number, $\mathrm{Pe} = (U_o L)/D$, was kept large by way of setting the diffusivity constant D small (10^{-5}) for each of the scalar species blocks, yielding a small right hand side to Equation 3.17 and ensuring that scalar species would follow fluid streamlines without mixing much by way of diffusion compared with advective mixing.

Figure 3-60 through Figure 3-62 show evolution of the first scalar species block, centered about $L = 8.0$, through the duodenum during the first peristaltic contraction cycle, for each of the three Reynolds numbers. Frames (B) through (G) represent the same instants of time illustrated previously in plots of axial velocity, vorticity, and streamlines. It is clear that increasing flow field complexity leads to more vigorous mixing of the scalar species present in the channel. The $\text{Re} = 20$ case simply acts to stretch the block axially during contraction, and subsequent expansion and flow reversal simply leave the species oriented in a quasi-parabolic shape that matches well with expected internal flow behavior. Increasing the Reynolds number to 100 yields similar results, though small boundary layer instabilities in the region of $L = 11.0$ somewhat separate the block into two regions. This results in the banded quasi-parabolic structure seen in Figure 3-61 (F) and (G). Increasing the Reynolds number further, to 500, results in an entirely different flow regime which is marked by large-scale instabilities and subsequent vortex formation occurring during channel expansion and concurrent flow reversal/deceleration. These vortices stretch the scalar block into long, thin sheets wrapping around the vortex cores. The large increase in area that results from such topological stretching of the scalar species field leads to a rapid decrease in species concentration; concentration appears to have been reduced by approximately half during this first peristaltic contraction alone (Figure 3-62).

One of the difficulties inherent to quantifying flow field mixing with advecting species blocks is that the blocks cannot visit every region of the domain, and so mixedness is only measured where the species are transported to at any given instant. This introduces the possibility that important flow features may be missed altogether, which is illustrated in Figure 3-63: (A) shows the initial placement of scalar block 1, and its configuration after one peristaltic contraction with $\text{Re} = 500$; (B) shows the same for scalar block 2. Scalar 1 has clearly been exposed to a region of the flow field that exhibits more vigorous mixing than the region visited by scalar 2. After all 7 primary peristaltic contraction cycles, scalar 1 has decreased by well more than an order of magnitude in concentration and spatially occupies nearly the whole of the duodenum, while scalar 2 is relatively higher in concentration and confined to a small region in the distal end of the channel (Figure 3-64).

Scalar variance measurements were recorded for each species block φ in the same manner as in Chapter 2, where variance is defined as

$$\mathrm{var}(\varphi) \equiv \langle \varphi^2 \rangle = \iiint (\varphi - \bar{\varphi})^2 dV \qquad \textbf{(3.20)}$$

and

$$\bar{\varphi} = \frac{\iiint \varphi dV}{\iiint dV} \qquad \textbf{(3.21)}$$

Normalized variance measurements for scalar blocks 1 and 2 at each Reynolds number are provided in Figure 3-65 through Figure 3-69. Variance and field contour plots for species block 3 are not included, because initialization of the block so close to the right hand side of the channel eventually caused it to pass out of the domain's outlet during simulations, rendering it useless for data collection. Figure 3-65 and Figure 3-66 illustrate the increased rapidity of mixing that comes with increasing the Reynolds number, as does Figure 3-70, which shows the final form of species block 1 after all 7 peristaltic cycles for $\mathrm{Re} = 20, 100, \text{and } 500$. All of the Reynolds number regimes reduce the scalar variance of each block by a considerable amount throughout the course of the peristaltic contractions—at least an order of magnitude in most cases. With $\mathrm{Re} = 500$, both species blocks appear to be nearly homogenized throughout the domain according to the plots. However, this raises one of the limitations of quantifying mixedness by way of scalar variance measurements that average species concentration over the entire domain: in Figure 3-69 ($\mathrm{Re} = 500$), species blocks 1 and 2 appear to have virtually the same value at the end of the simulation. Yet, examination of Figure 3-64 clearly indicates otherwise. It appears that a low species concentration occupying a large portion of the domain gives results that are similar to those of a relatively higher species concentration occupying a smaller portion of the domain, at least when the absolute magnitudes of both are small. In future simulations, it may be wise to calculate scalar variance measurements only in regions actually occupied by the scalar species, rather than averaging over the entire domain.

The duodenal segment study has given some interesting results, in that they seem to support the notion that peristaltic contractions in the duodenum are at least partly responsible for enhancing chemical species mixing for the purpose of more rapidly effecting enzymatic catalysis and nutrient absorption in the gut. Unfortunately, the sort of data used for direct experimental comparison with Tytell's eel does not exist for the duodenum; the duodenum's opacity presents a challenge to flow visualization that did not have to be overcome with the eel experiments. In addition, the temporal mismatch between the data provided by Dr. Schulze and the image frames used to computationally model the experimental setup forced the development of a means to construct boundary conditions using projected channel area. This method was only valid during the time period between 40 and 80 seconds in the actual experiment, when quasi-steady-state behavior developed—likely as a result of fluid accumulation in the distal cup reservoir providing a pressure head at that end of the duodenum in the experimental setup. Thus, only the "sloshing" motion generated during peristaltic contractions was captured in these simulations, and overall proximal-to-distal transport was neglected. Looking at the proximal cup volume before and after the burst of peristaltic contractions in Figure 3-47, it appears that the volume changed from roughly 5.2 ml to 4.6 ml over the course of 90 seconds; this gives a volumetric flow rate of $6.\overline{66} \times 10^{-3}$ cm^3 s^{-1}, or an average transport velocity of $0.034\ cm\ s^{-1}$. This velocity is more than 12 times smaller than the peak velocity generated by peristaltic contractions in the guinea pig duodenum, so it is hoped that the skewness toward a mixing regime rather than a transport regime is not sufficiently ignorant as to completely invalidate these results.

3.8 Conclusions

So far, the idea of constructing 2-D computational flow models around moving boundaries using image files has been shown to be a viable one. The power of the current method is more clearly illustrated in the duodenal segment model in some ways, because the motions exhibited by the duodenum are highly complex and would likely be difficult, perhaps even impossible, to reproduce functionally. With the image-based approach, the functional complexities of geometry and motion no longer matter: it simply takes a series of pictures of something and creates a moving surface out of them. Simplicity is its primary appeal.

However, true simplicity has yet to be achieved with the approach taken presently, at least as far as implementation and generality are concerned. Eight tedious steps were required to generate the moving eel surface used in running these simulations (Figure 3-71), and the results were, in the end, possibly less accurate than what could be achieved by defining the eel's body as a long tapered ellipsoid undulating sinusoidally with a wave that grows steadily from head to tail. Still, higher resolution images should capture all of the nuances of shape more accurately than such an idealized geometry would, and the method as developed here did serve as a means to solve the problem of how to simulate the moving guinea pig intestine realistically.

Because image-based modeling holds so much promise in the way of offering a simple route to modeling complex phenomena, it is desirable to eliminate tedium from the process to the greatest extent possible. The level set techniques used in generating sharp Eulerian surfaces in the flow field as outlined in this work are attractive in their simplicity, because problems associated with moving meshes and Lagrangian point tracking are completely averted; and so an image-based modeling approach should strive to match it in its elegance. To this end, work outlined in the rest of this dissertation has been conducted with the aim of constructing a suitable method for segmenting images and directly generating level set surfaces from them, surfaces that can in turn be used directly by the flow solver to perform CFD analyses on a wide variety of complicated moving objects in the appropriately simple manner deserved by the Cartesian approach.

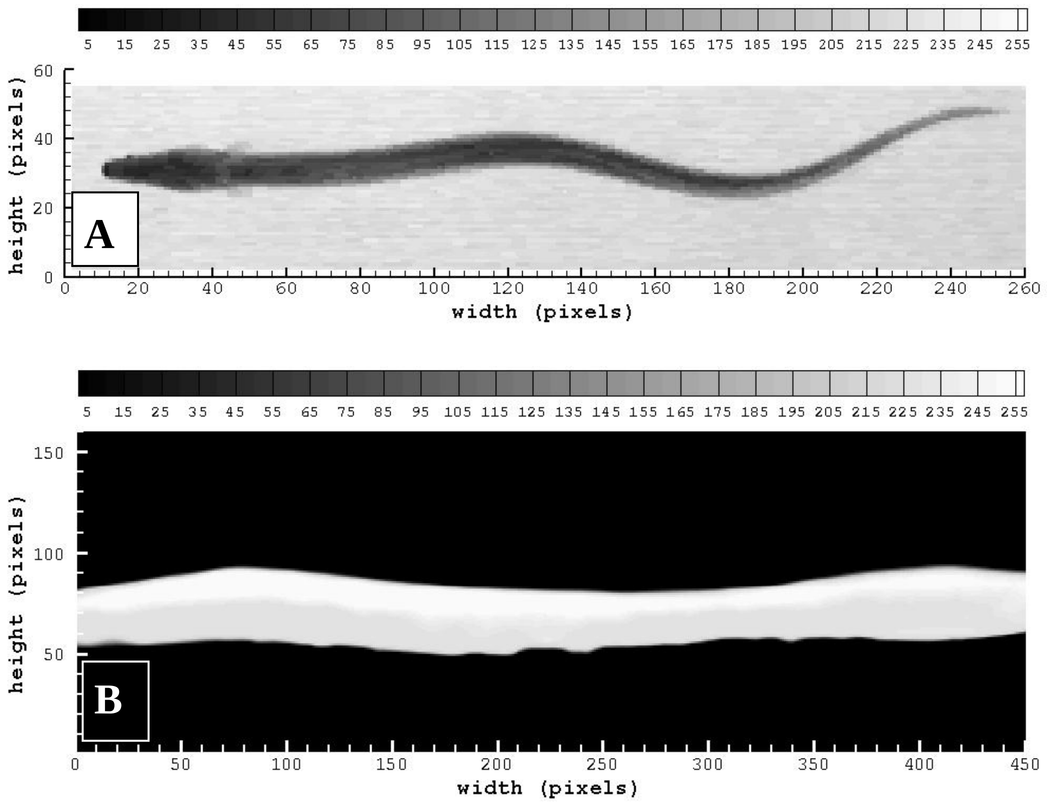


Figure 3-1. The first frames of the American eel (A) and the duodenum video sequences (B), respectively. A high level of contrast allowed for straightforward segmentation, but the relatively small number of pixels yields coarse segmentation contours that are not resolved well enough to perform flow calculations on directly. Thus, heavy population and smoothing of Lagrangian surface points is required.

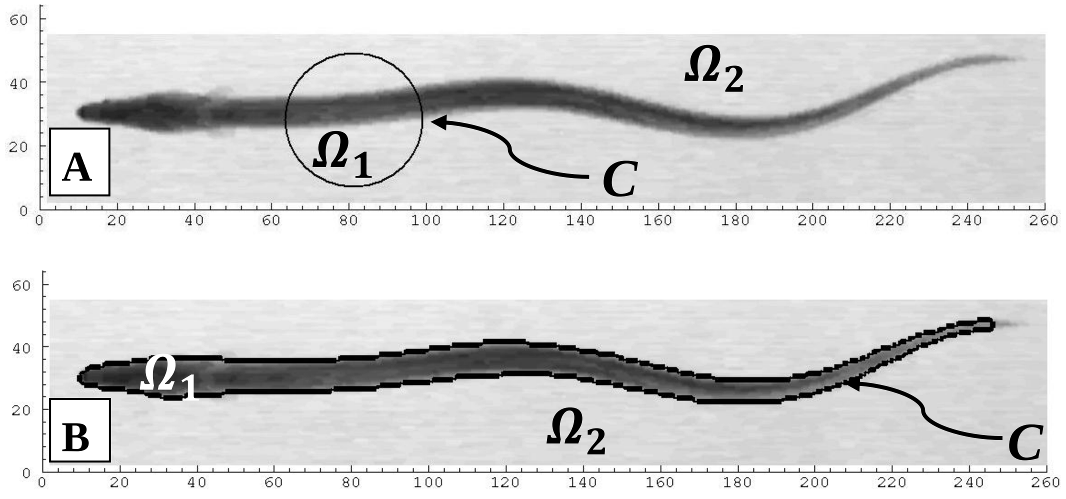


Figure 3-2. An arbitrary curve is initialized on the image pixels (A) and evolved based upon the relative intensities of inner region Ω_1 and outer region Ω_2 until curve C closely matches the contour of the body (B). The entire process generally takes 5 to 6 iterations to complete.

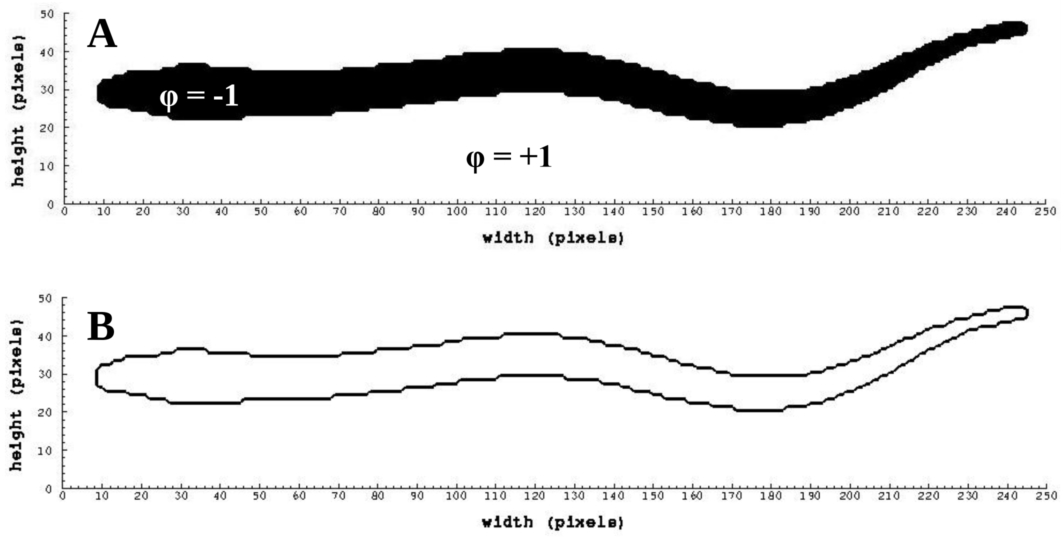


Figure 3-3. The pixel-resolution binary field (A) and resultant segmented boundary (B) of the American eel, during the first image frame of the video sequence, segmented using the K-means approach outlined in Fedkiw and Gibou.

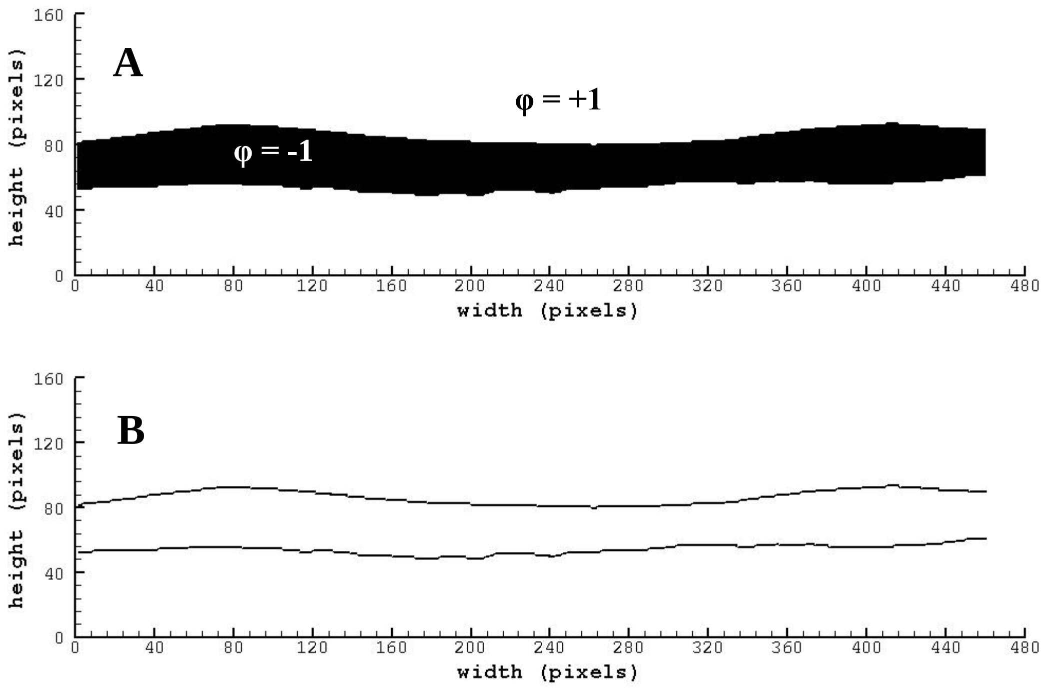


Figure 3-4. The pixel-resolution binary field (A) and segmented boundary (B) of the guinea pig duodenum studied *in vitro*, during the first image frame of the video sequence, segmented using the K-means approach outlined in Fedkiw and Gibou.

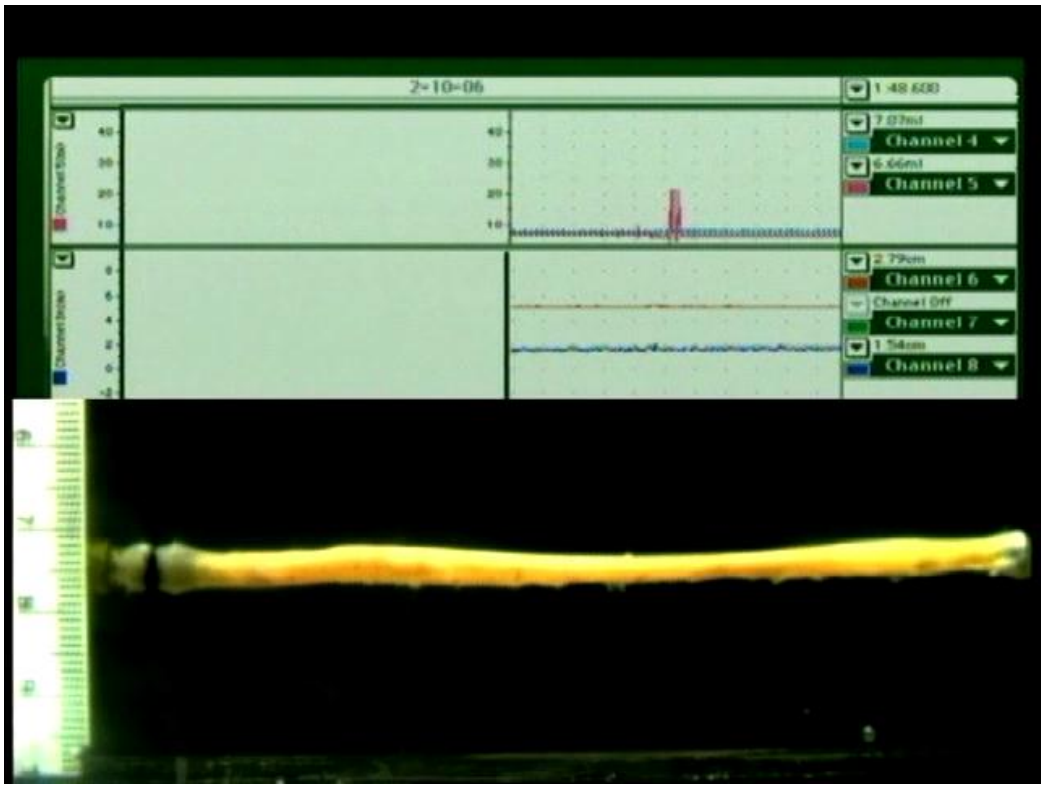
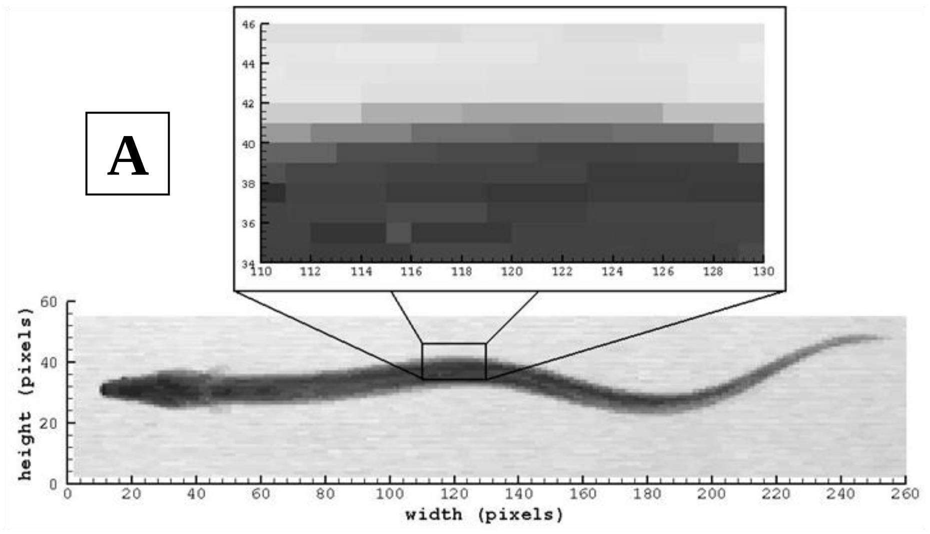


Figure 3-5. A duodenal image frame prior to cropping. The image was cropped to isolate the duodenum before any segmentation or other operations were performed.



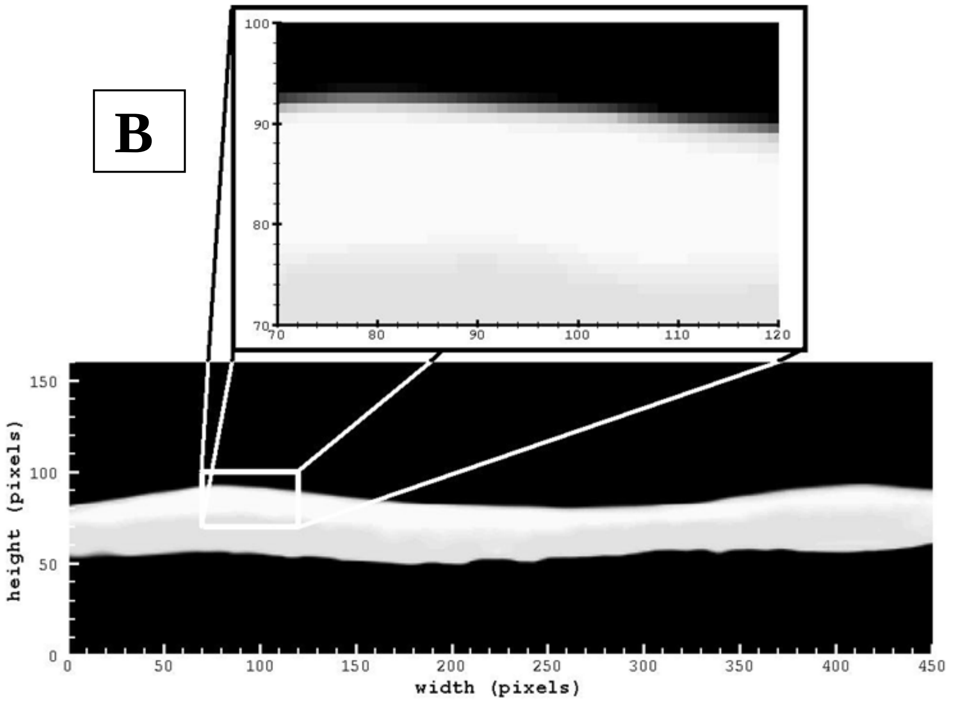


Figure 3-6. Blending along object contours leads to a checkerboard pattern in the image pixels, necessitating preprocessing of the image to create smooth segmentation boundaries. This problem is far more marked in the eel video frames (A), which have roughly half the resolution of the duodenal images (B).

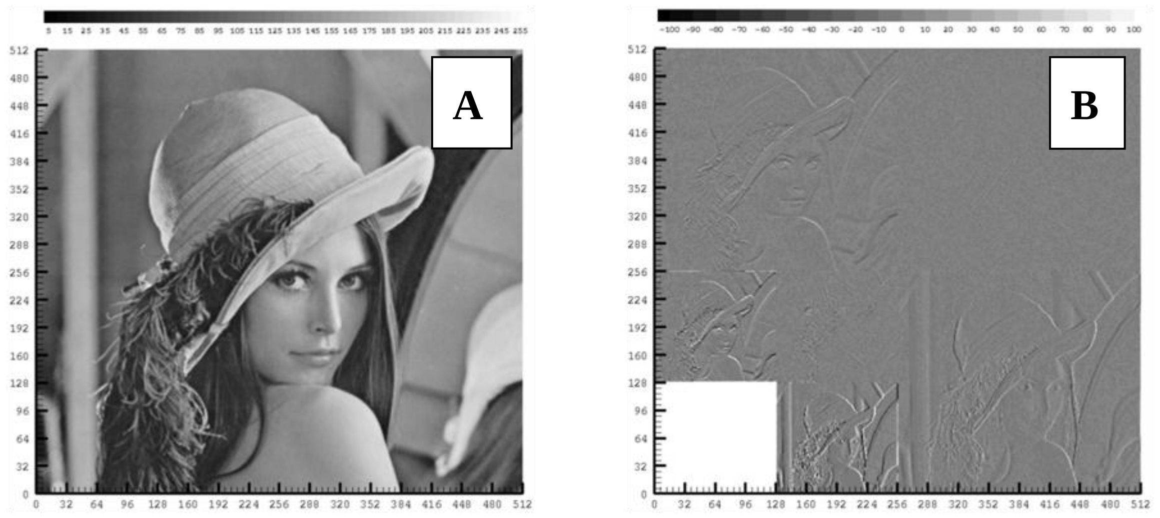


Figure 3-7. The discrete wavelet transform of the IEEE standard test image “Lena”: Original image (A) and the image transformed using a Haar wavelet (B), with 2 levels of scaling coefficients shown for illustration. Transforming the image in this manner continues to reduce it in detail until only 2 scaling coefficients remain.

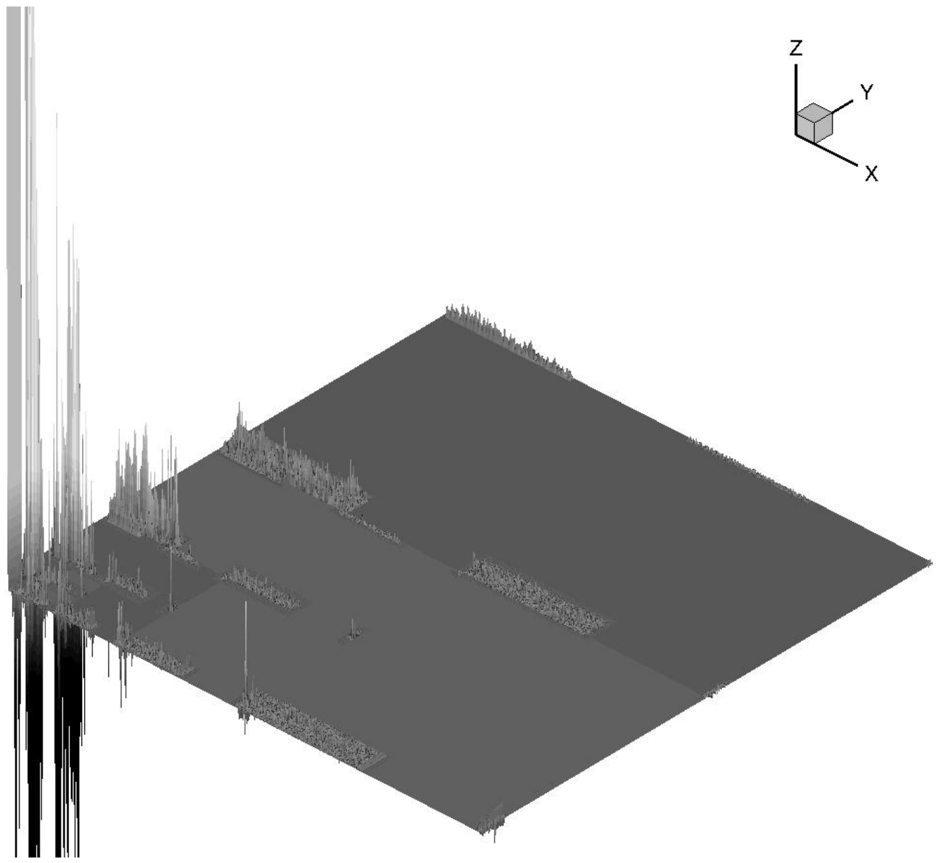


Figure 3-8. Wavelet energy at all scales in the first American eel image frame. Threshold filtering and smoothing can be accomplished in three ways: removing wavelet coefficient scales (generally from smallest to largest), removing wavelet energy that falls below a specified threshold at any given data point, or removing some percentage of the total wavelet energy (starting with the lowest energy values and working upward until the specified fraction is reached). Filtering by scale removes features of a particular size, while filtering by energy retains dominant features regardless of their size in the domain. Note that the image has been padded with extra cells to give it dimensions of length 2^N.

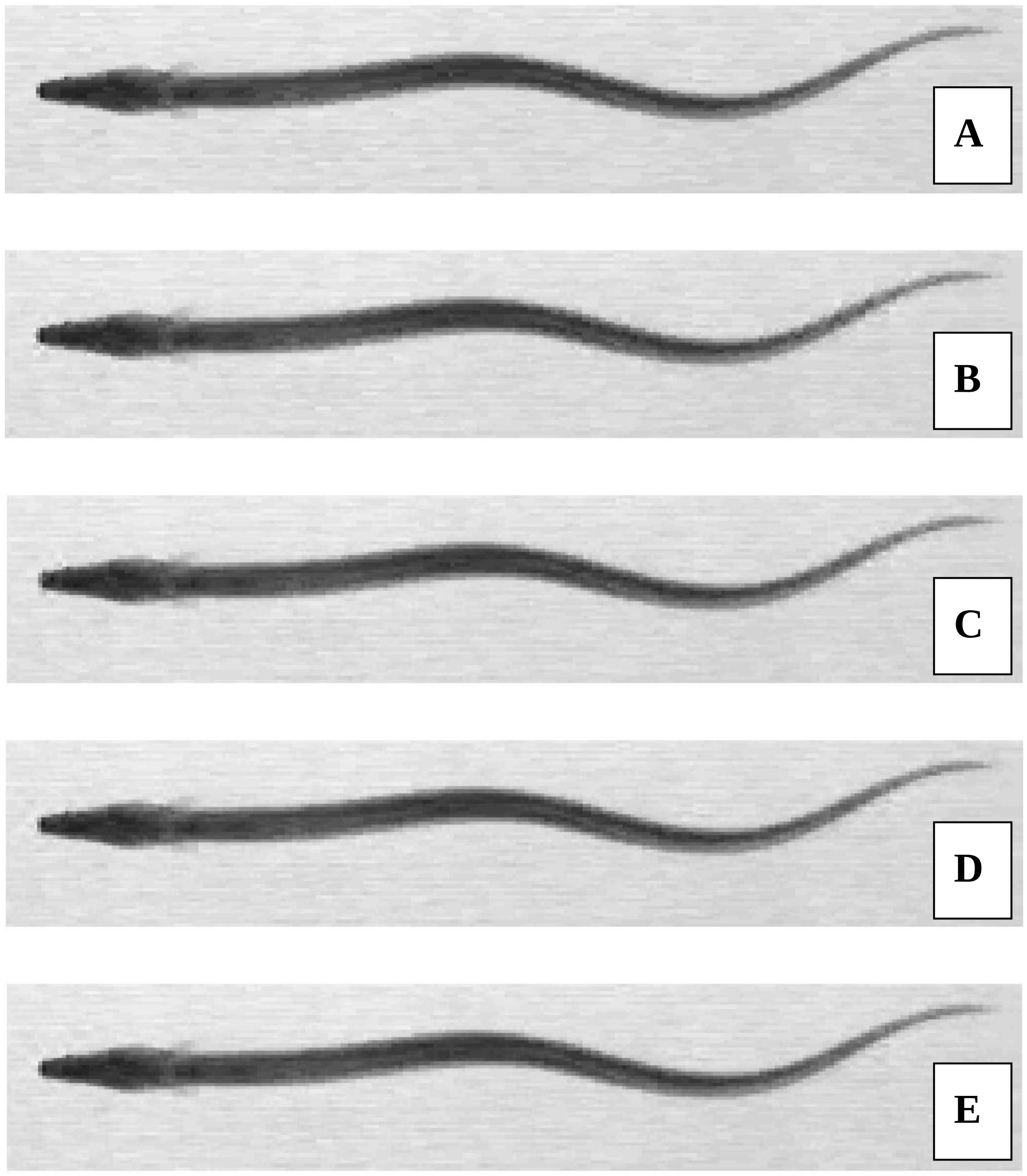


Figure 3-9. First frame of the American eel image, filtered using wavelets with 2 scales removed: Haar (B), Daubechies-4 (C), Daubechies-12 (D), and Daubechies-20 (E). The original image is shown in (A) for comparison.

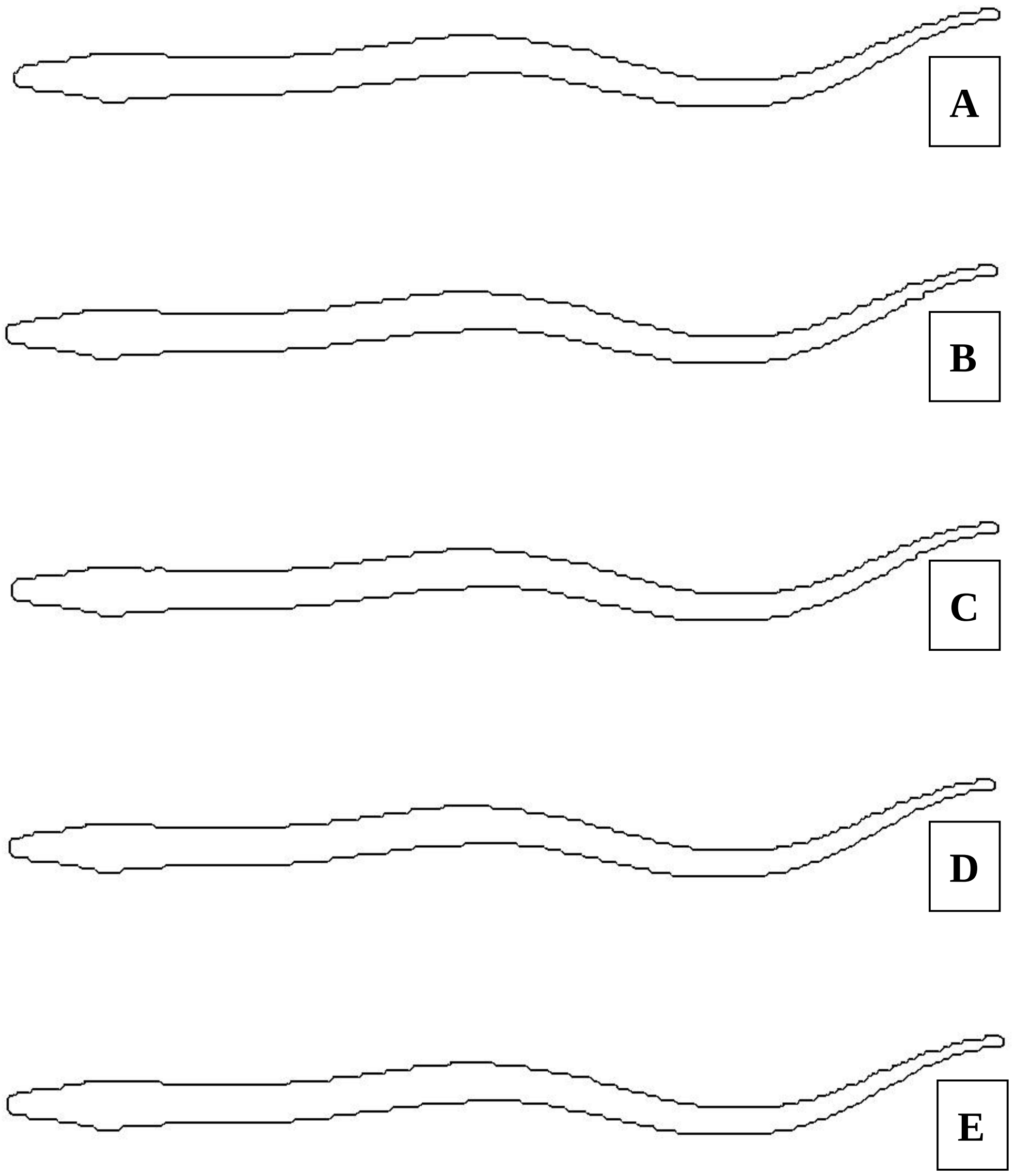


Figure 3-10. Segments resulting from filtering images by removing wavelet scales, corresponding to the images in Figure 3.9. The differences are subtle, but they are there, particularly in the tail region.

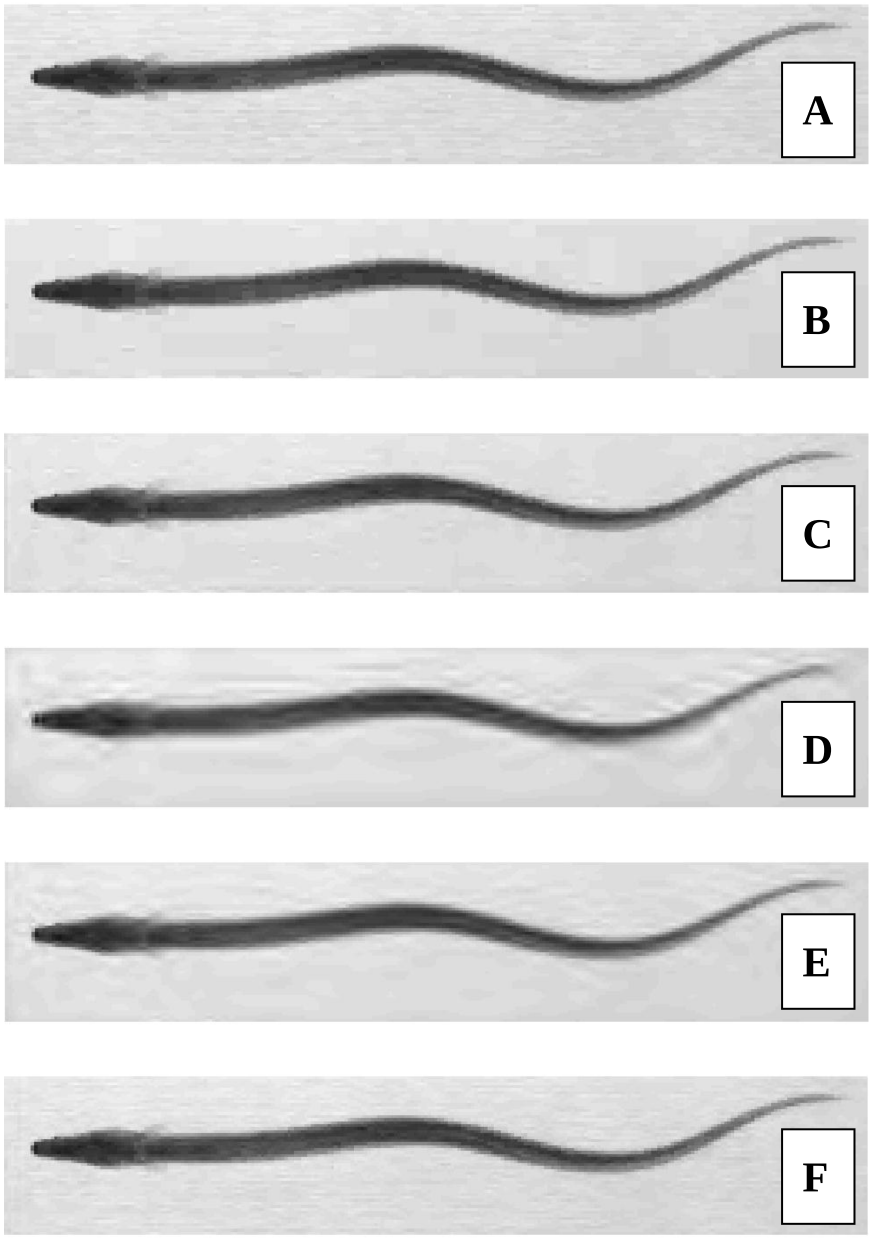


Figure 3-11. The American eel image with energy removed using different wavelet shapes: Haar wavelet with 1.0% energy removed (B), Daubechies-4 with 1.0% removed (C), Daubechies-12 with 0.01% removed (D), Daubechies-20 with 0.005% removed (E), and Daubechies-20 with 0.001% removed. The original image is shown in (A) for comparison.

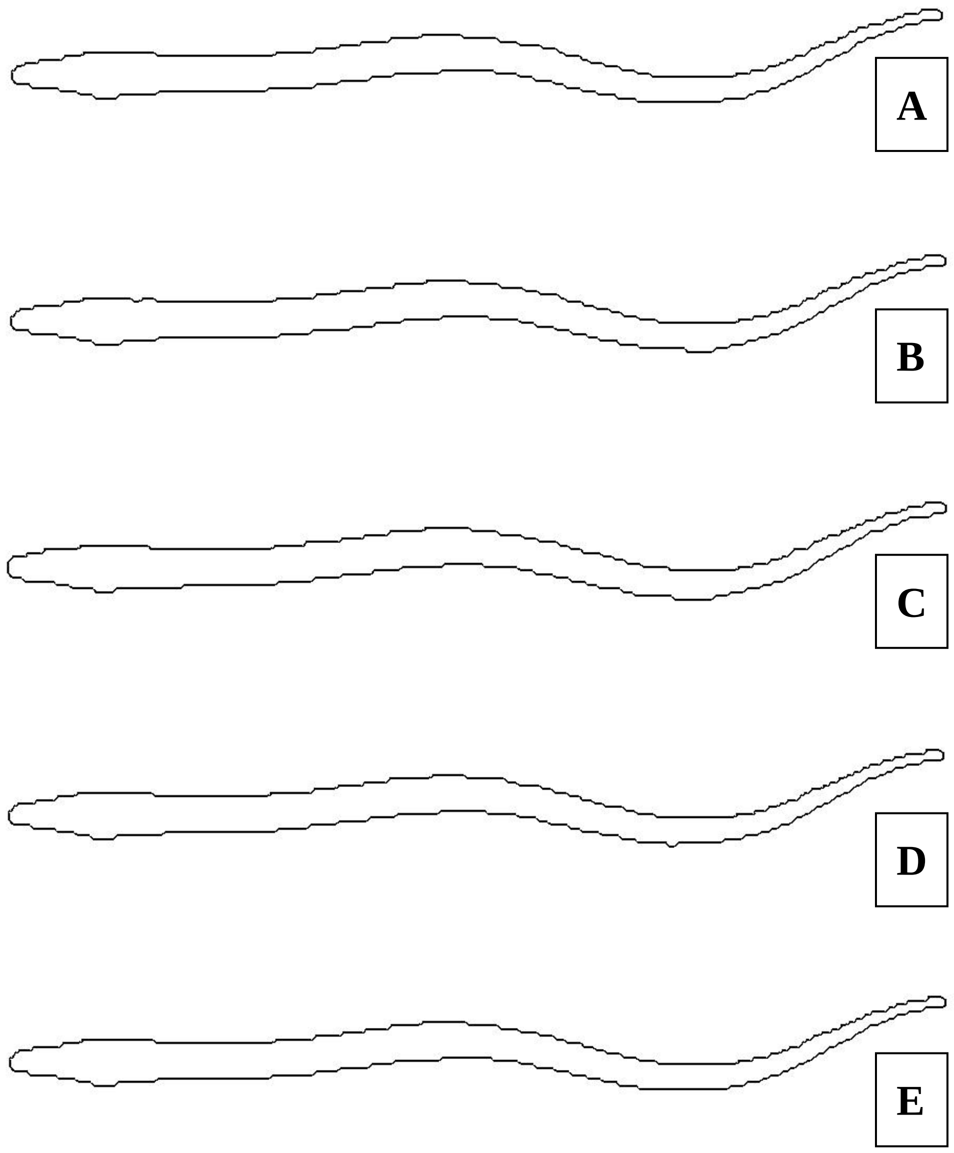
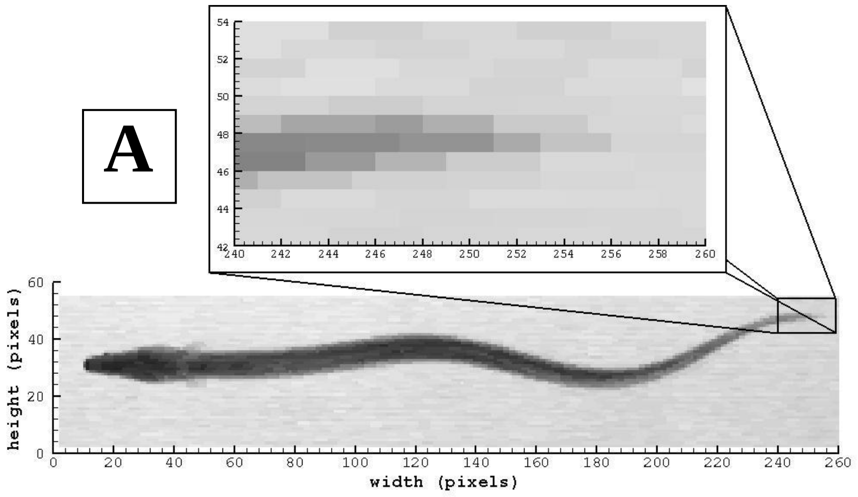


Figure 3-12. Haar 1% (A), Daub4 1% (B), Daub12 0.01% (C), Daub20 0.005% (D), Daub20 0.001% (E). The Daubechies-20 wavelet with 99.99% of the total energy retained (E) was found to give the most consistent segmentation results for all of the simulations performed in this work.



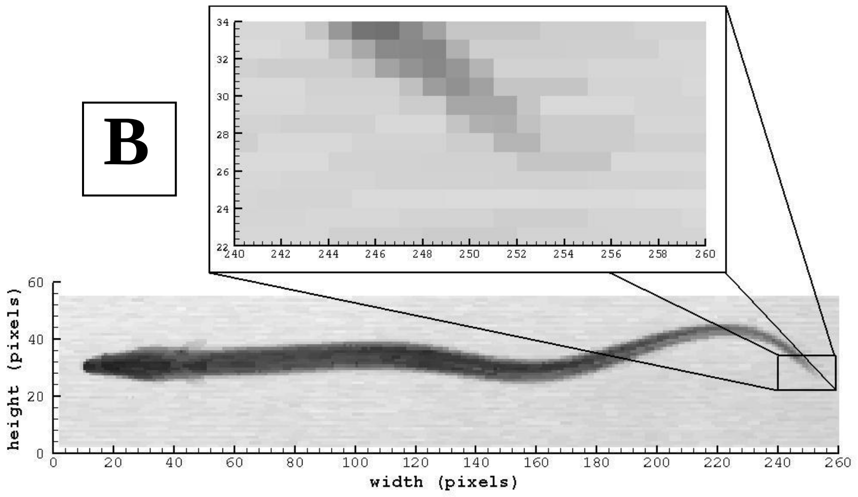


Figure 3-13. Poor contrast in the eel's tail made this region particularly susceptible to inconsistencies in the final segmented shape between any two image frames. Shown are tail positions in the first frame (A) and 28 frames later (B) in the swimming sequence. The problem was mollified by "chopping off" the tail: image frames were cropped at $\boldsymbol{x = 245}$ pixels and the tip of the tail was thus eliminated from view. Note that this has the undesirable effect of changing the eel's overall length as it swims.

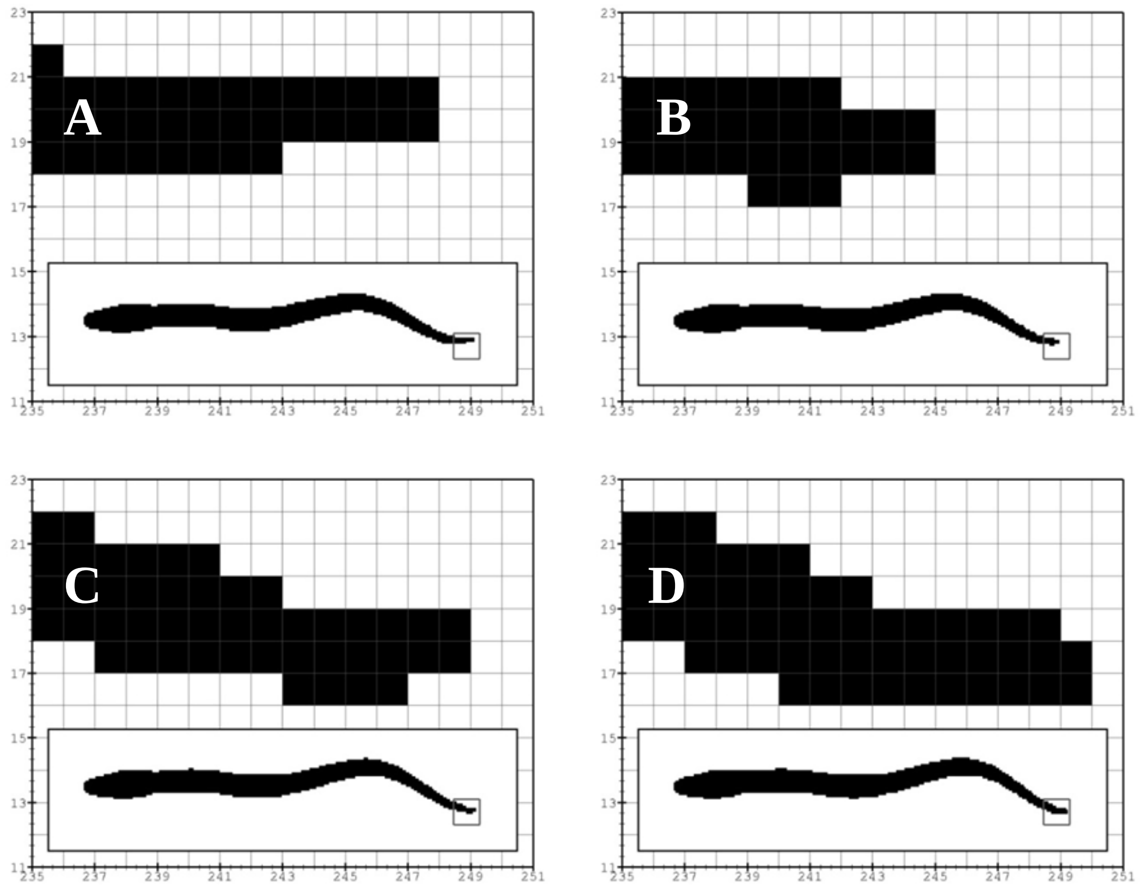


Figure 3-14. Jumps in the segmented tail tip location, caused by poor contrast in this region of pixels, were eliminated by “chopping off” the tail at $\boldsymbol{x} = \mathbf{245}$ prior to populating the surface with Lagrangian points. Shown are 4 sequential segmented image frames (18 through 21) out of a set of 36 frames representing one complete tail beat cycle. The X-location of the tail tip on the pixel grid has values 248.0 (A), 245.0 (B), 249.0 (C), and 250.0 (D), potentiating unrealistic flow field calculations brought on by spurious interfacial movements—particularly troublesome in a region so important in governing vortical transport behavior.

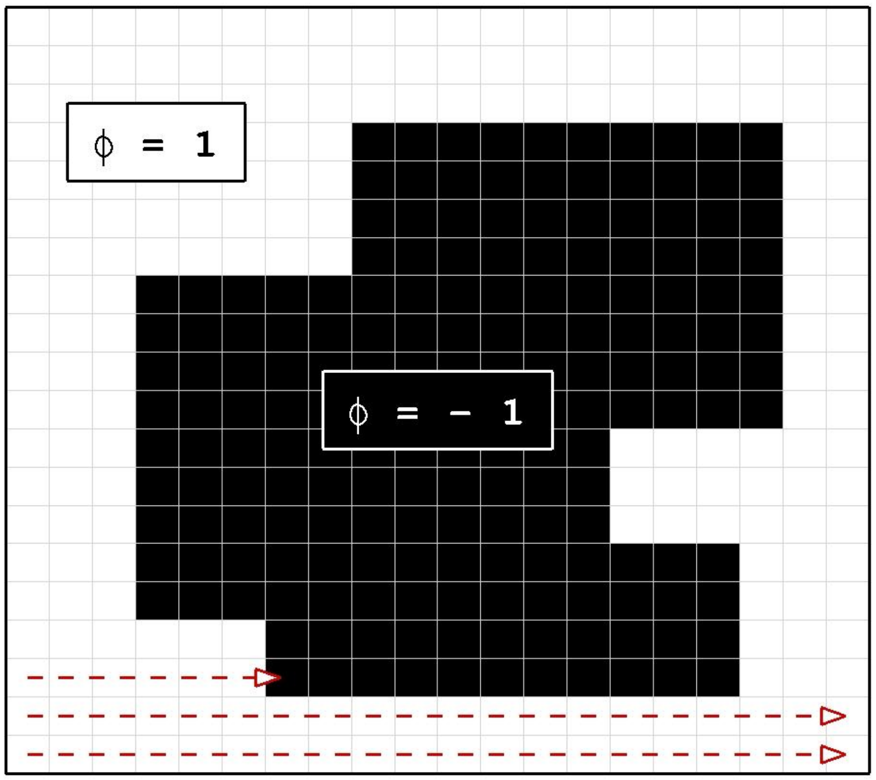


Figure 3-15. In the general algorithm, searching proceeds until the first pixel is found that has a value which is different from that of the starting pixel (the pixel at the lower left corner, in this case).

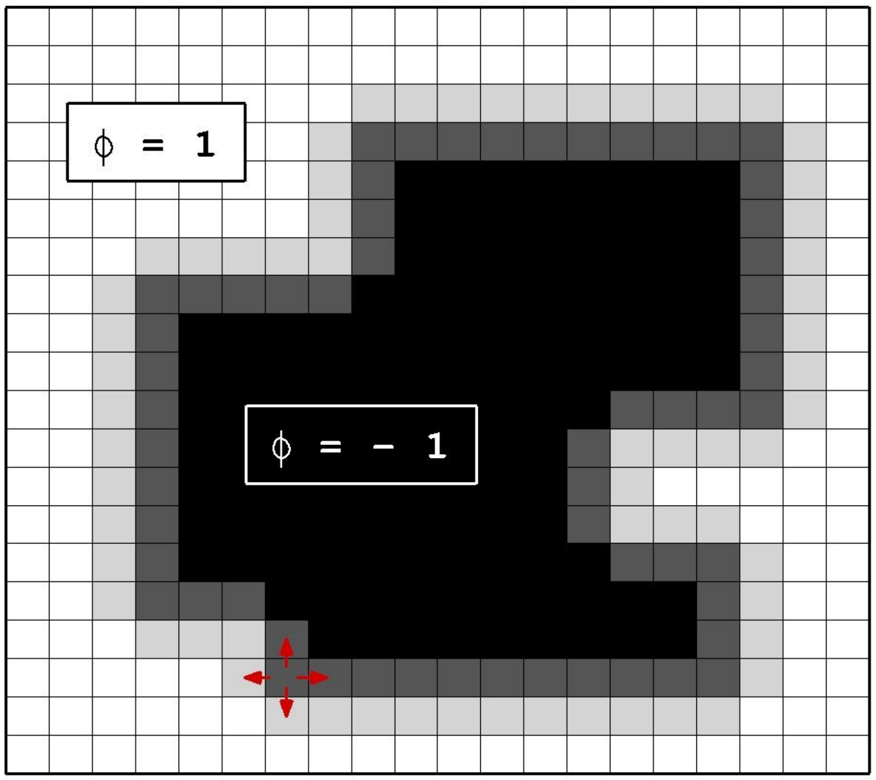


Figure 3-16. The neighbors of each cell are searched, and their field value and interfacial cell status is recorded. In this figure, interfacial cells inside the object are colored dark grey; interfacial cells outside the object are colored light grey. Each cell is tagged once it has been visited by the searching algorithm, so that it will not be visited again.

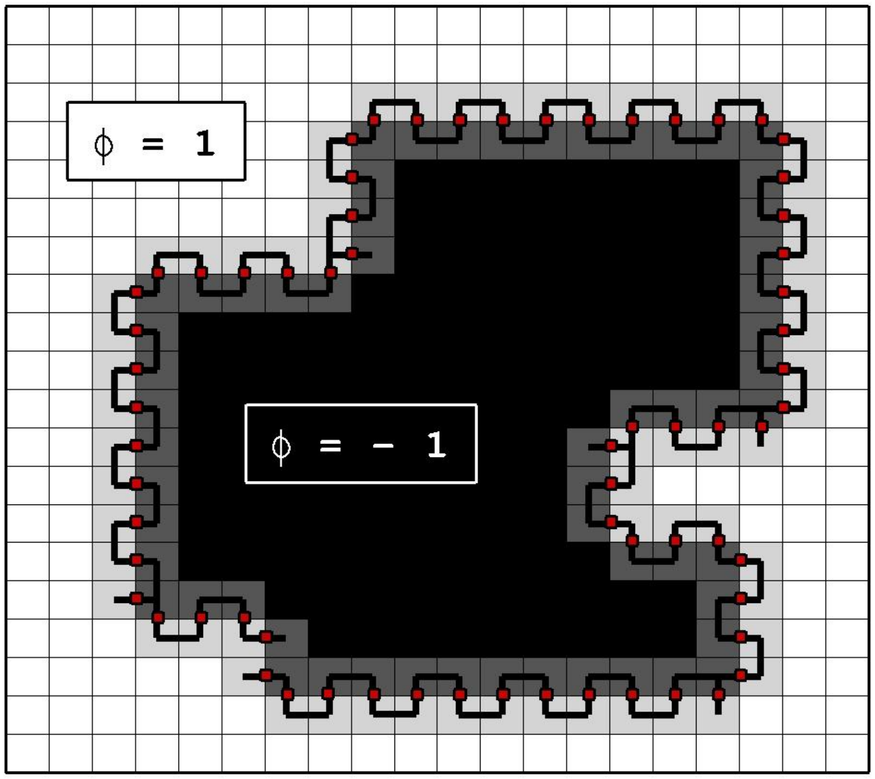


Figure 3-17. The search path; interfacial locations are stored whenever a change in field value is detected.

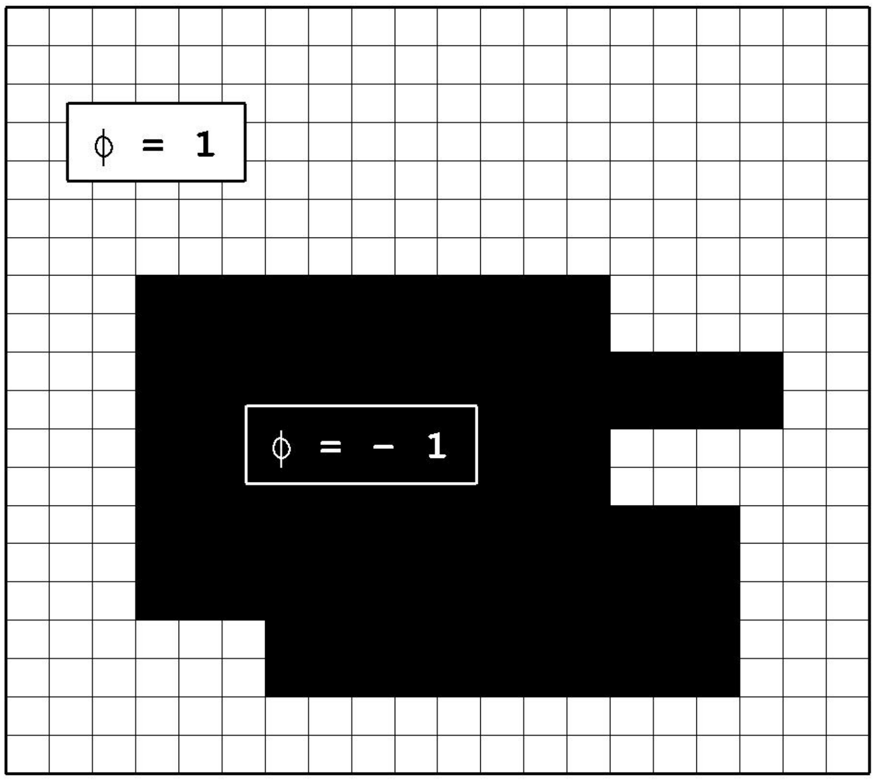


Figure 3-18. A geometric configuration that will promptly kill the general search algorithm outlined here. Inner and outer regions both feature protrusions that are two pixels in width, a situation this algorithm cannot handle.

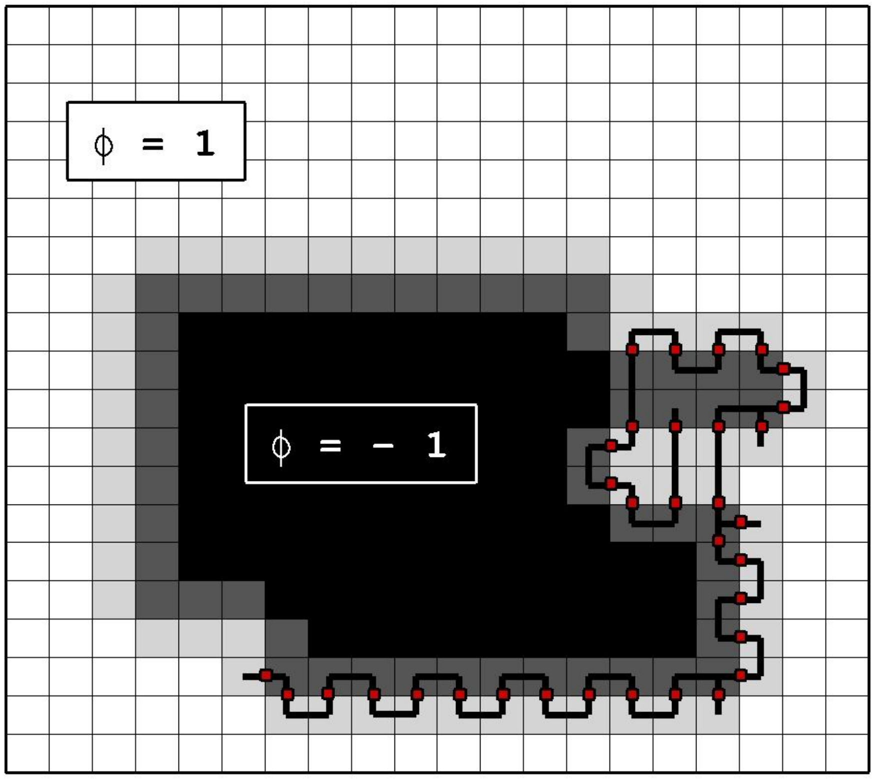


Figure 3-19. The troublesome geometry outlined in the previous figure, with interfacial cells and the failed search path illustrated.

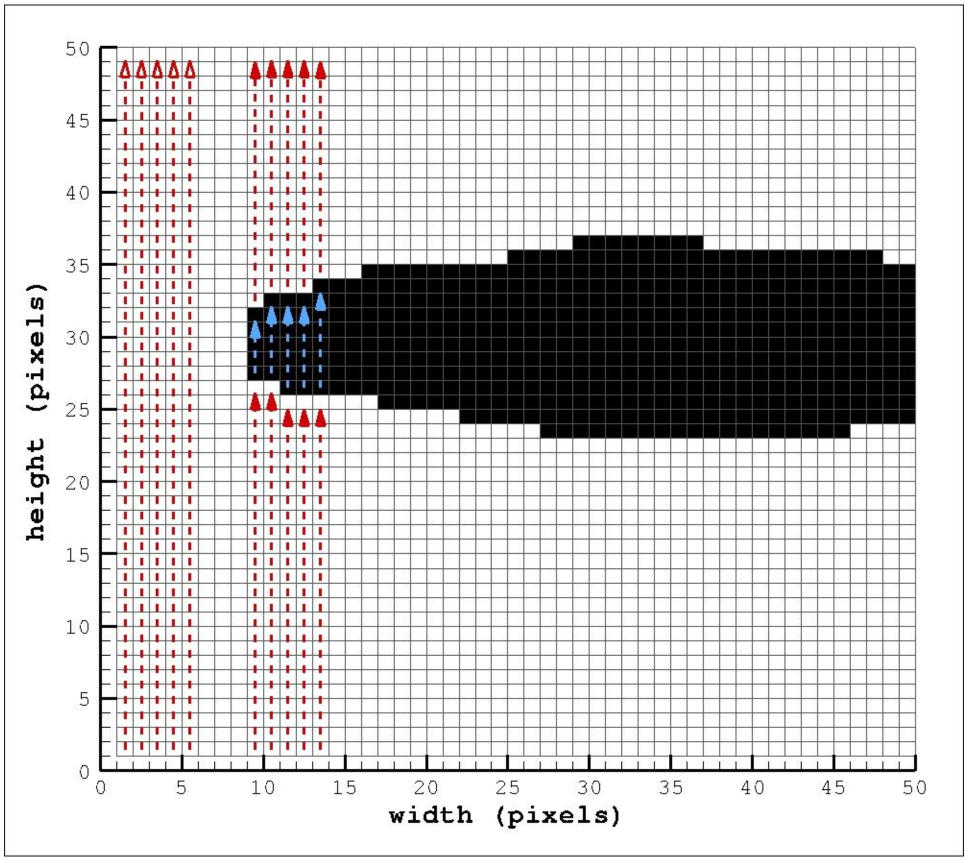


Figure 3-20. In order to avoid the potential problems of the generalized searching algorithm, searching is performed in the direction perpendicular to the segmented object's dominant length direction (the Y-direction was searched for both the eel and duodenum), in order to find the locations of interfacial pixels in the binary field and populate the segmented boundary with Lagrangian surface points. The eel's head is shown here for illustration of the process.

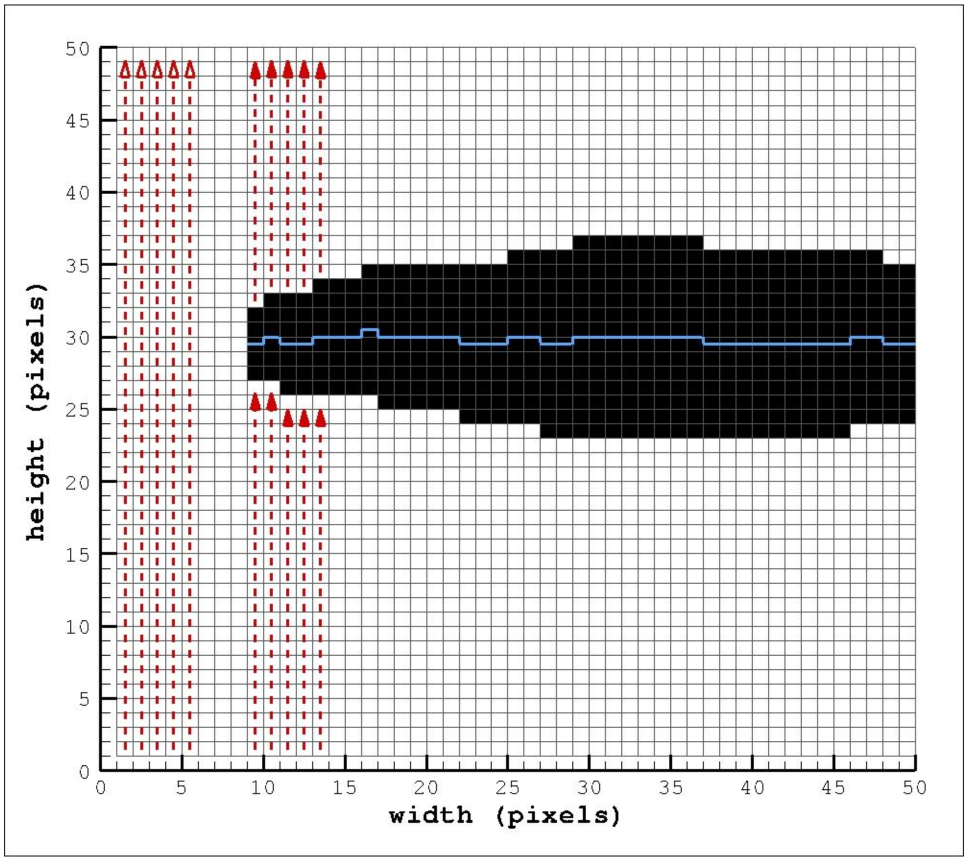


Figure 3-21. The centerline location of each column of segmented pixels was stored for each of the eel image frames, then smoothed using a Savitzky-Golay filter with a window size of 30 points. Upper and lower surface locations were subsequently shifted to maintain their original distance from the smoothed centerline. For images from the duodenal experiment, the upper and lower surfaces of the channel were treated independently, smoothing each with a Savitzky-Golay filter and eliminating the centerline step.

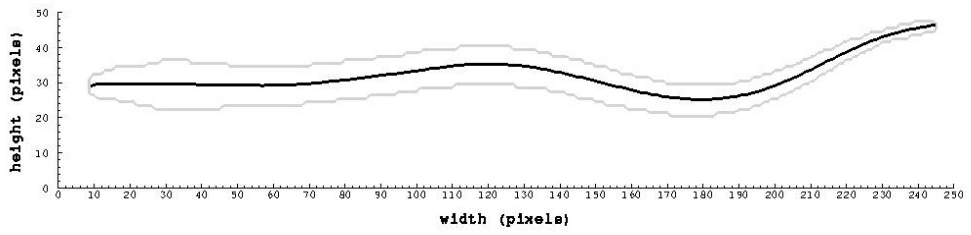


Figure 3-22. The smoothed centerline of the American eel overlaid on the original segmentation contour for the first image frame.

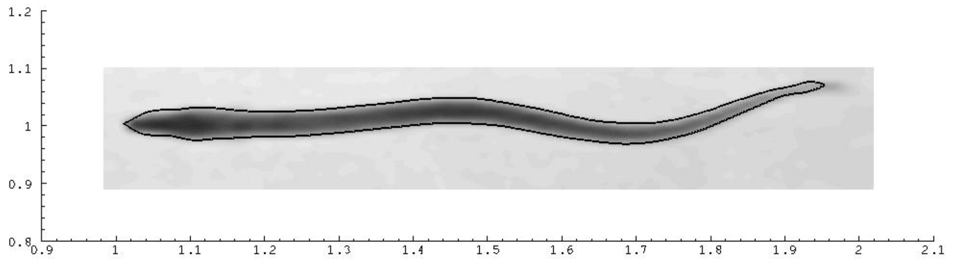


Figure 3-23. Smoothed points mark the boundary of the American eel, ready for conversion to a zero level set contour in the Eulerian flow domain. After the original upper and lower surface points were shifted with respect to the smooth centerline, they were re-spaced to produce even segment lengths between them and populated with additional intermediate points to avoid an undesirably sparse surface description on the fine flow calculation mesh. Further smoothing was performed using a Daubechies 20 wavelet transform and removing contributions from the four smallest (highest resolution) wave numbers. Finally, the surface points were spatially scaled and shifted to produce a body of near unit length, positioned for convenient domain construction.

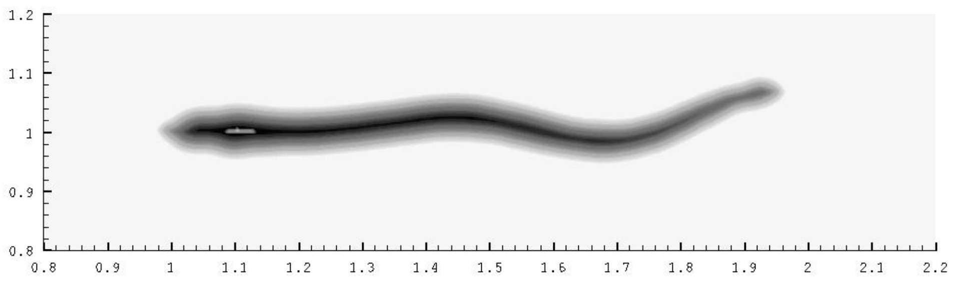


Figure 3-24. A plot of the narrow band level set field for the first fiducial time step of the American eel calculations.

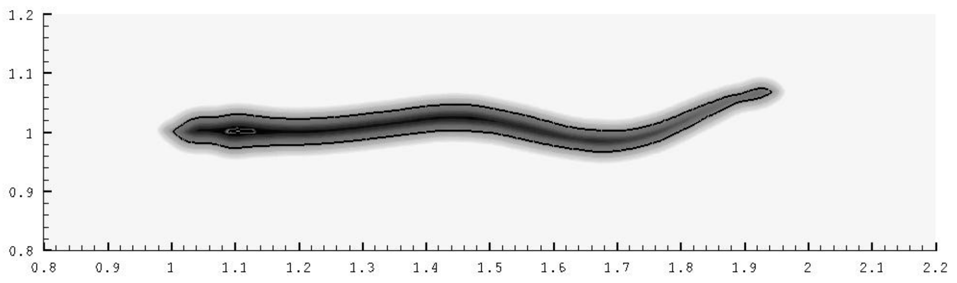


Figure 3-25. A plot of the narrow band level set field for the first fiducial time step of the American eel calculations, with the zero-level set contour shown for clarity.

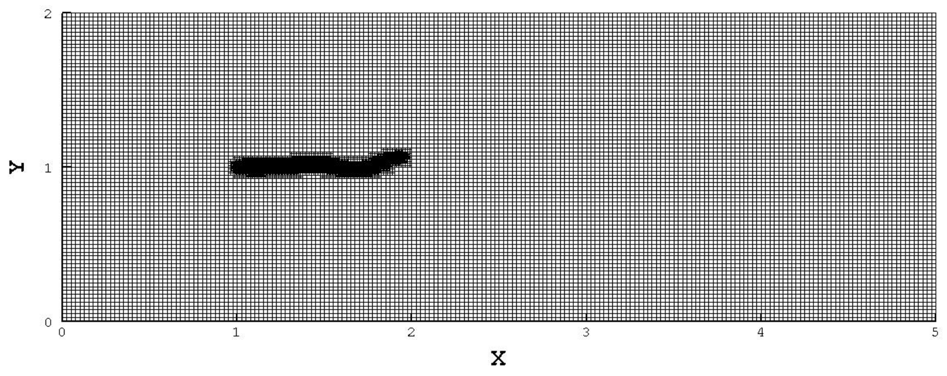


Figure 3-26. Illustration of the initial computational domain for all of the eel calculations. The mesh is refined 3 levels around the eel’s body, left coarsened throughout the rest of the domain for computational efficiency until further refinement is needed for resolution of flow structures.

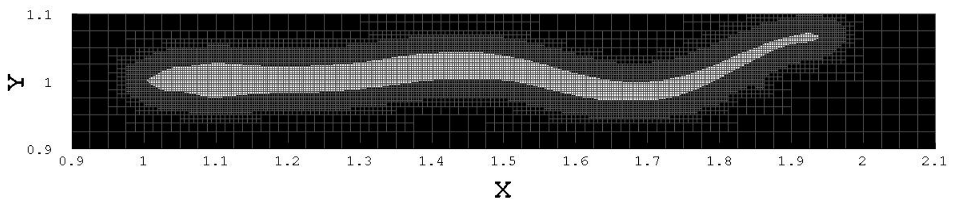


Figure 3-27. Zero levelset contour embedded within the flow mesh, showing adaptive grid refinement near the interfacial boundary.

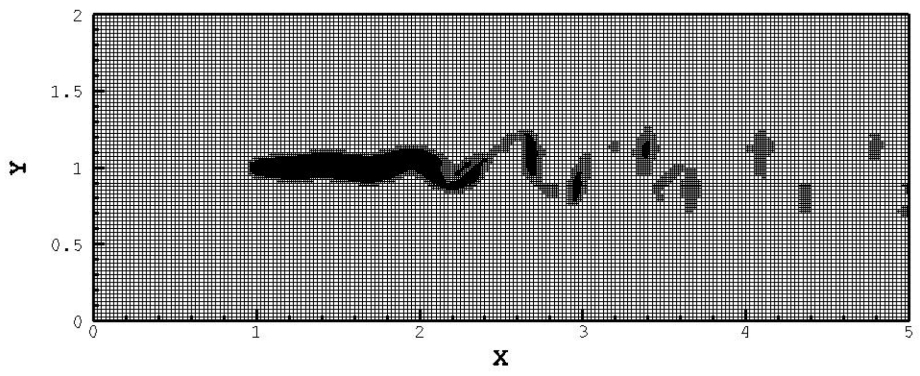


Figure 3-28. The computational mesh for the swimming eel: Mesh refinement captures regions of high velocity gradient with greater fidelity, while saving computational time by leaving the mesh coarse elsewhere.

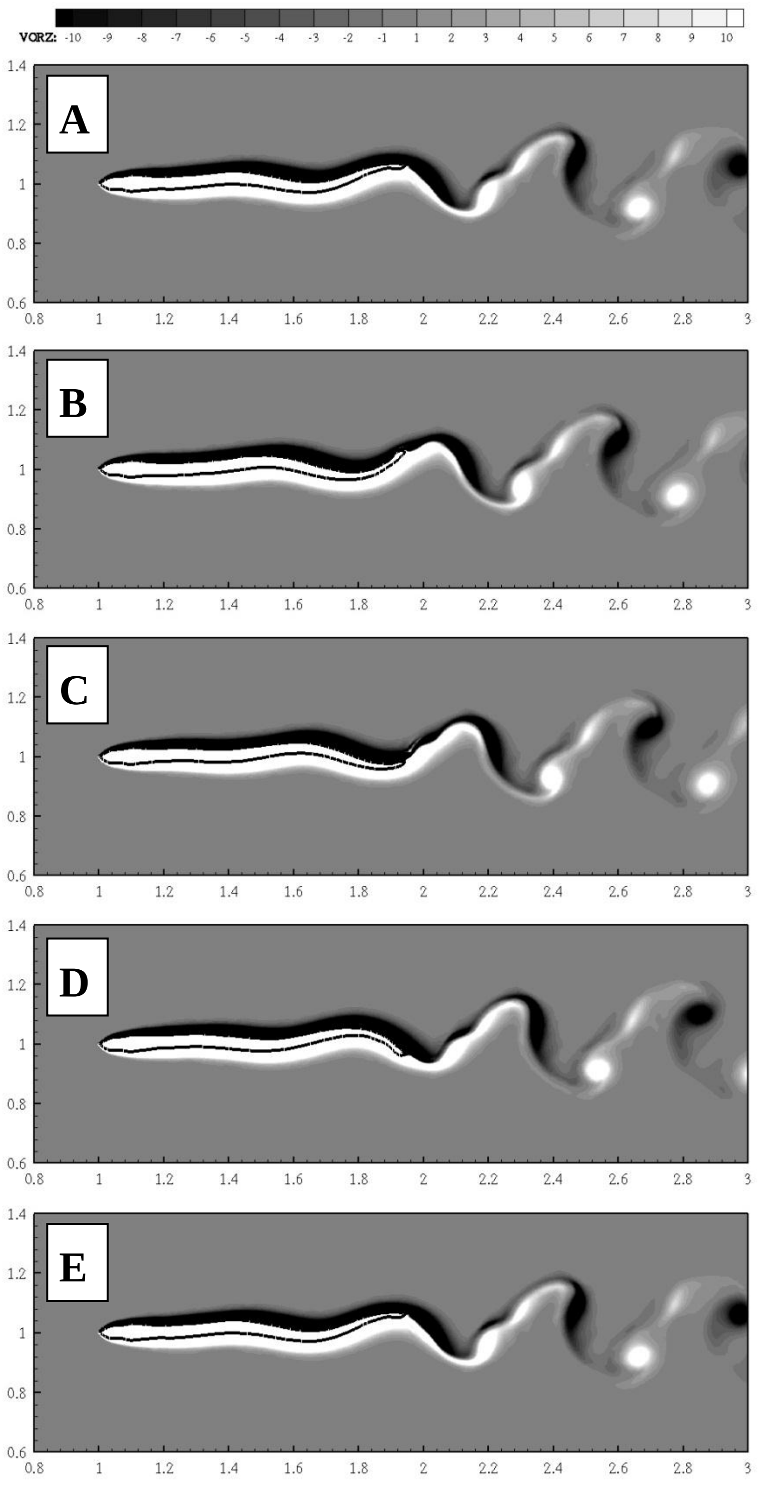


Figure 3-29. Contours of vorticity plotted through one tail beat. St = 0.3; Re = 5000.

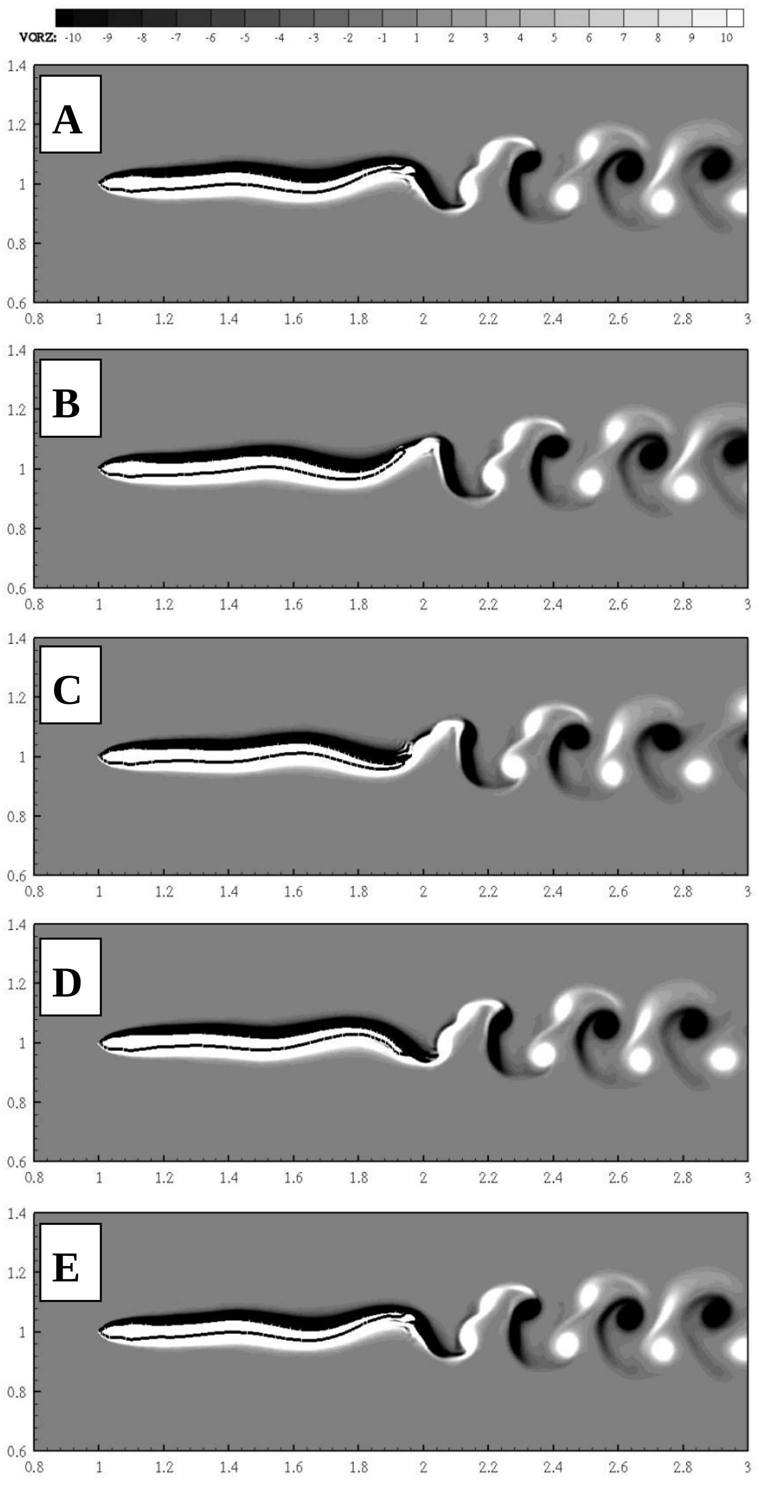


Figure 3-30. Contours of vorticity plotted through one tail beat. St = 0.5; Re = 5000.

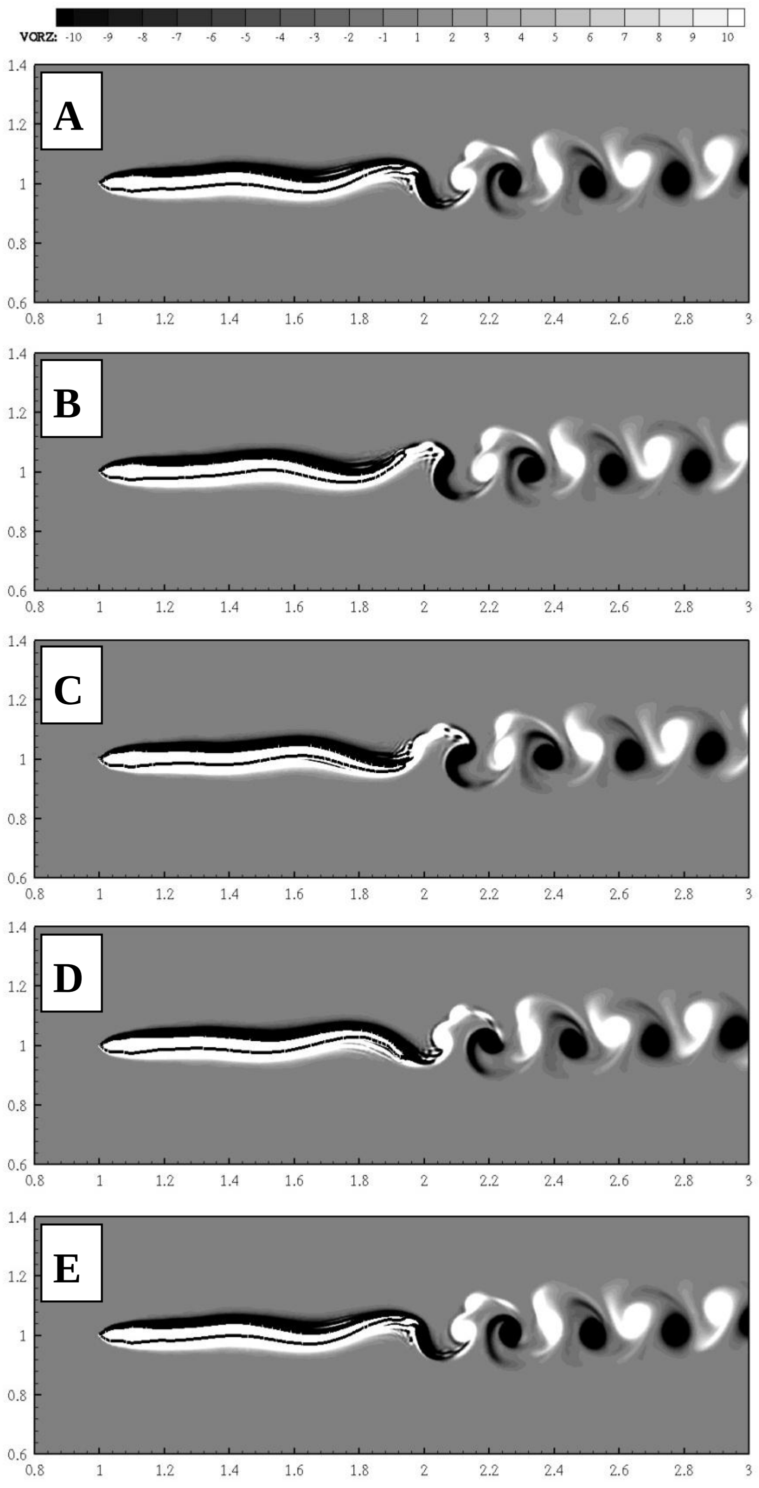


Figure 3-31. Contours of vorticity plotted through one tail beat. St = 0.7; Re = 5000.

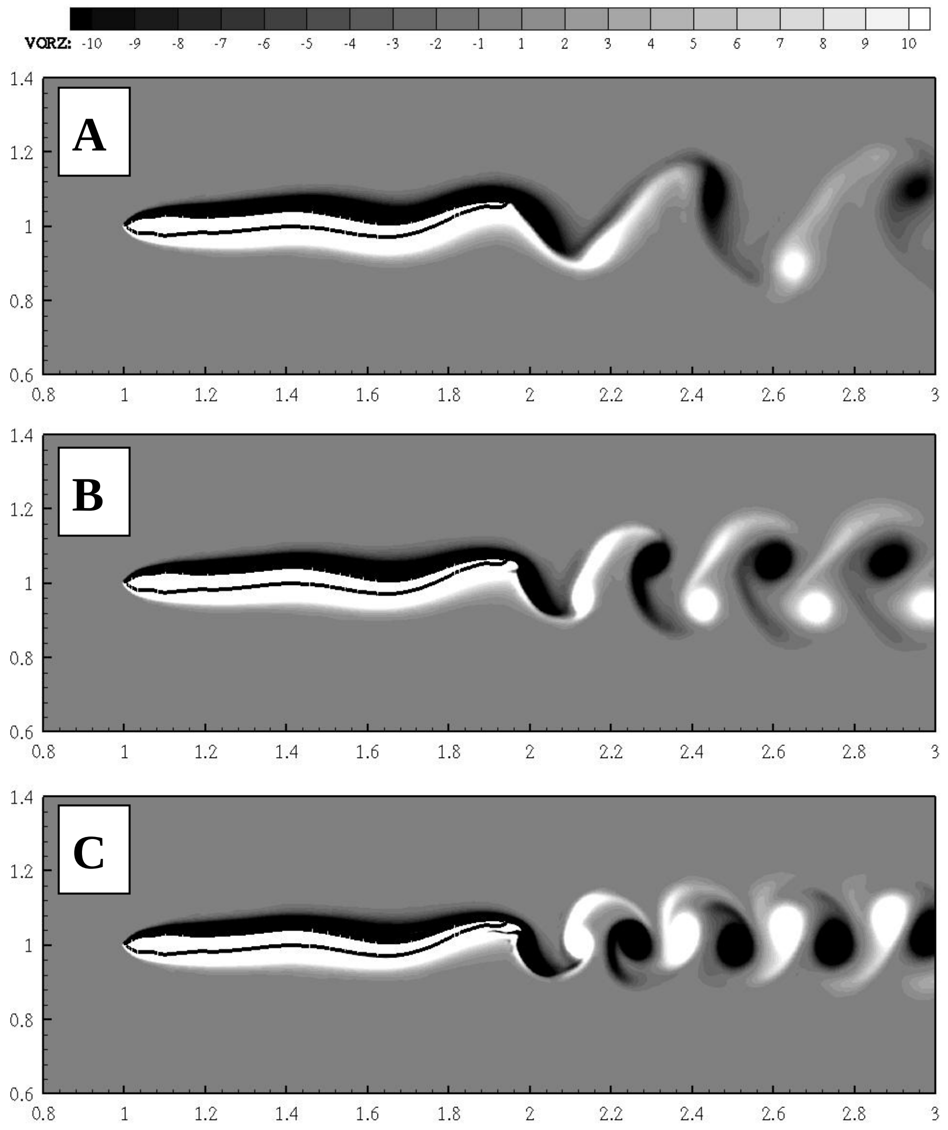


Figure 3-32. Contours of vorticity at 3 different Strouhal numbers with Re = 2500: (A) St = 0.3, (B) St = 0.5, (C) St = 0.7.

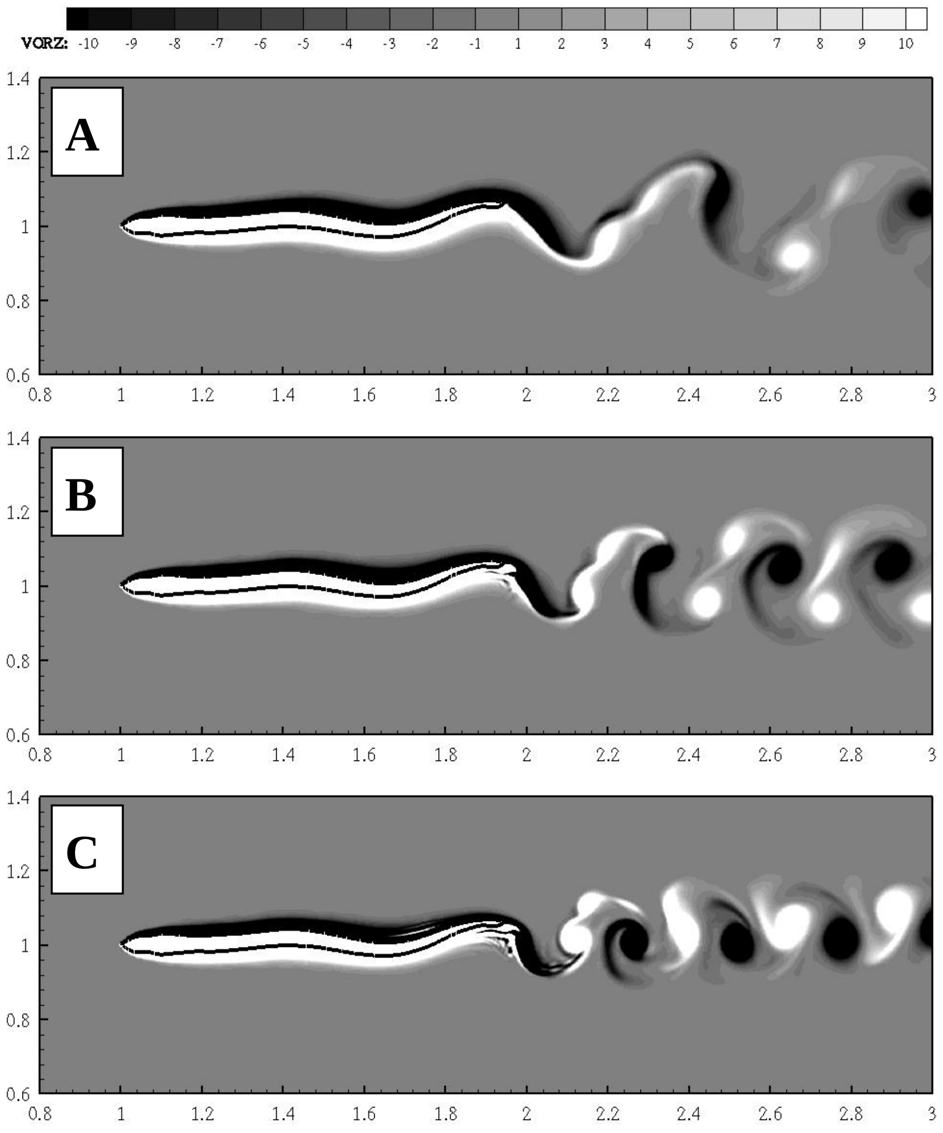


Figure 3-33. Contours of vorticity at 3 different Strouhal numbers with Re = 5000: (A) St = 0.3, (B) St = 0.5, (C) St = 0.7.

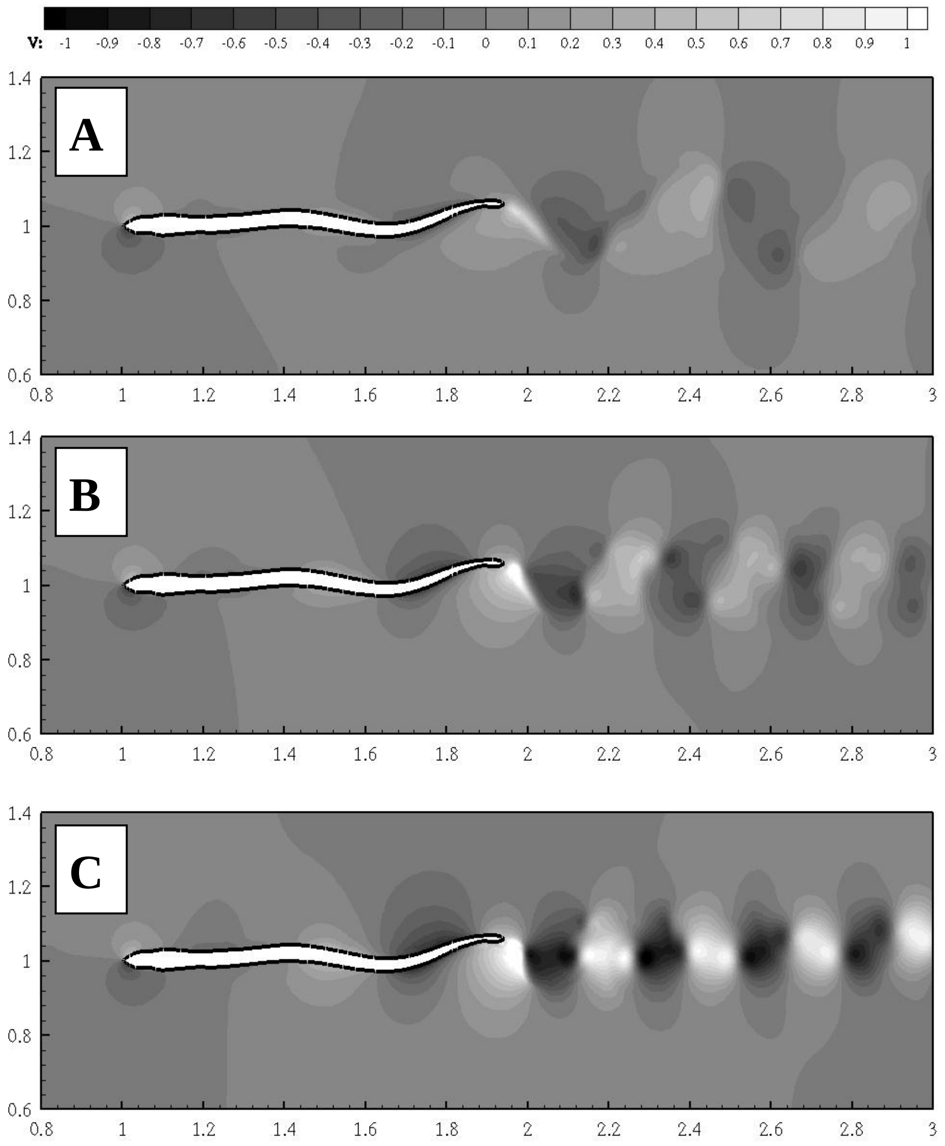


Figure 3-34. Lateral component of velocity at the 3 Strouhal numbers evaluated, for Re = 5000: (A) St = 0.3, (B) St = 0.5, (C) St = 0.7.

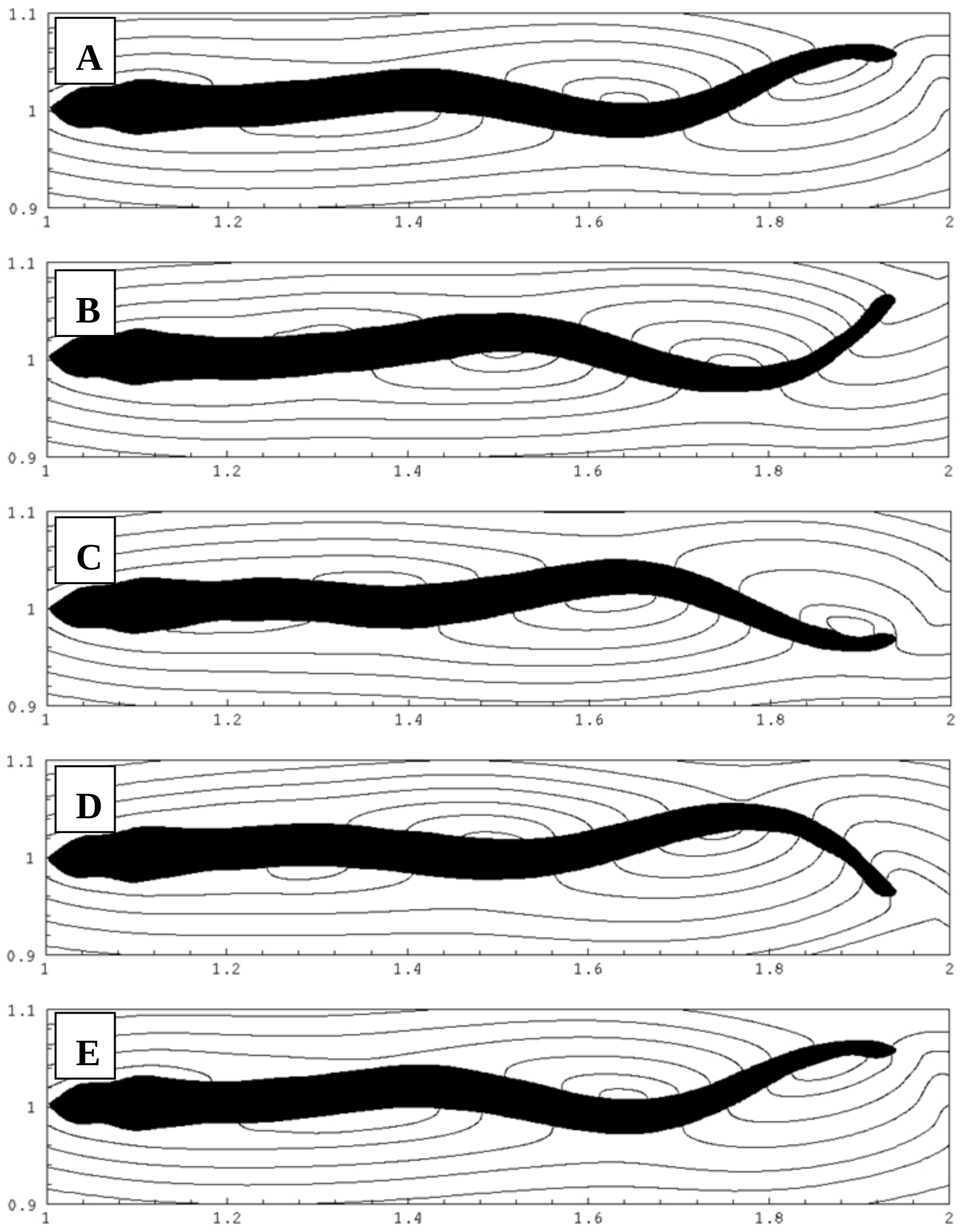


Figure 3-35. Streamlines plotted through one complete tail beat; St = 0.7, Re = 5000. The streamlines appear nearly identical for all cases, exhibiting “boluses” of fluid being transported along the eel’s body in wave troughs as it swims.

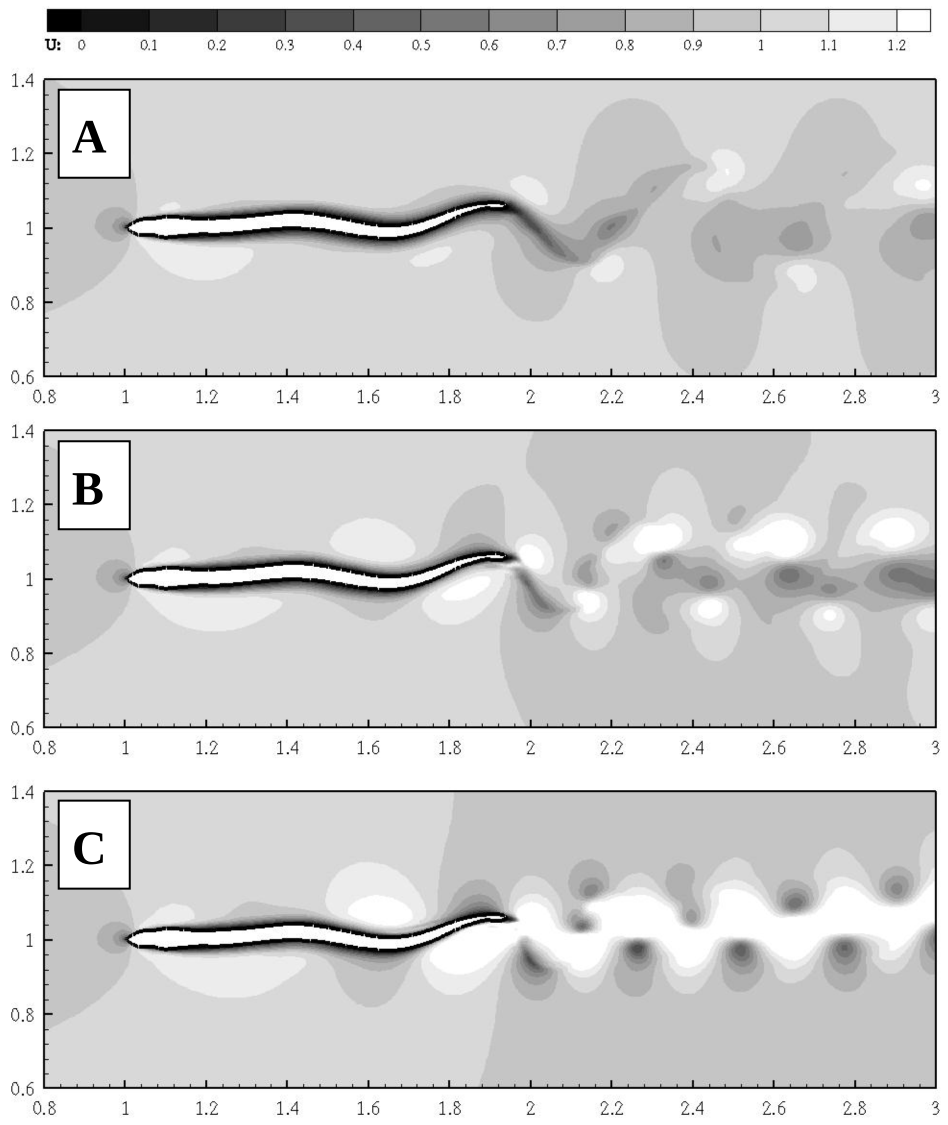


Figure 3-36. Axial component of velocity at the 3 Strouhal numbers evaluated, for Re = 5000: (A) St = 0.3, (B) St = 0.5, (C) St = 0.7. The eel's anguilliform swimming motion produces a set of vortical structures from which momentum is primarily ejected laterally; thus, a clear thrust generation signature such as that found in the wakes of carangiform swimmers is absent here.

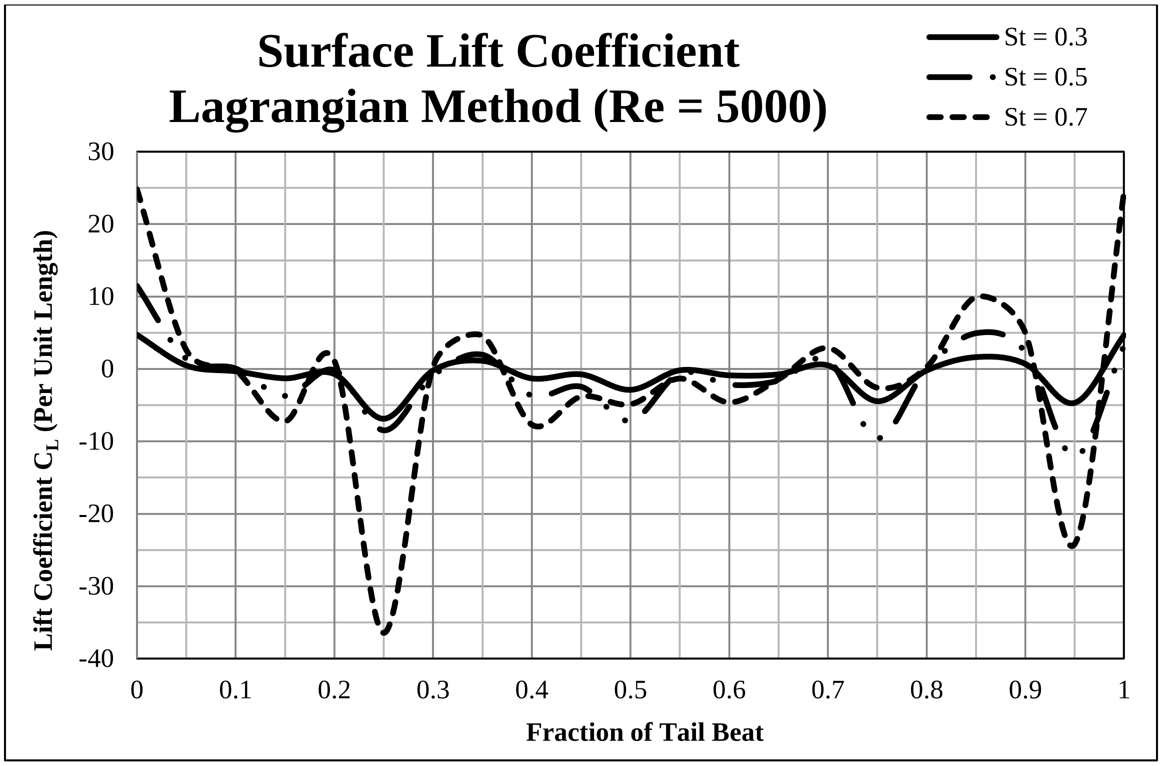


Figure 3-37. Lift coefficient on the eel’s body plotted through one tail beat; Re=5000

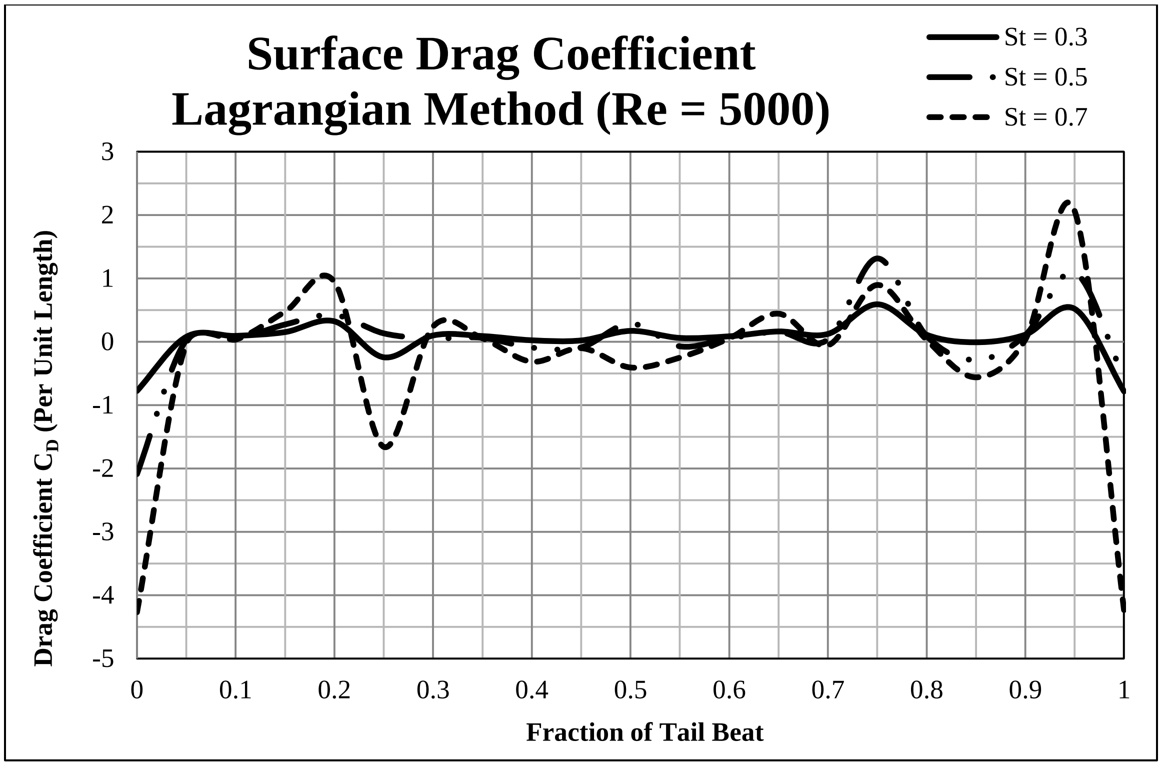


Figure 3-38. Drag coefficient on the eel's body plotted through one tail beat; Re=5000

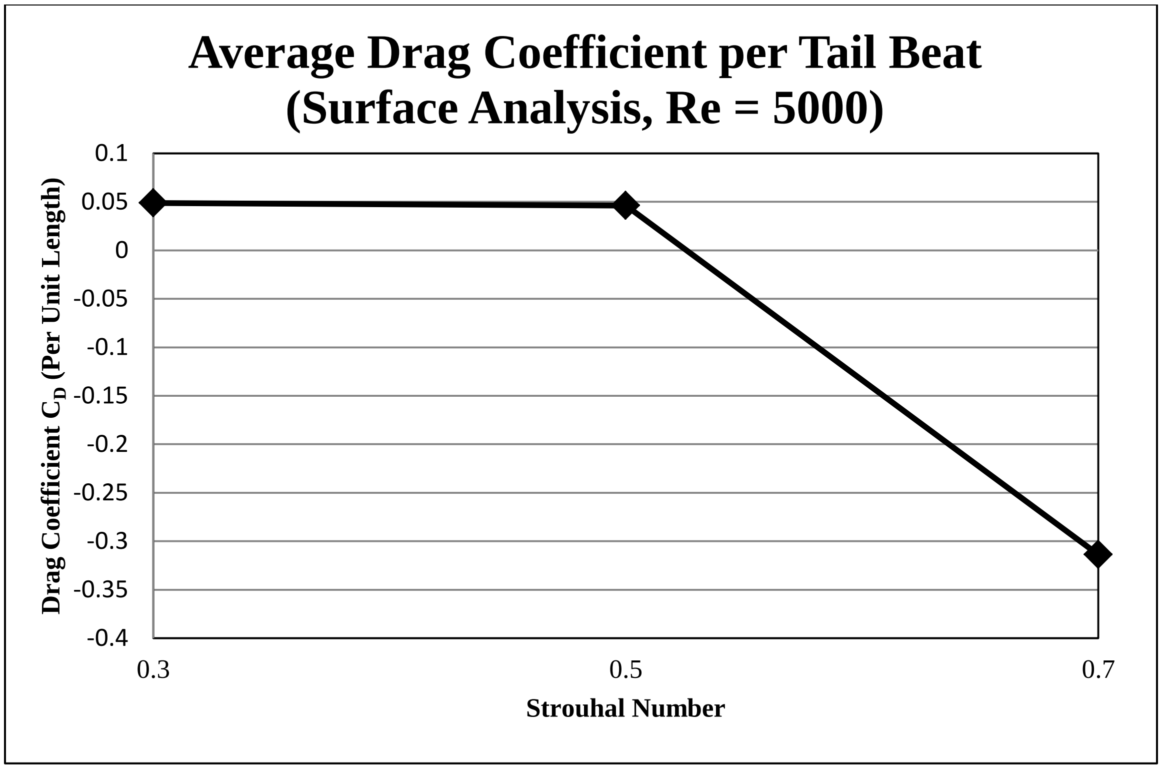


Figure 3-39. Average surface drag coefficient during one tail beat; Re = 5000.

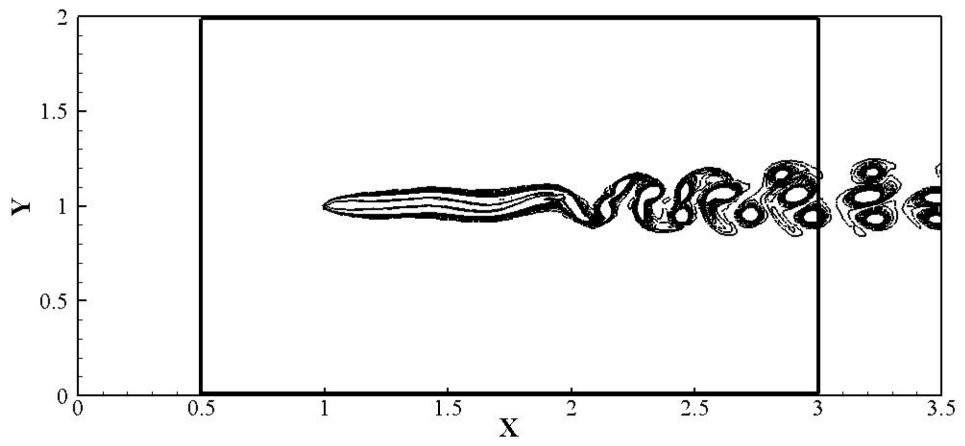


Figure 3-40. A control volume was defined in the domain for analyzing thrust/drag, spanning from 0.5 units length upstream to 1 unit length downstream of the swimming eel in the horizontal direction, and spanning the entire domain in the vertical direction.

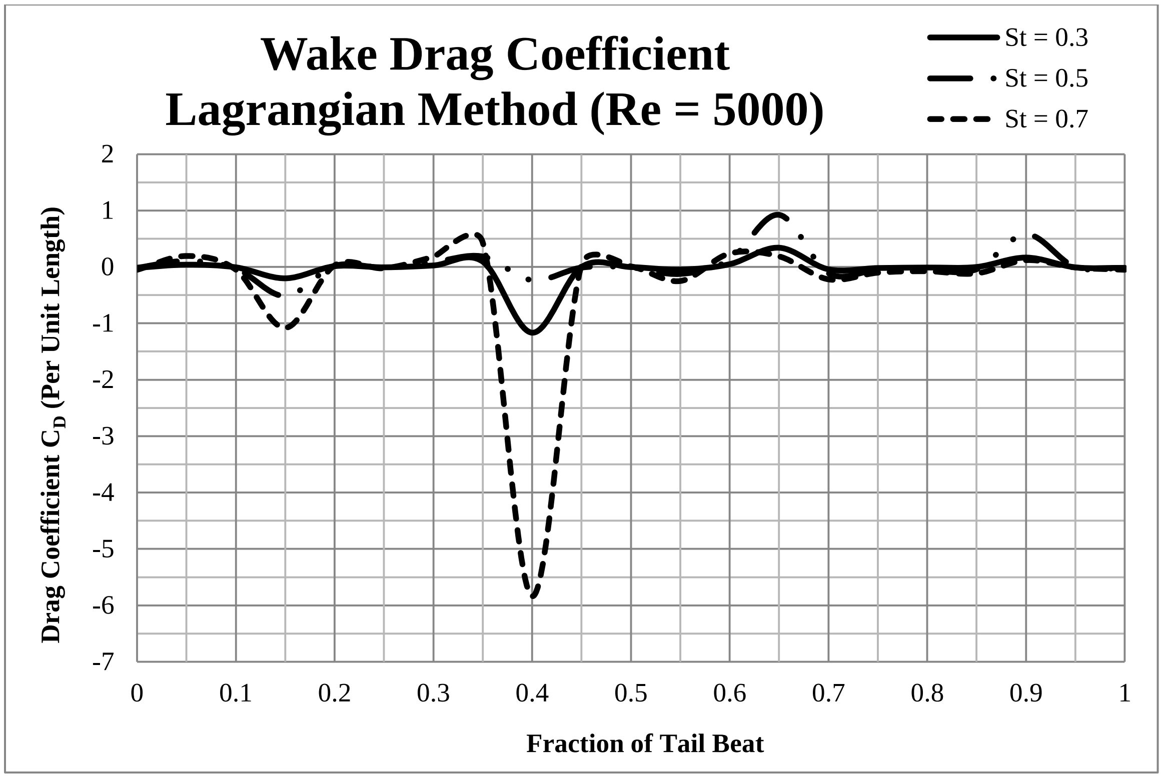


Figure 3-41. Drag coefficient measured in the eel’s wake plotted through one tail beat; Re = 5000.

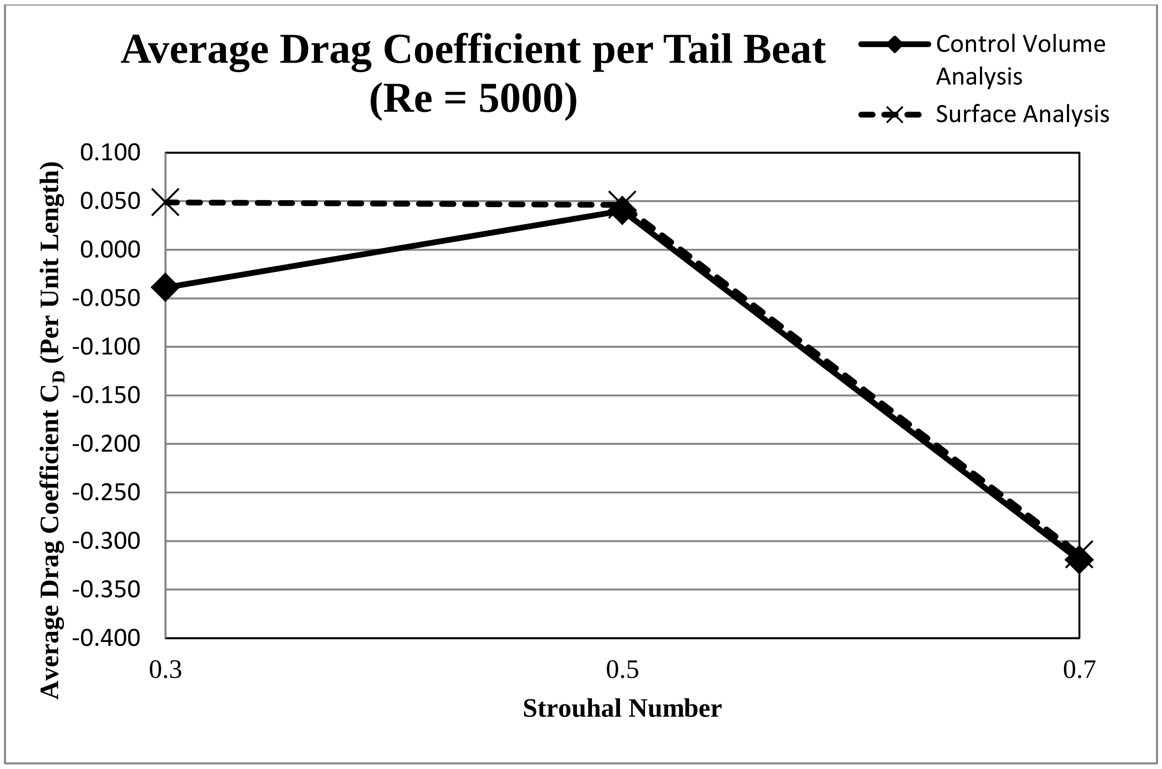


Figure 3-42. Average drag coefficient during one tail beat (control volume and surface analyses); Re = 5000.

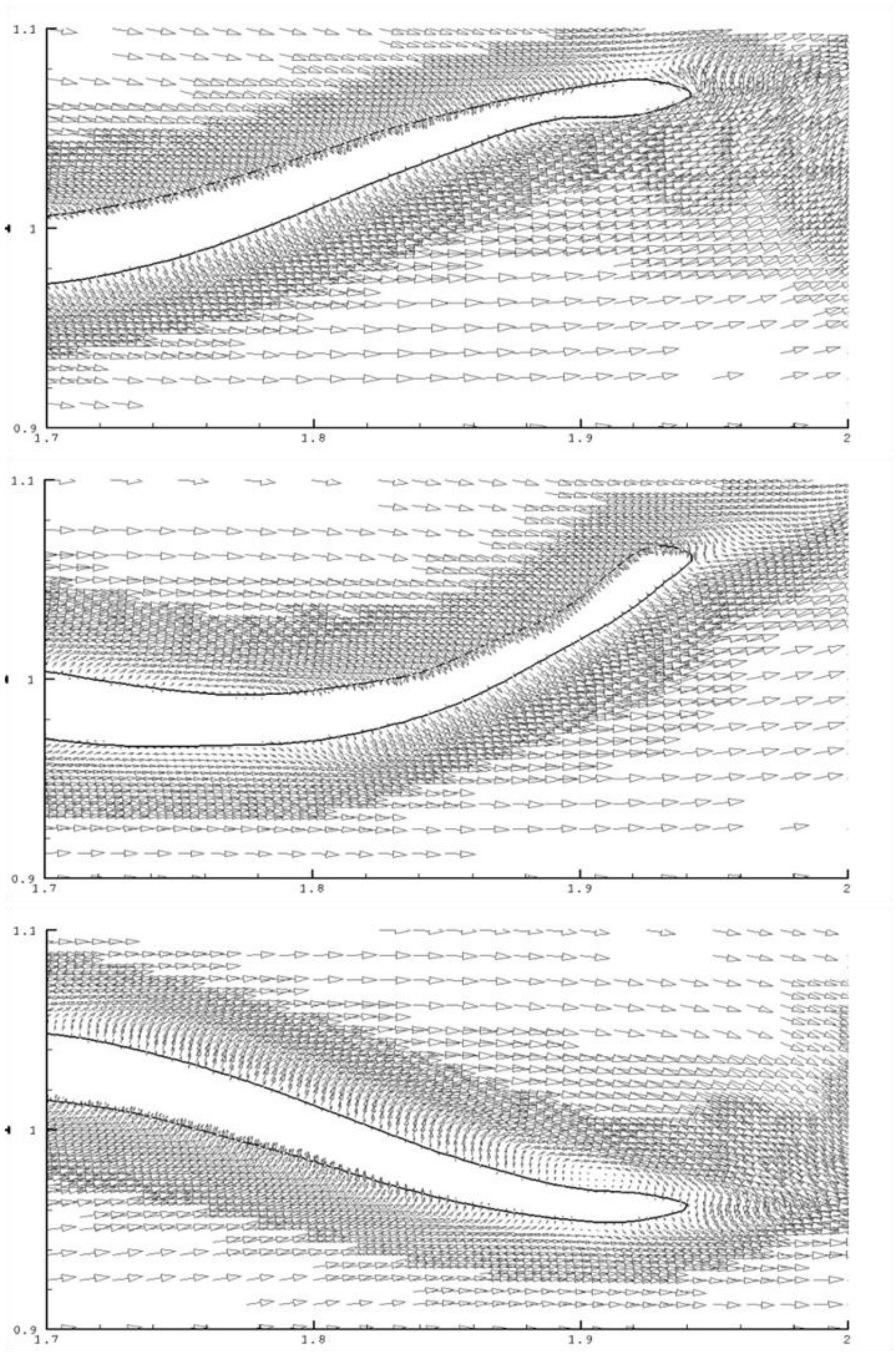


Figure 3-43. The changing shape of the eel's tail, combined with its rounded tip, may lead to a lack of fidelity in generating the correct wake morphology in these simulations.

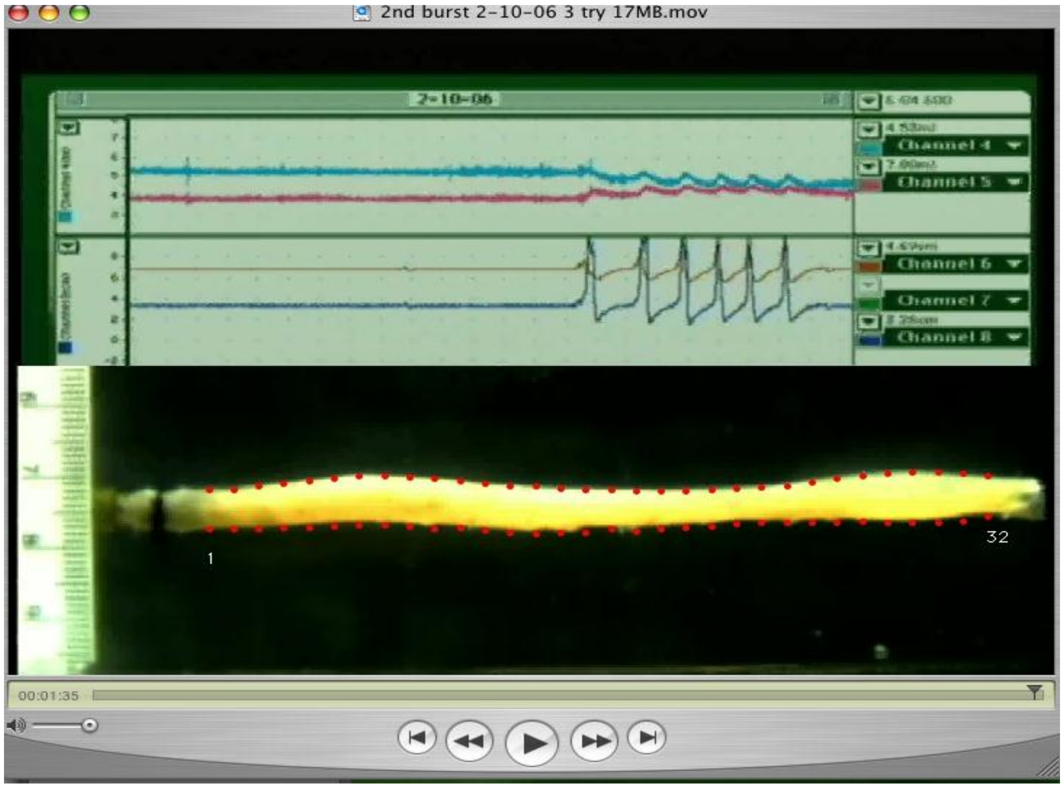


Figure 3-44. Screen shot of the Motility Mapping program used by Dr. Schulze to obtain the duodenal segment’s diameter at 32 locations along its length (provided by Dr. Schulze’s research laboratory).

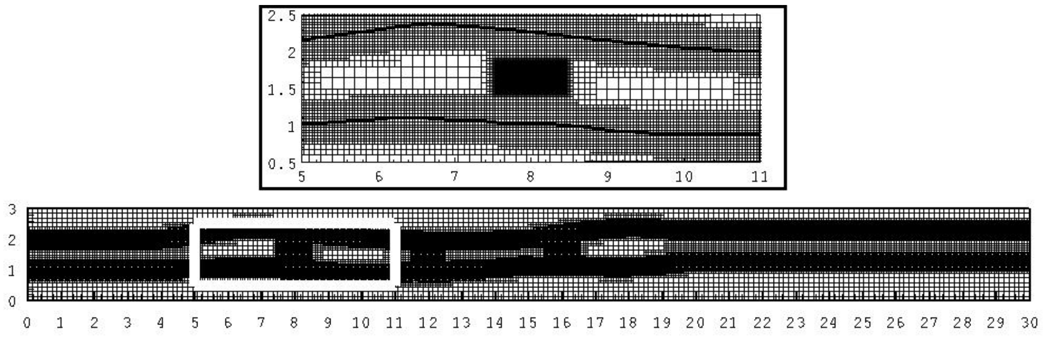


Figure 3-45. The duodenum channel domain is 30 units length by 3 units height with a maximum grid spacing of $\Delta\boldsymbol{x}_{\boldsymbol{max}} = \mathbf{0.15}$. Local mesh refinement reduces this spacing up to 2 more levels where necessary to resolve boundaries, flow features, and regions containing scalar species (illustrated in the close-up view; $\Delta\boldsymbol{x}_{\boldsymbol{min}} = \mathbf{0.0375}$.

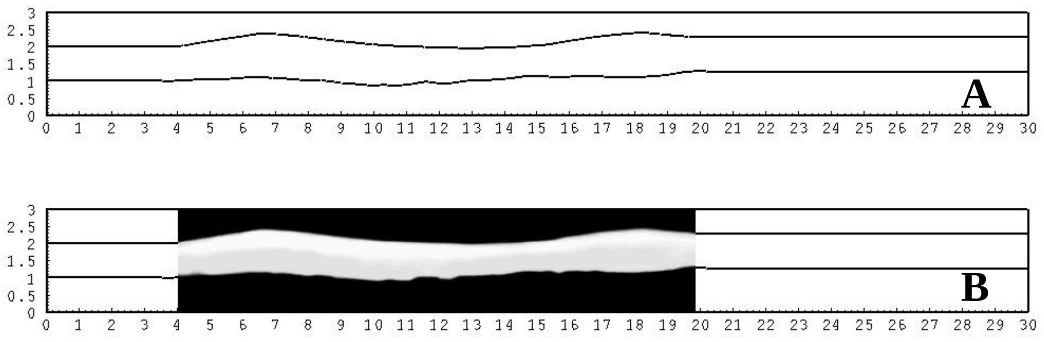


Figure 3-46. The duodenal segment channel (A) was extended beyond the inlet and exit regions of the image (B). This ensured that the inlet and exit to the domain maintained the same width, and that the scalar species blocks used to calculate mixedness would not pass out of the domain during peristaltic contractions.

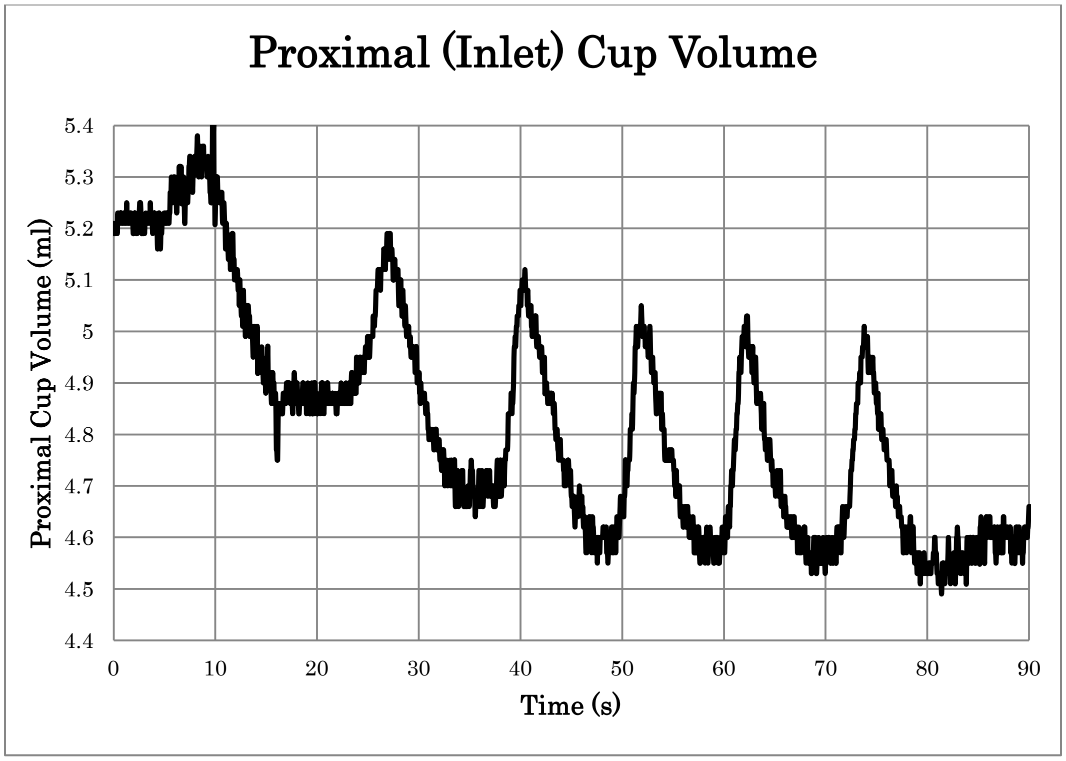


Figure 3-47. Changes in proximal cup volume, and thus flow through the experimental duodenal segment, matched temporally with changes in imaged area nearly exactly; i.e. peristaltic contractions immediately effect flow through the duodenum. Knowledge of this led to a method of constructing channel inlet boundary conditions despite having data that were mismatched with the video sequence.

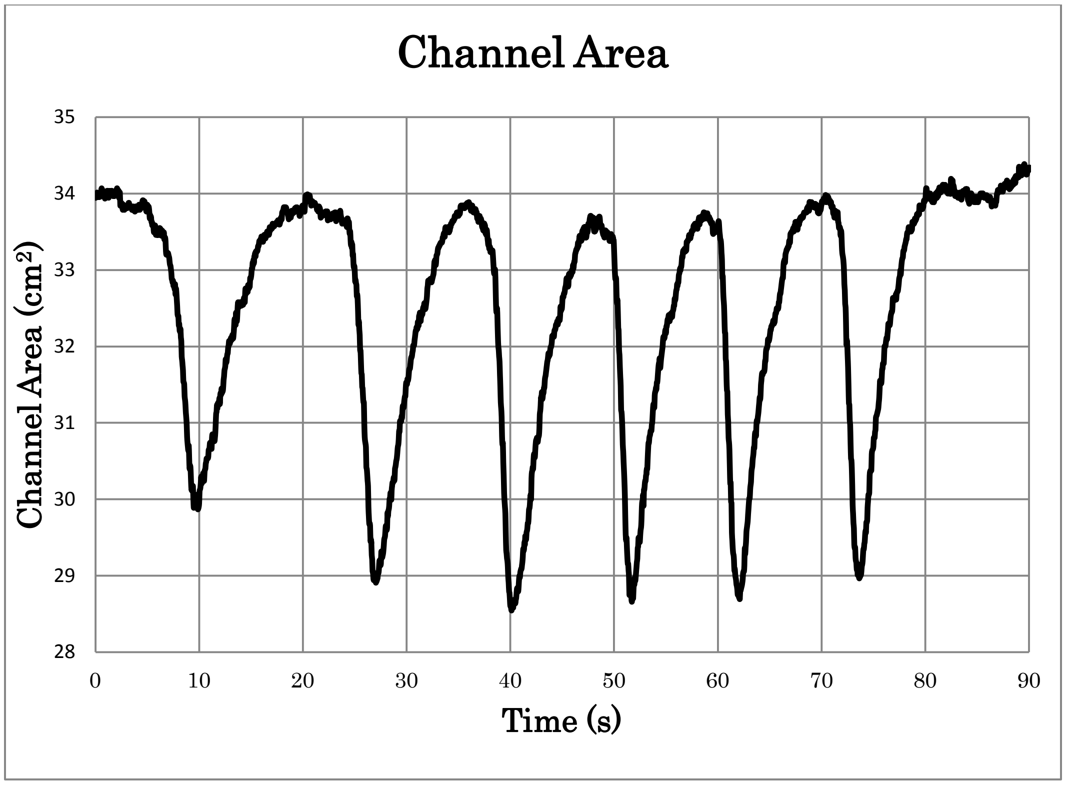


Figure 3-48. The projected channel area of the contracting duodenum. Data provided did not match temporally with the provided video file frames, so these channel area data, along with corresponding fluid volumes in the proximal cup supplying fluid to the duodenal segment, were used in constructing boundary conditions for the CFD calculations performed here.

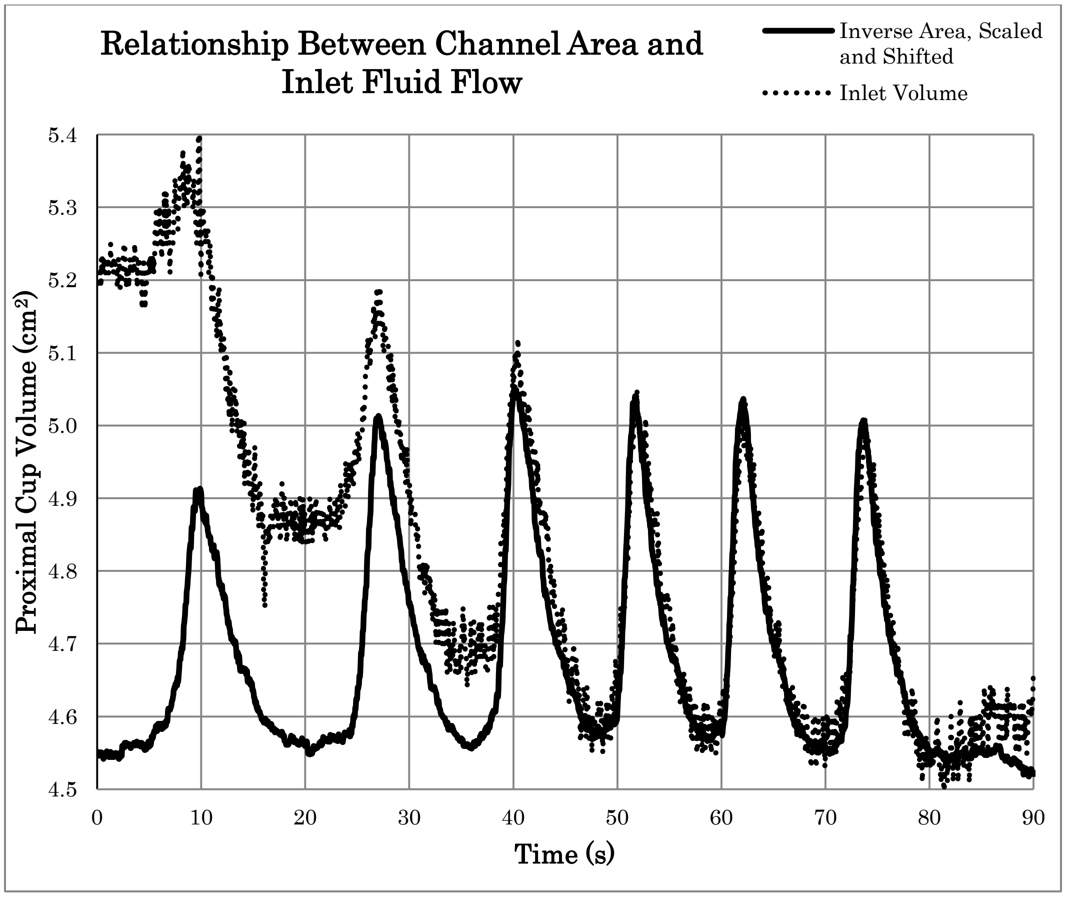


Figure 3-49. The strong temporal correlation between fluid volume in the proximal reservoir and projected area of the contracting duodenal segment is clearly illustrated here. The channel area has been inverted, then scaled and shifted, so that it closely matches the steady state behavior of the fluid reservoir. Temporal changes in fluid volume define flow rates, which in turn define velocity if the rate of volumetric flow is through a constant cross-sectional area. Thus, inlet velocity can be defined in terms of changing inverse channel area.

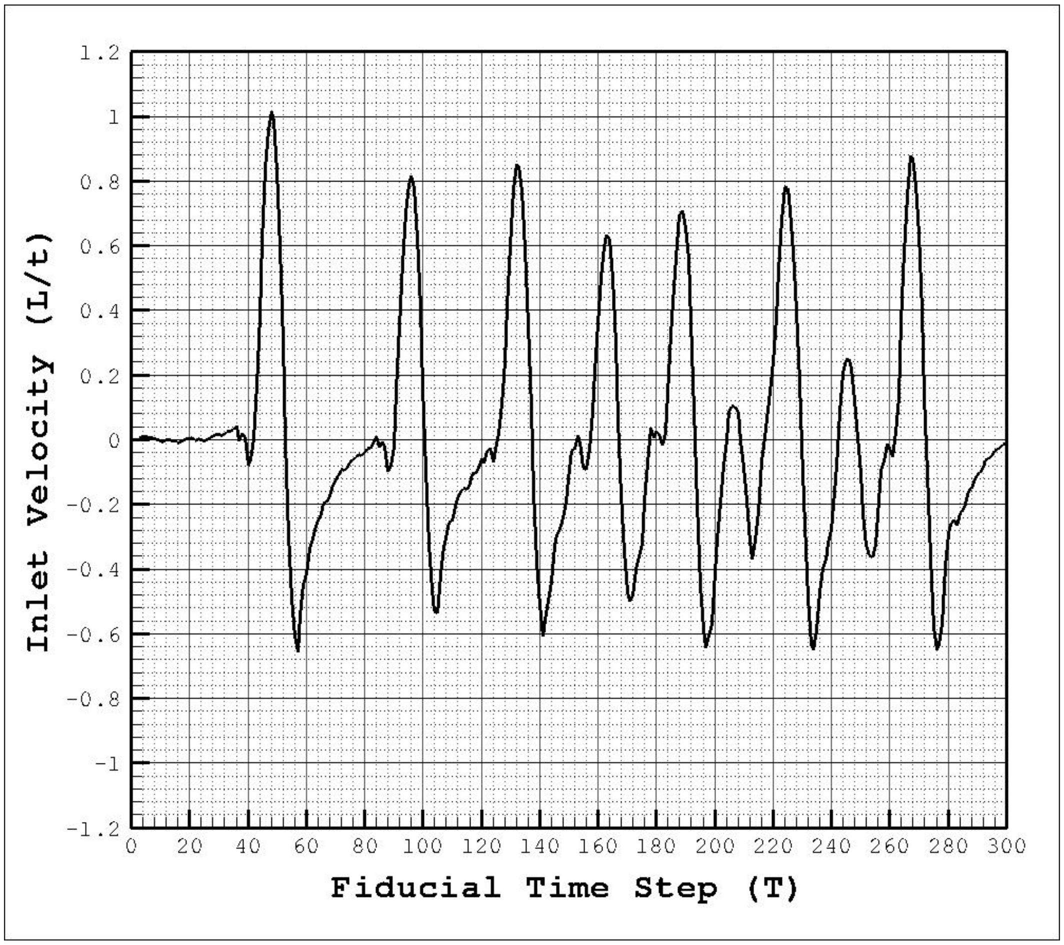


Figure 3-50. Inlet velocity boundary conditions for the guinea pig duodenum, calculated based upon the rate of change of 2D channel area between image frames. The velocity was scaled so that its peak value would be unity, making for convenient assignment of maximum Reynolds number based upon the channel's inlet diameter.

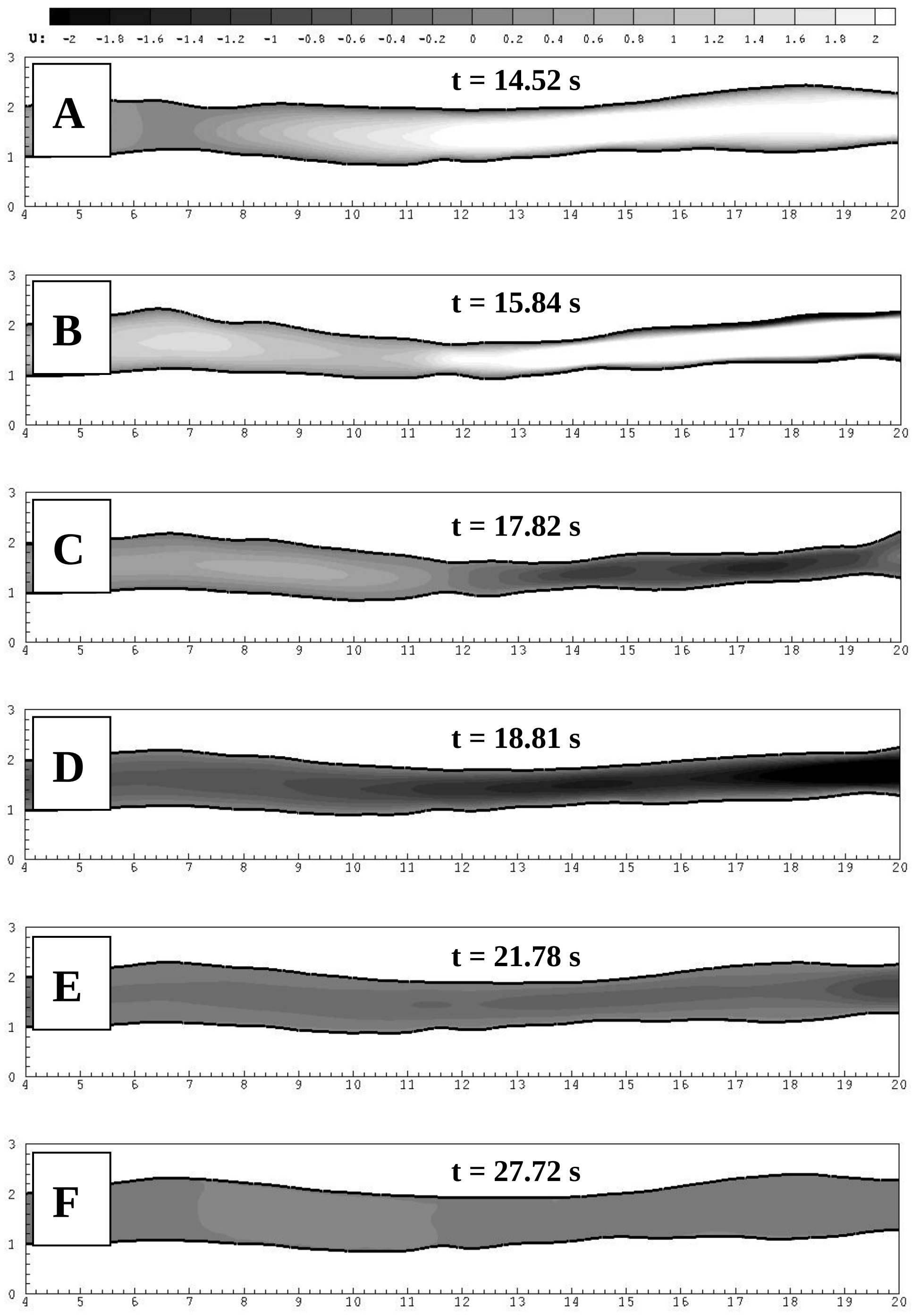


Figure 3-51. Contours of X-velocity during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 20.

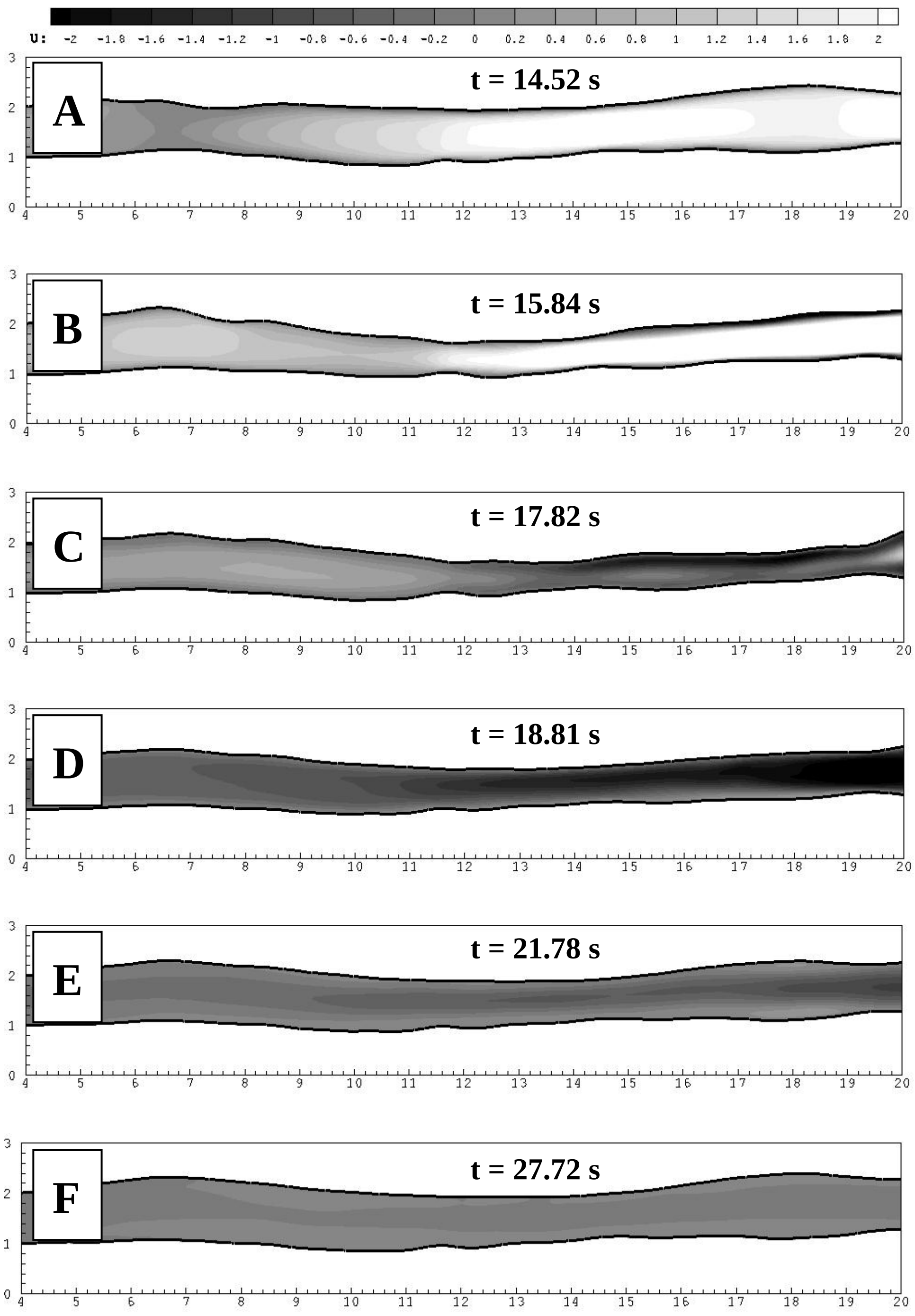


Figure 3-52. Contours of X-velocity during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 100.

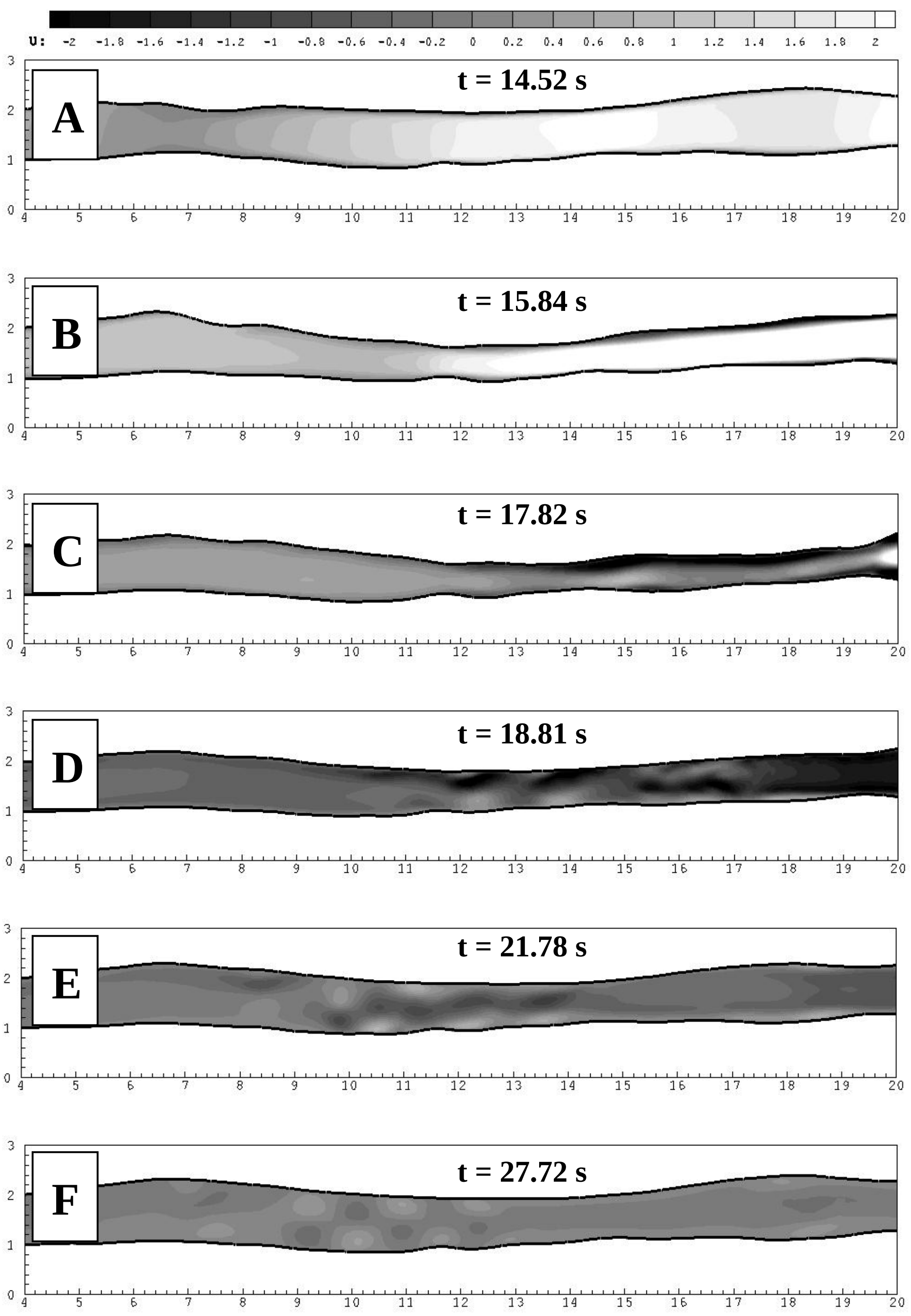


Figure 3-53. Contours of X-velocity during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 500.

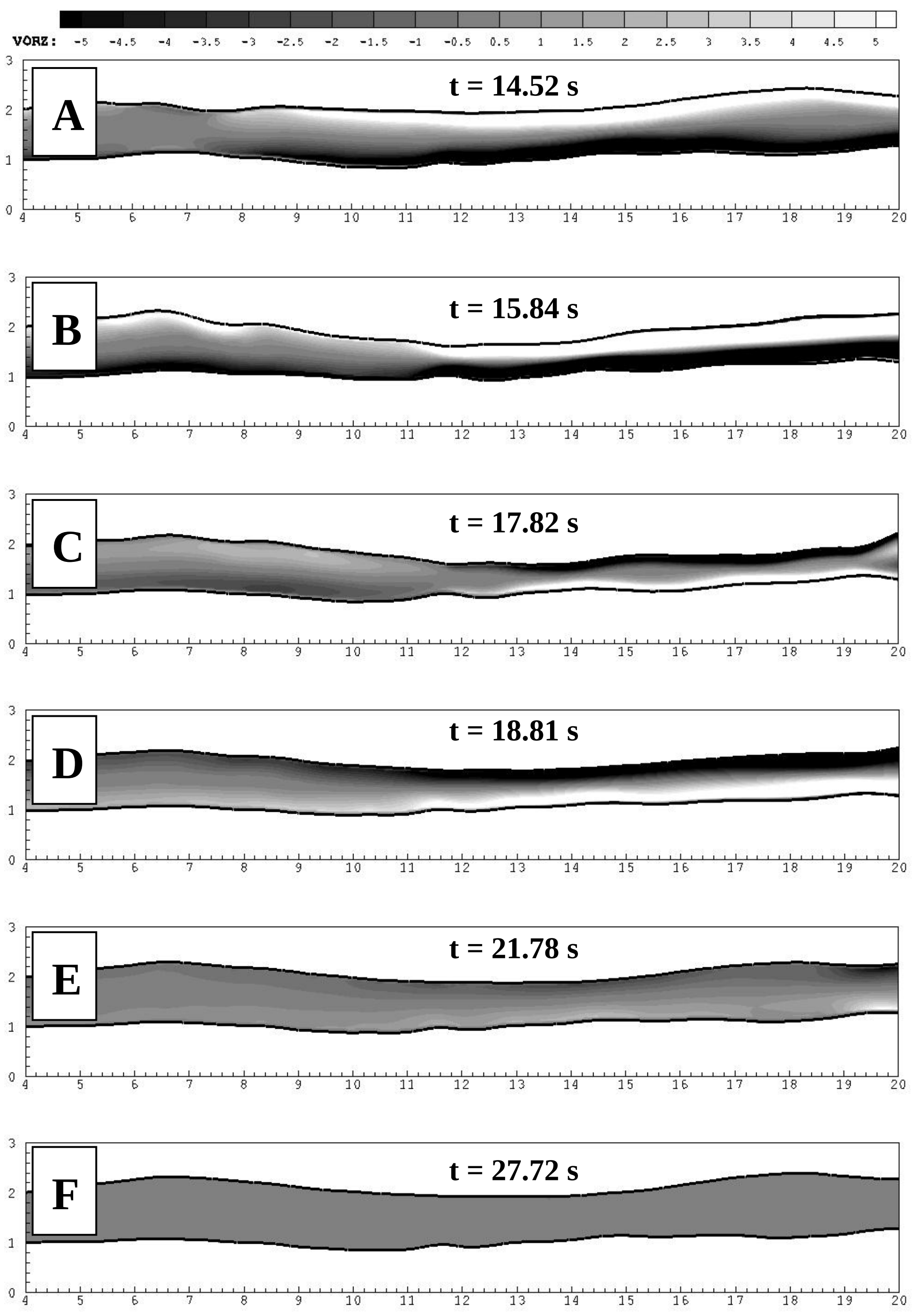


Figure 3-54. Contours of vorticity during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 20.

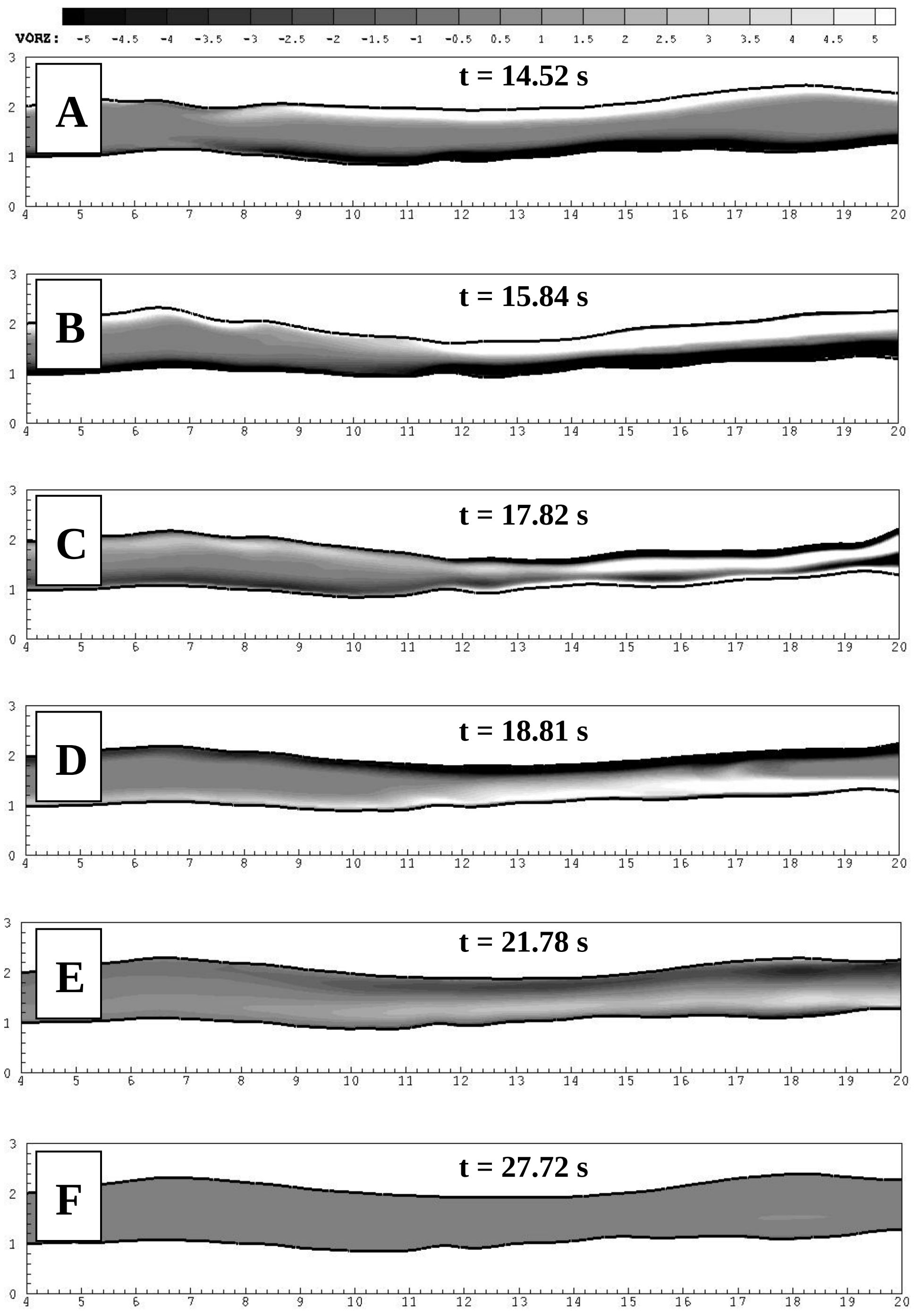


Figure 3-55. Contours of vorticity during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 100.

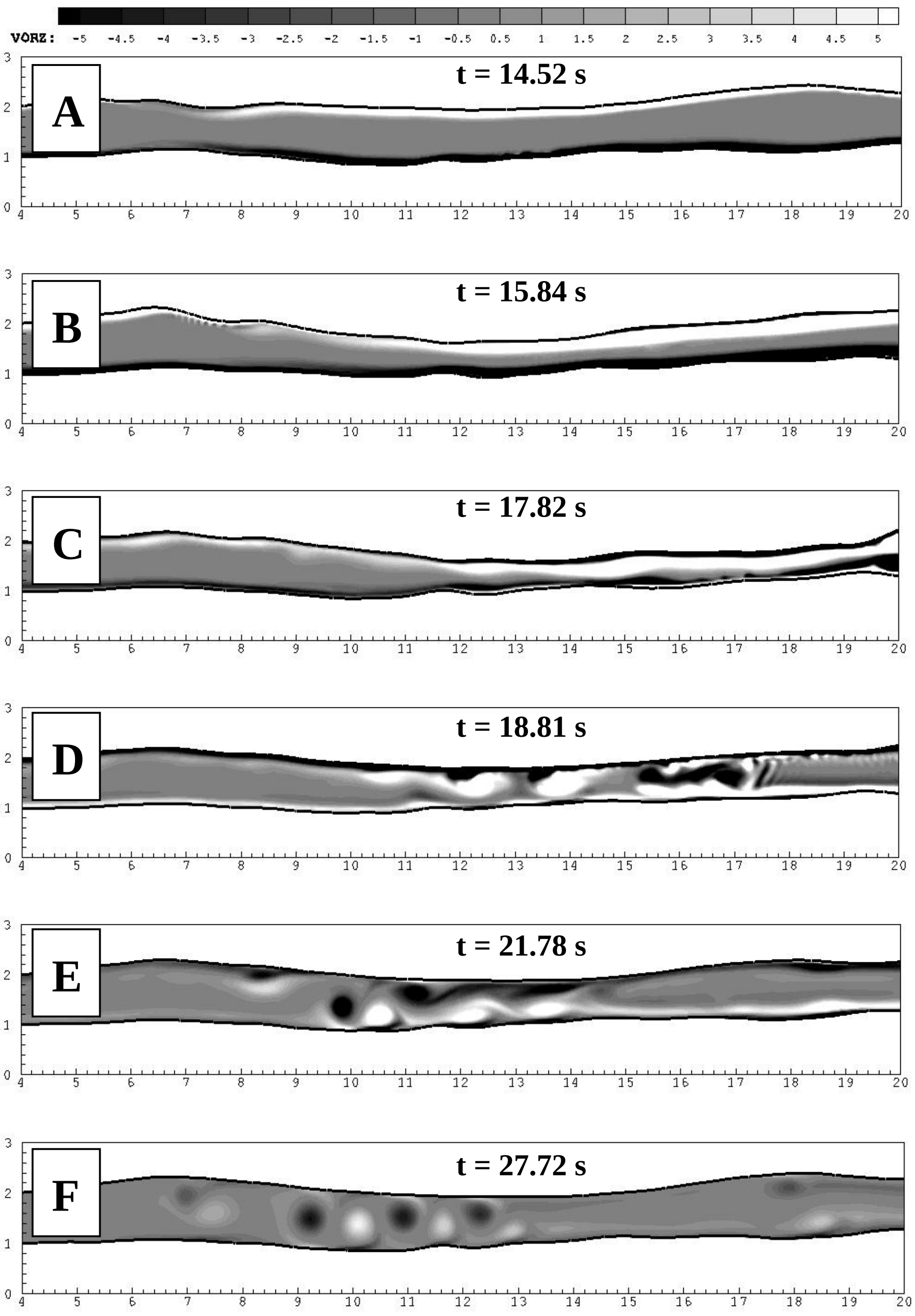


Figure 3-56. Contours of vorticity during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 500.

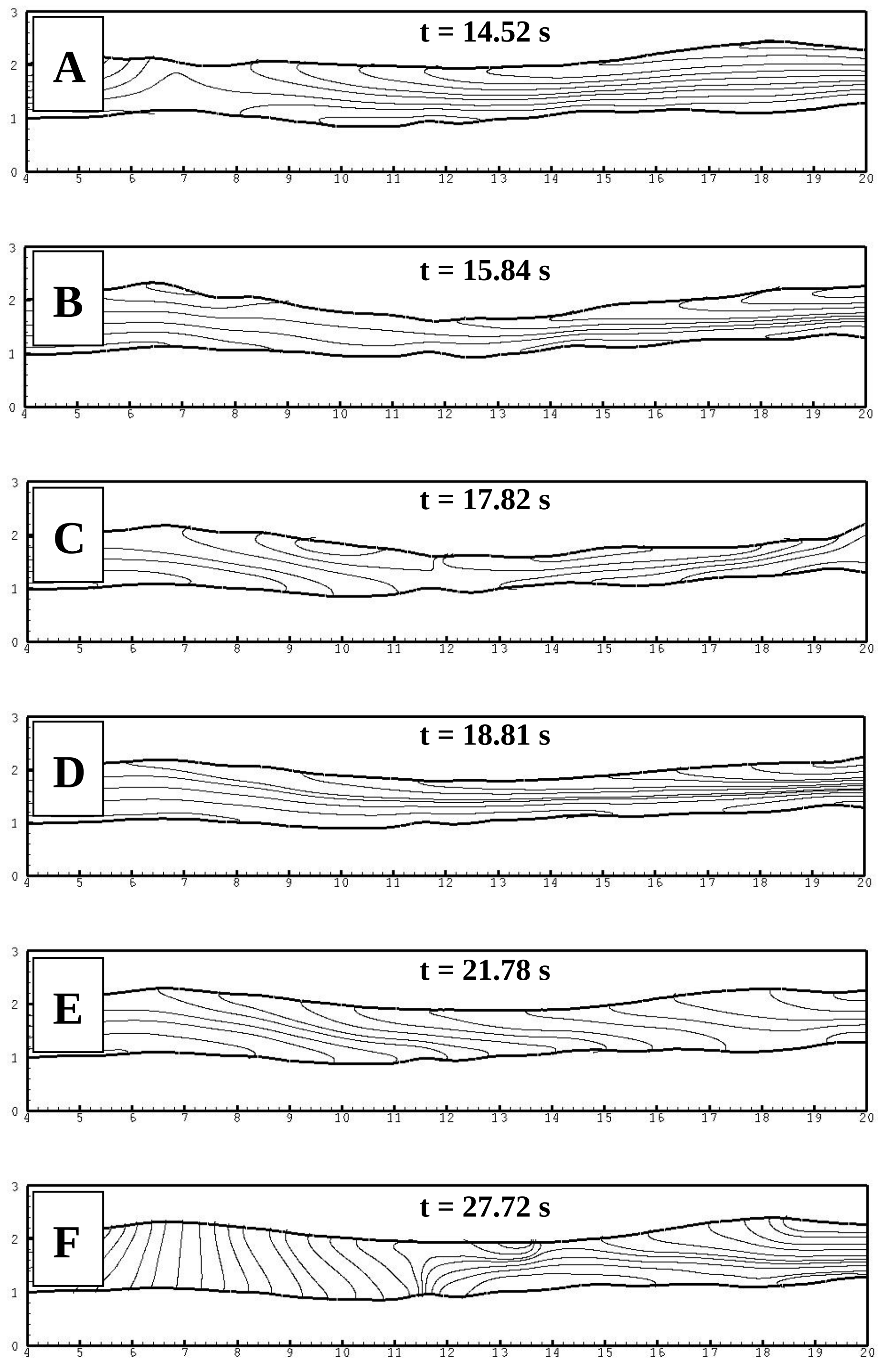


Figure 3-57. Streamlines plotted during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 20.

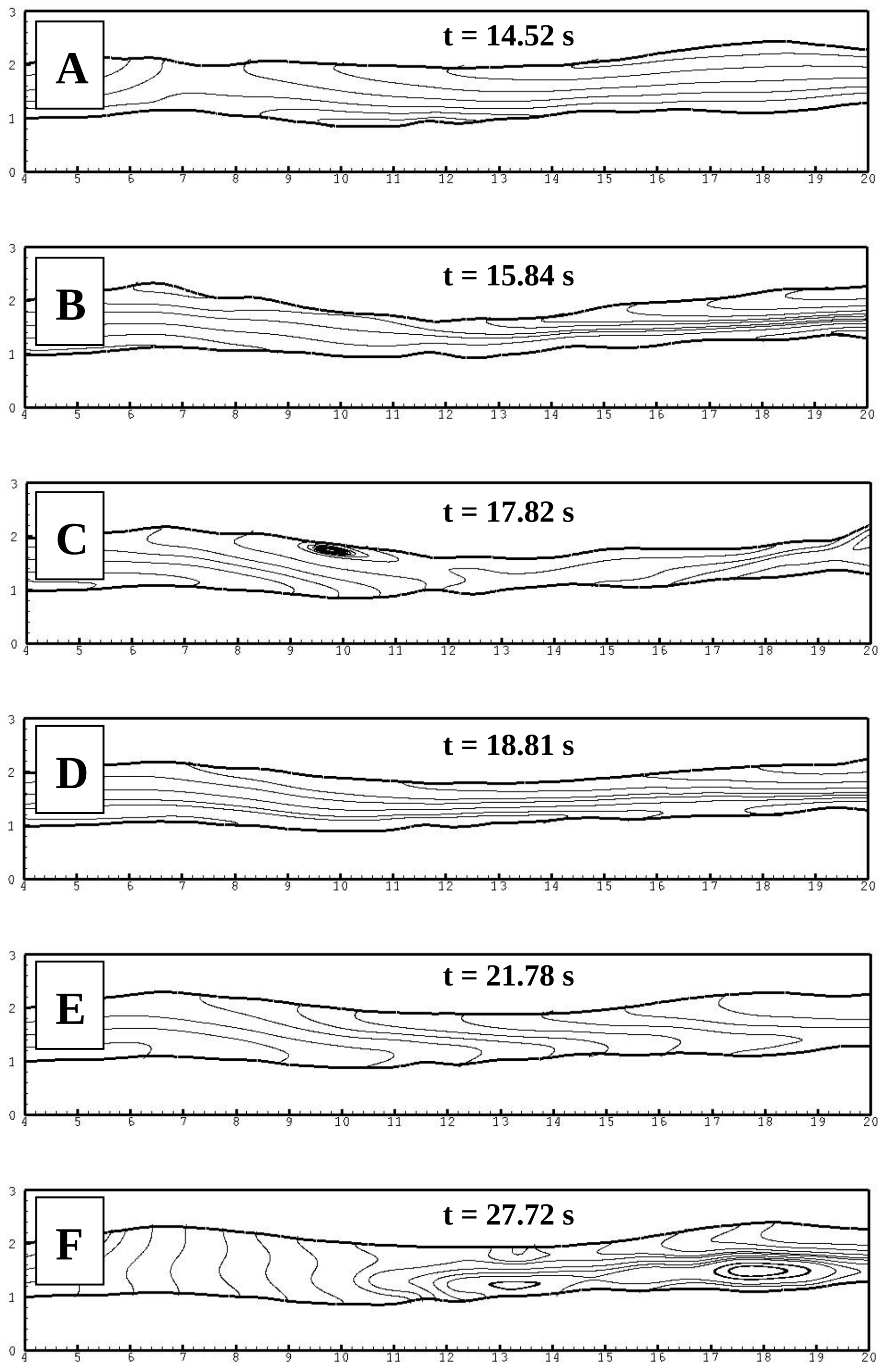


Figure 3-58. Streamlines plotted during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 100.

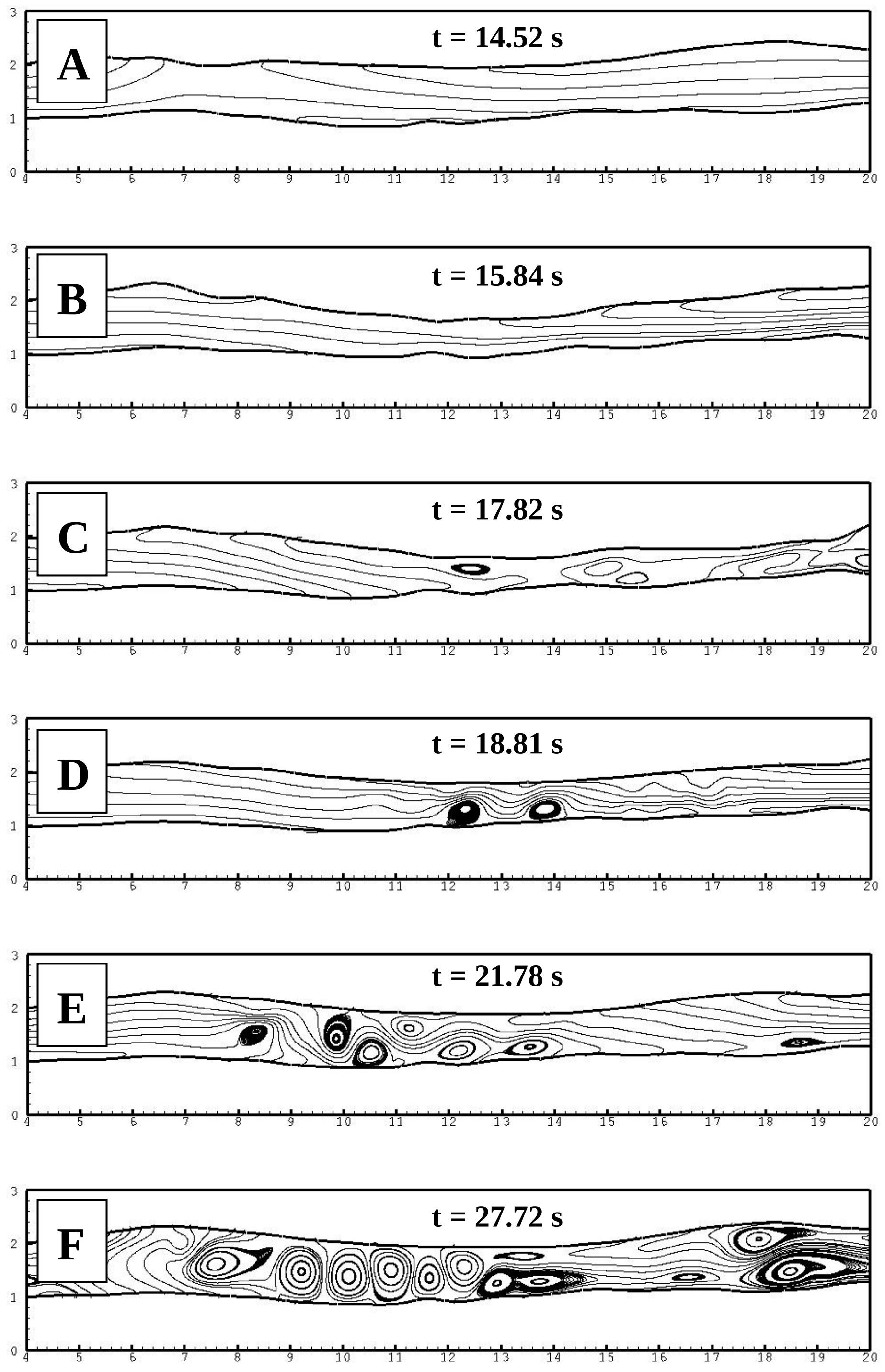


Figure 3-59. Streamlines plotted during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 500.

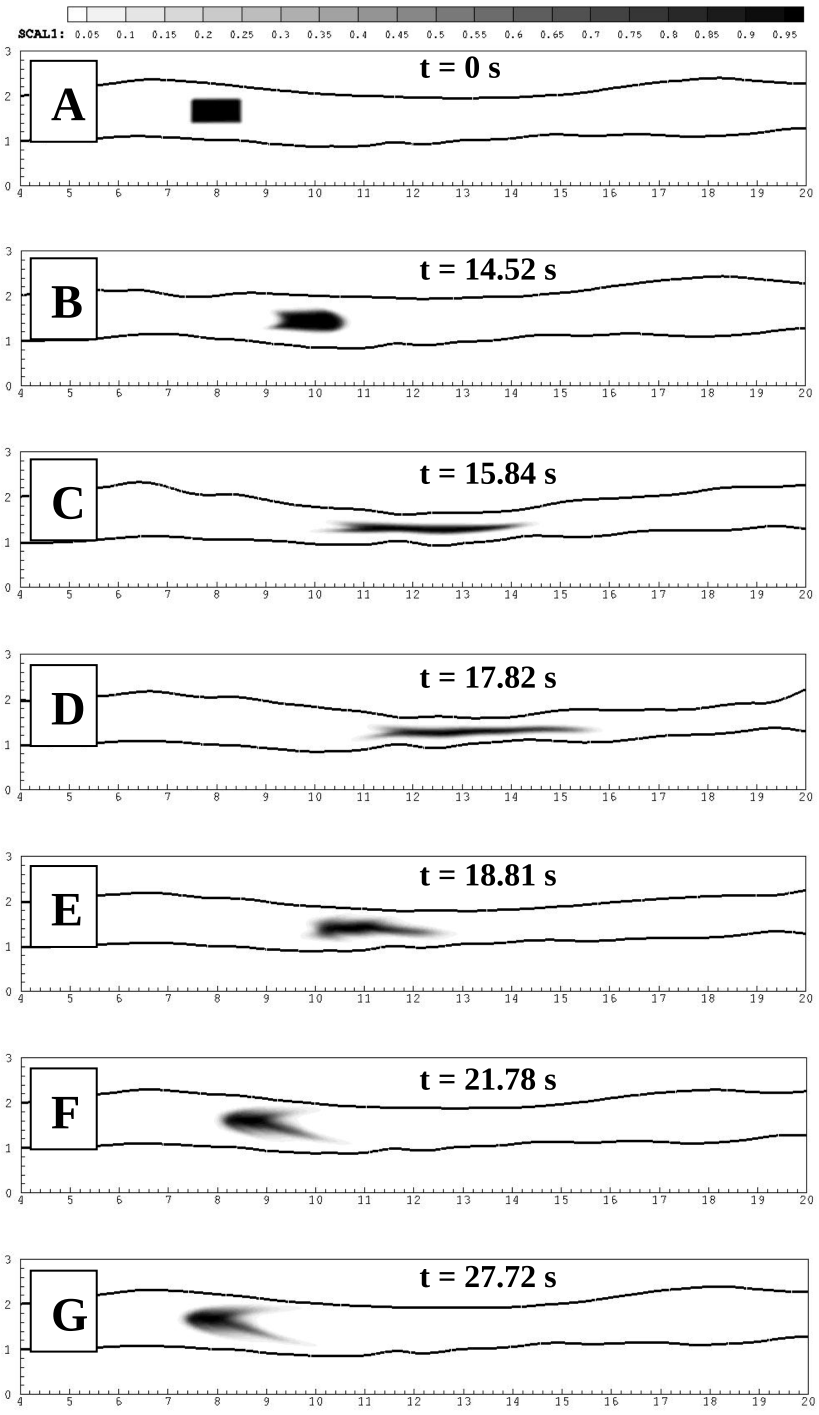


Figure 3-60. Contours of species mixing during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 20.

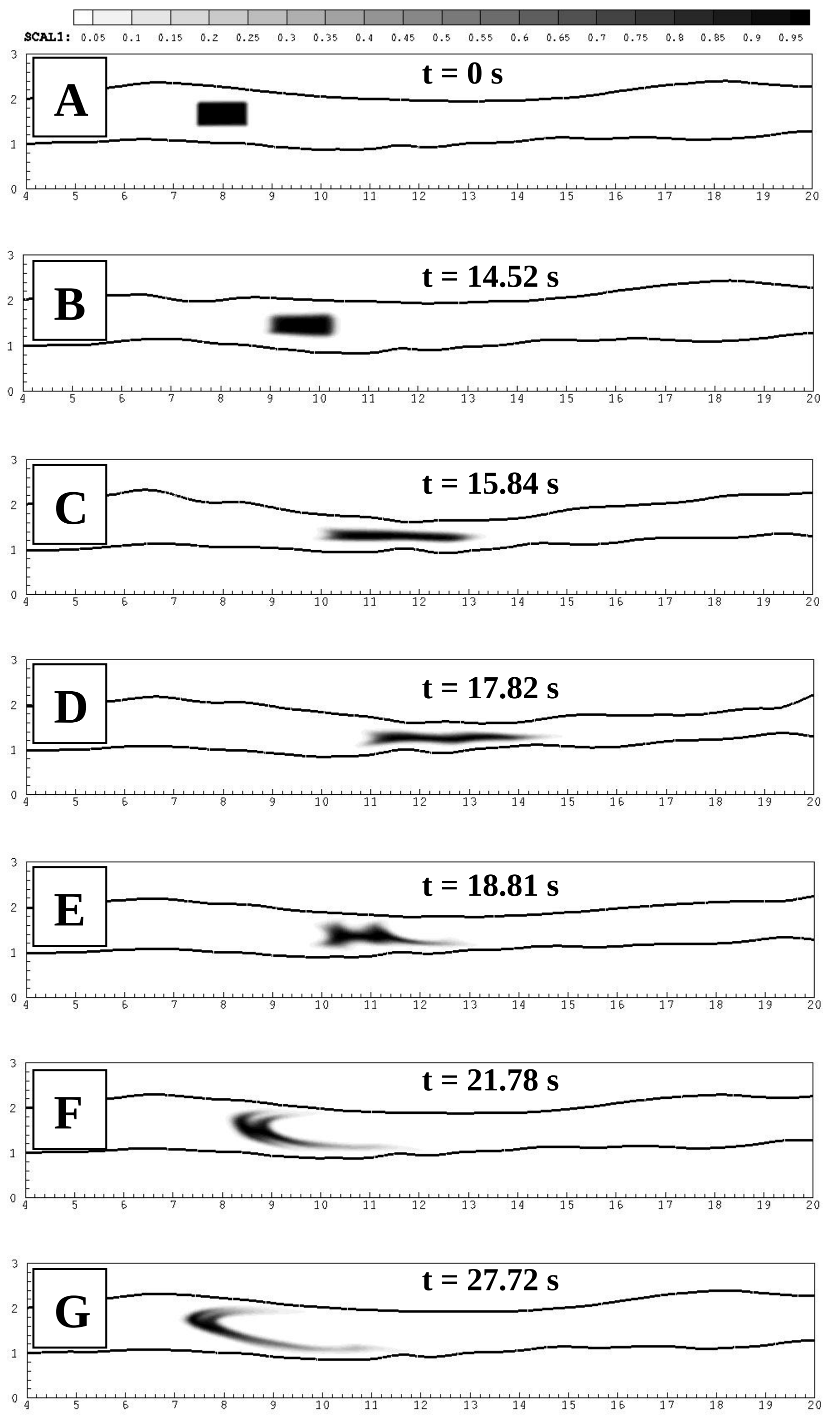


Figure 3-61. Contours of species mixing during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 100.

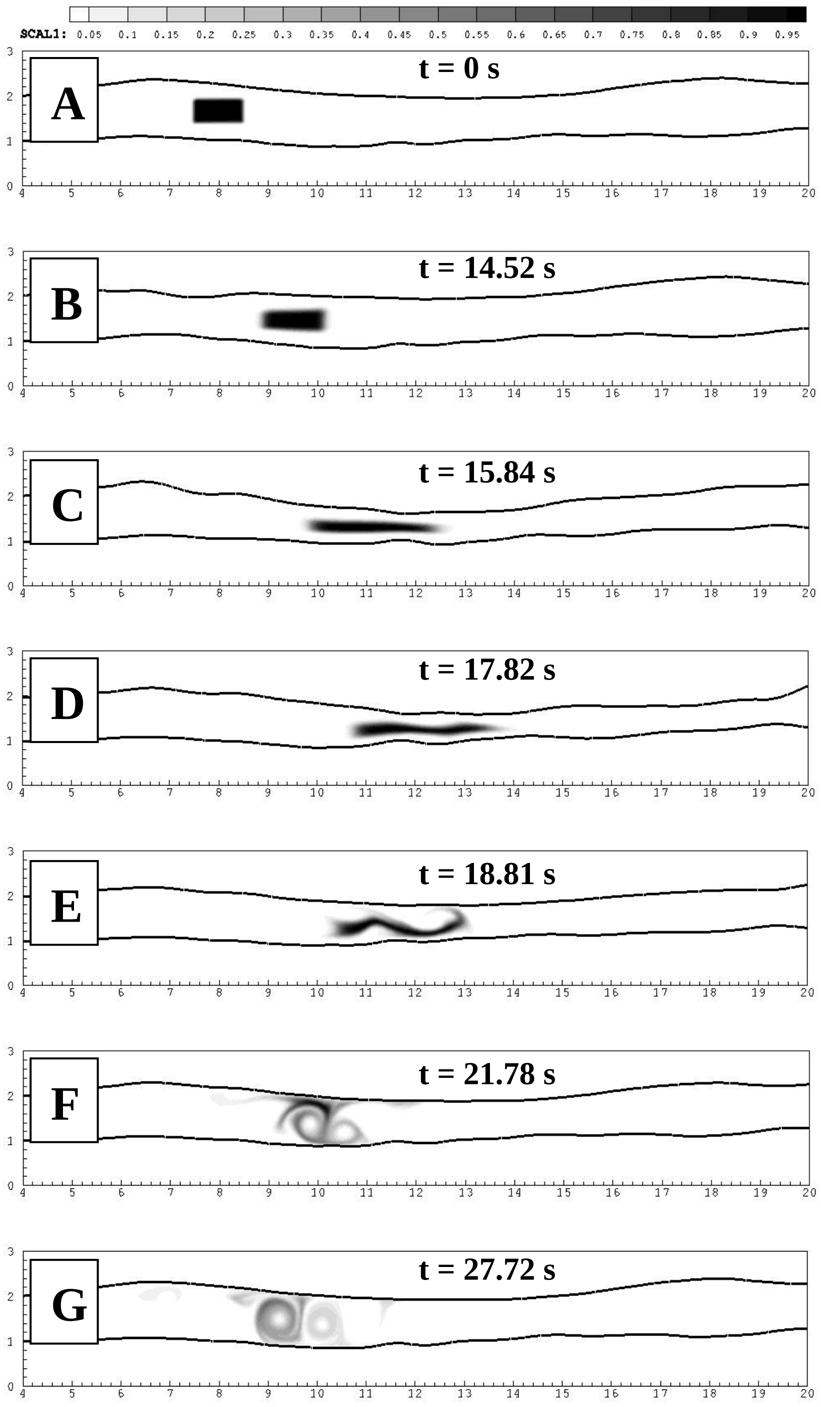
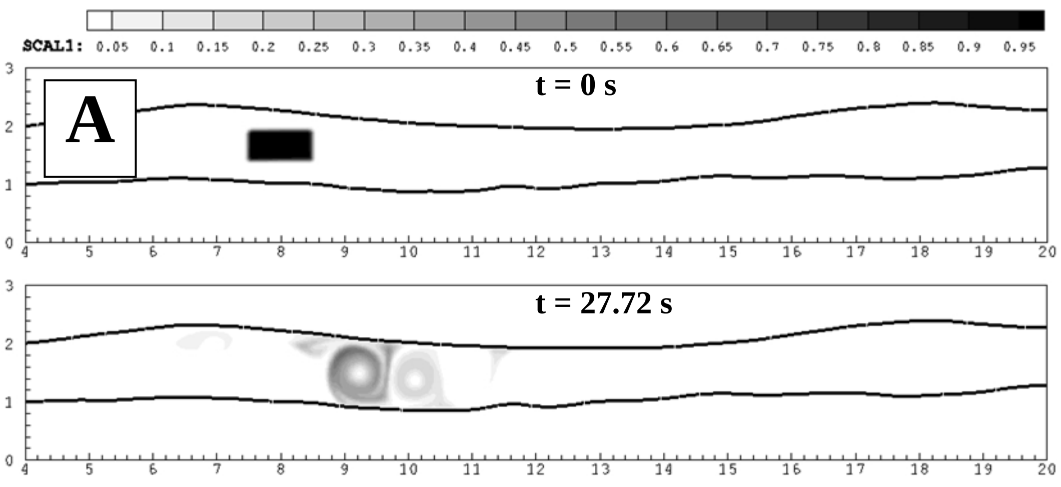


Figure 3-62. Contours of species mixing during the first peristaltic contraction of the guinea pig duodenum in vitro; Re = 500.



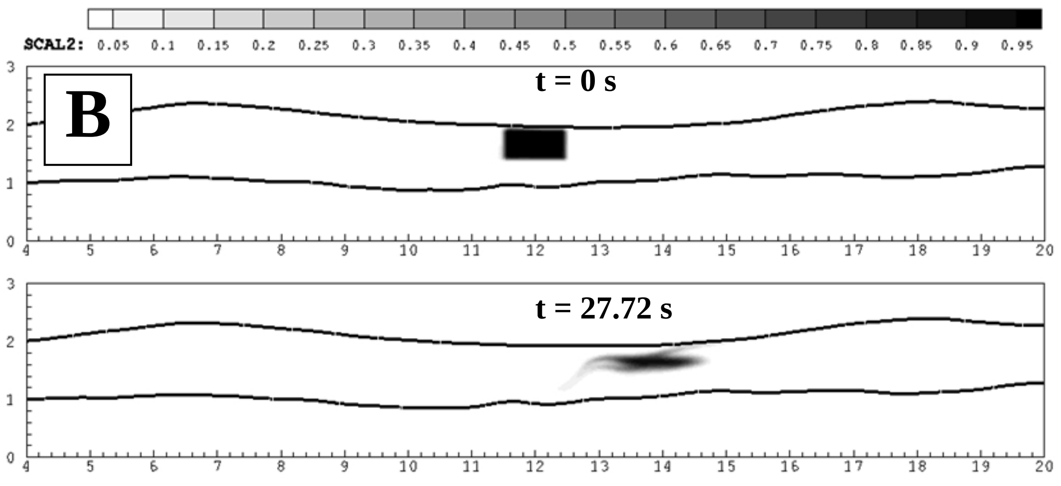
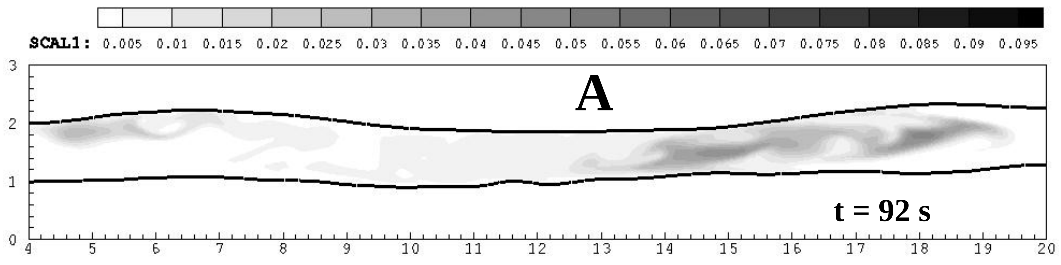


Figure 3-63. Initial scalar block placement greatly impacts the quality of mixing produced by one peristaltic contraction. Re = 500.



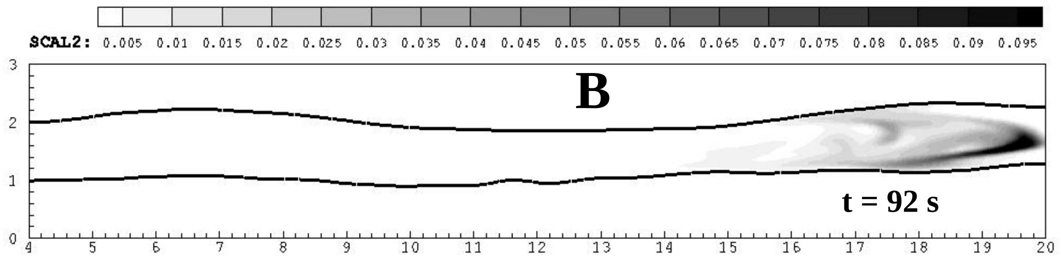


Figure 3-64. Final scalar species concentration field following 7 complete peristaltic contraction waves; Re = 500: scalar block 1 (A) and scalar block 2 (B) have assumed very different morphologies, though their scalar variance measurement is quite similar. Which should be considered better mixed? Note that the concentration scale has decreased by an order of magnitude.

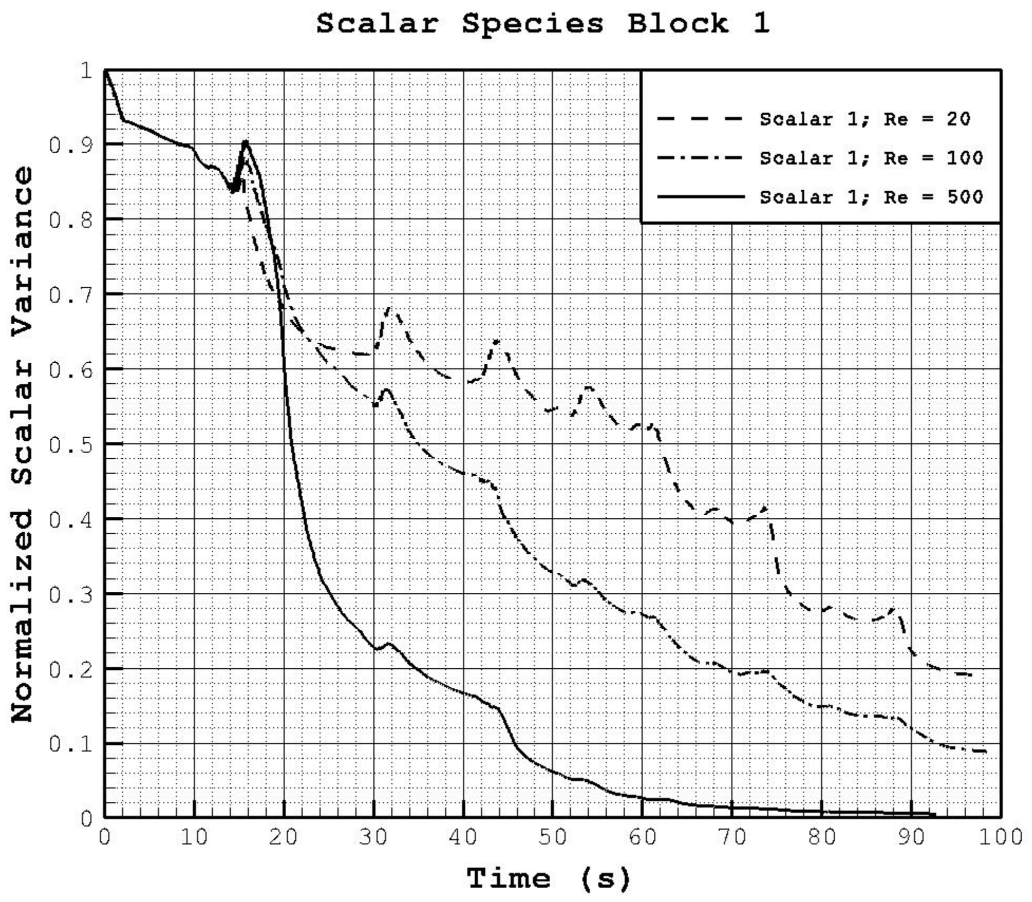


Figure 3-65. Scalar variance plotted for species block 1 at all 3 Reynolds numbers simulated.

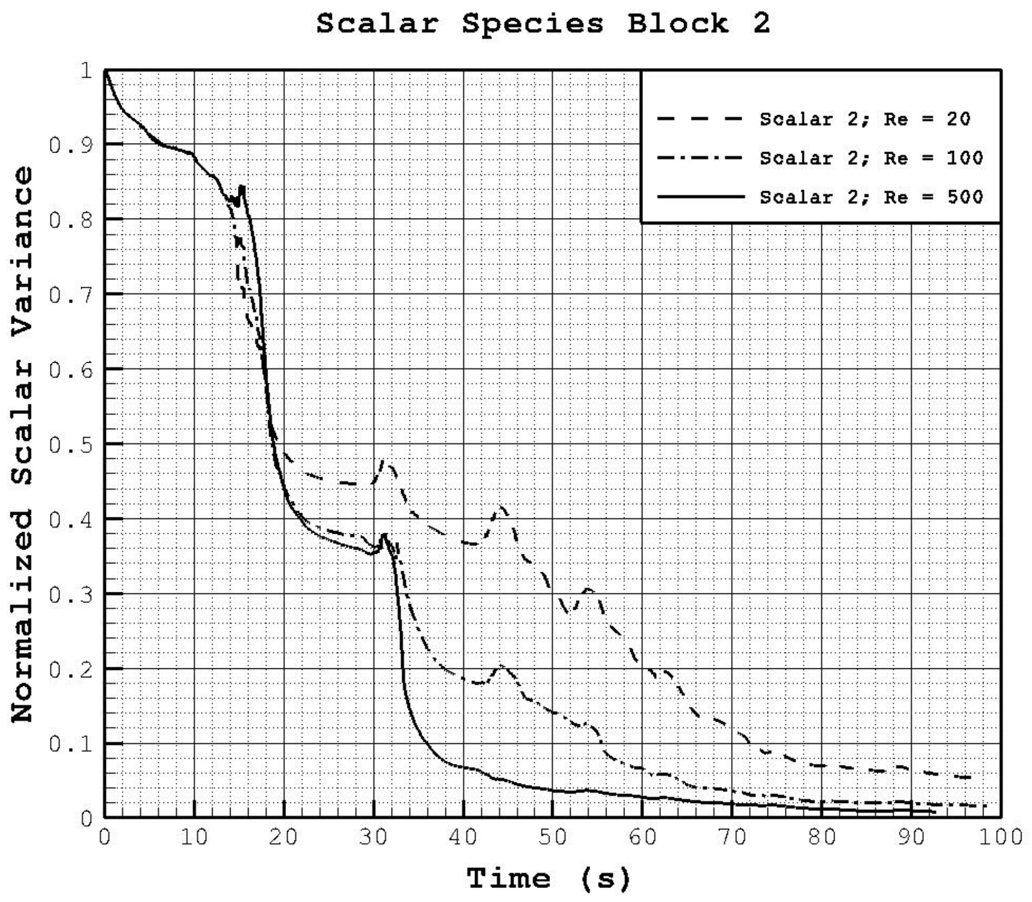


Figure 3-66. Scalar variance plotted for species block 2 at all 3 Reynolds numbers simulated.

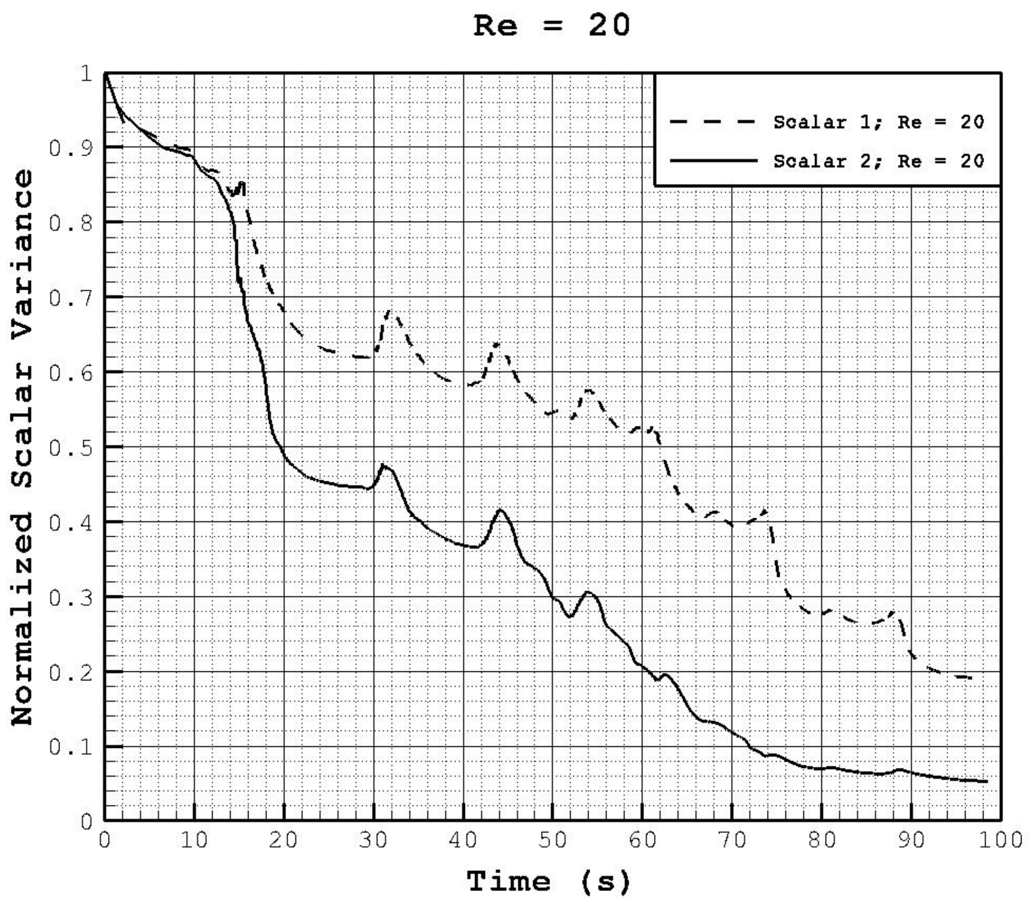


Figure 3-67. Comparison of measured mixedness between scalar species blocks 1 and 2; Re = 20.

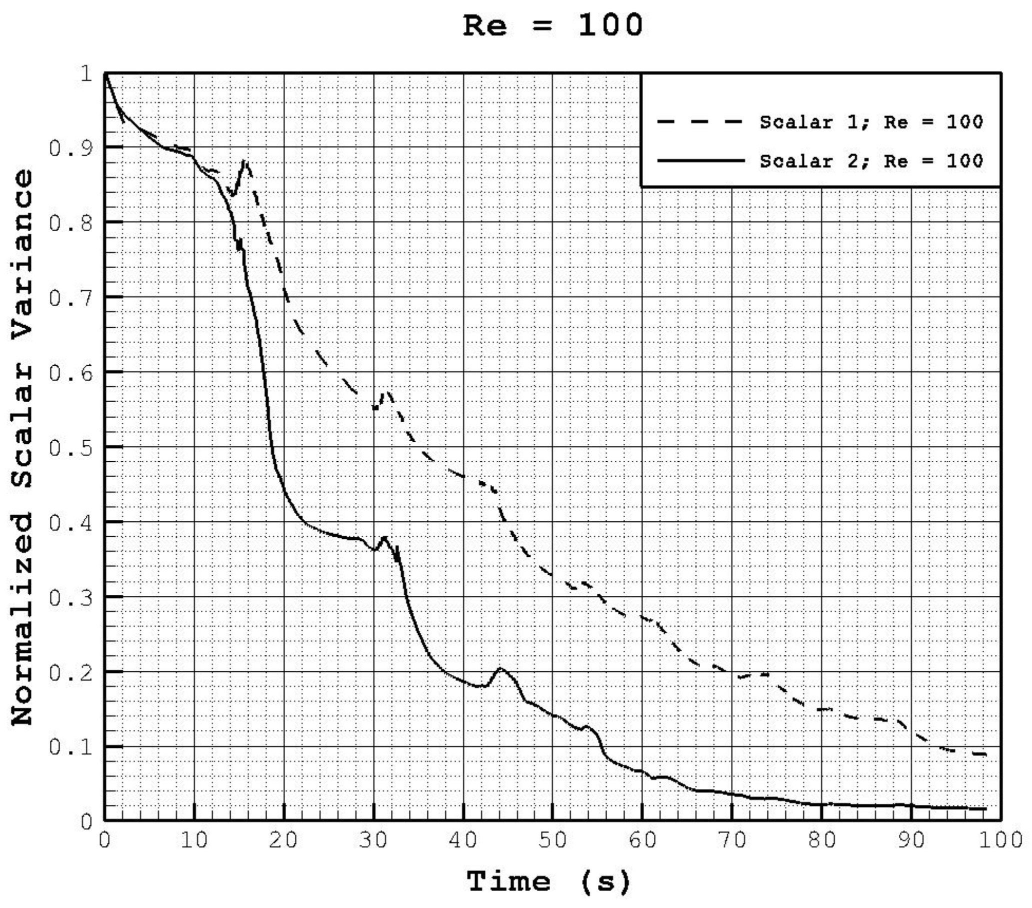


Figure 3-68. Comparison of measured mixedness between scalar species blocks 1 and 2; Re = 100.

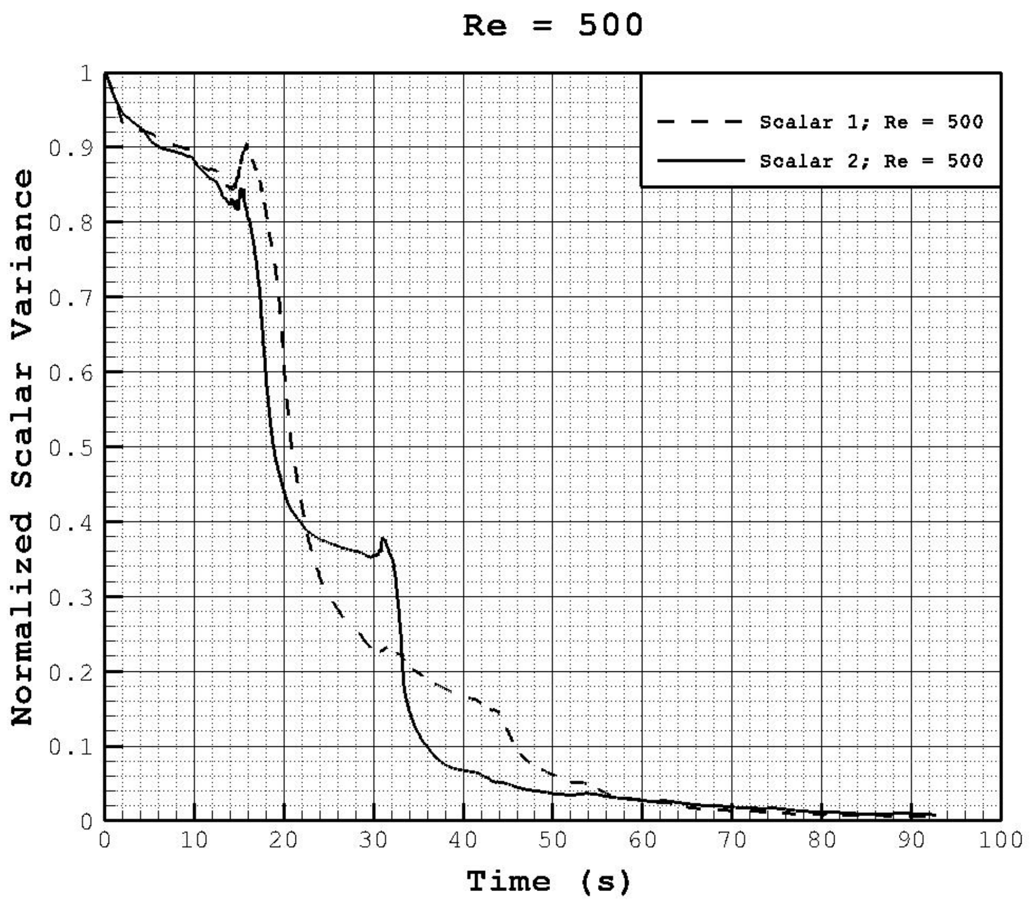


Figure 3-69. Comparison of measured mixedness between scalar species blocks 1 and 2; Re = 500.

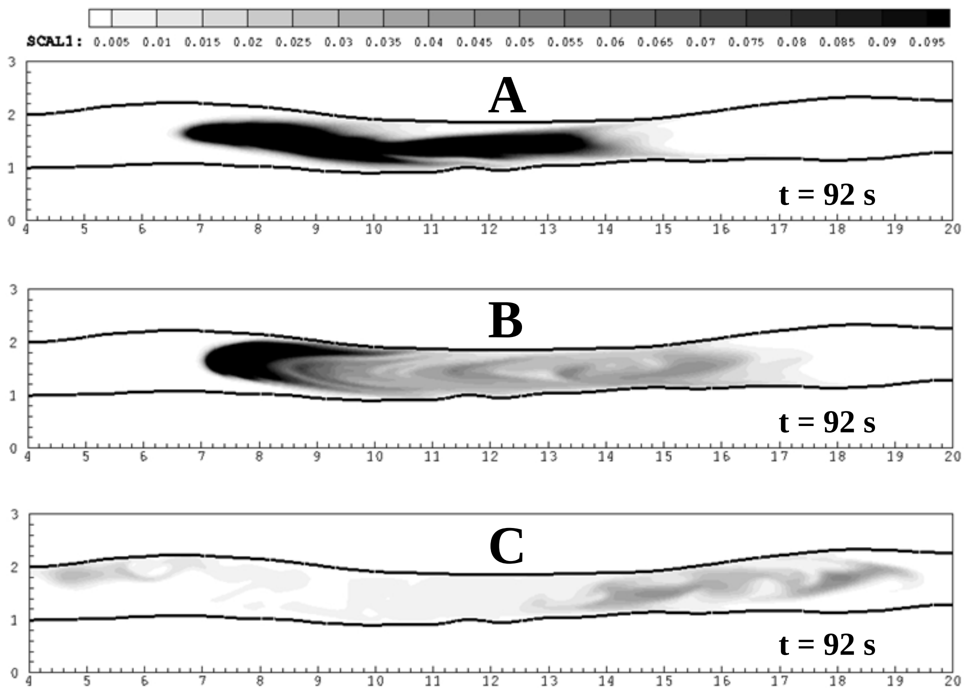


Figure 3-70. Final scalar species concentration field following 7 complete peristaltic contraction waves; Re = 20 (A), Re = 100 (B), Re = 500 (C).

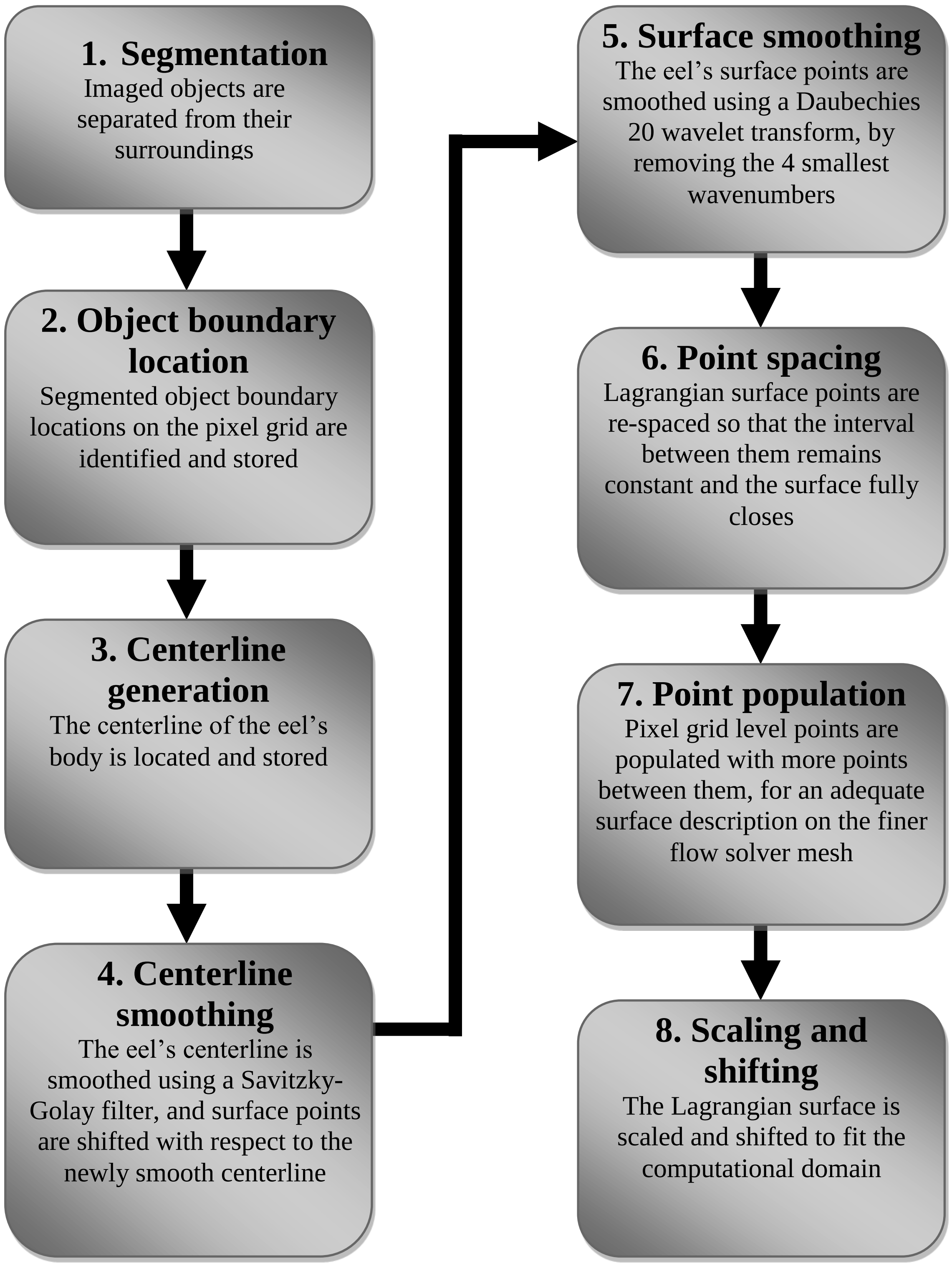

1. Segmentation
Imaged objects are separated from their surroundings
2. Object boundary location
Segmented object boundary locations on the pixel grid are identified and stored
3. Centerline generation
The centerline of the eel's body is located and stored
4. Centerline smoothing
The eel's centerline is smoothed using a Savitzky-Golay filter, and surface points are shifted with respect to the newly smooth centerline
5. Surface smoothing
The eel's surface points are smoothed using a Daubechies 20 wavelet transform, by removing the 4 smallest wavenumbers
6. Point spacing
Lagrangian surface points are re-spaced so that the interval between them remains constant and the surface fully closes
7. Point population
Pixel grid level points are populated with more points between them, for an adequate surface description on the finer flow solver mesh
8. Scaling and shifting
The Lagrangian surface is scaled and shifted to fit the computational domain

Figure 3-71. Meshing the surface of the eel for flow calculations requires 8 tedious steps.

CHAPTER 4
A MORE DETAILED LOOK AT IMAGE DENOISING AND SEGMENTATION

4.1 Introduction

In the previous chapter, the ideas of image denoising and segmentation were introduced in the context of image-based modeling. Preliminary results were encouraging, but a more in-depth look at transforming images (still or moving) from various modalities into a form that is compatible with the flow solver was found to be necessary to obtain a truly robust and general facility. This was primarily motivated by two issues. First, the shift from a Lagrangian to an Eulerian treatment of image-based models requires acquisition of quality segments from images directly, as there no longer exists a set of points to smooth and later construct level set contours from. Second, it is desirable for an image-based modeling approach to be extensible to 3-D imaging modalities such as CT and ultrasound, rather than being restricted to 2-D video files and the like. This introduces greater levels of complexity, because available image data acquired using such modalities rarely possess ideal levels of contrast and are typically plagued by considerable levels of noise. Noise in images can be of two kinds, viz. additive and multiplicative noise. While additive noise is common to almost all modalities, multiplicative noise (also called “speckle”) plays an important role in such image acquisition techniques as X-ray CT and ultrasound. Indeed, much literature exists pertaining specifically to noise in medical imaging and ways to deal with the problems caused by these two types of noise. Pertinent ideas resulting from a survey of this body of literature are discussed presently.

4.2 Image Denoising

The key idea that is pursued in this work is the following:

Since the Eulerian flow solver relies on a Cartesian mesh and an implicit, level set based representation of embedded boundaries, it is attractive to present geometries to the flow code in the form of level sets. To this end, images acquired by a wide variety of means can be processed through segmentation into level set fields that can then be employed directly by the flow code; this obviates the rather tedious process of converting images into surface meshes and fitting volume meshes to conform to this surface mesh. In order to achieve this rather nice ability to bypass mesh generation, level set-based segmentation can be employed. However, the quality of the segmentation can only be as good as the quality of information available within the image domain being segmented.

All real images $I(\boldsymbol{x}, t)$ contain some type of noise corruption $N(\boldsymbol{x}, t)$ of the original signal $s(\boldsymbol{x}, t)$, whether caused by film grain or from the acquisition process itself [20].

$$I(\boldsymbol{x}, t) = s(\boldsymbol{x}, t) + N(\boldsymbol{x}, t) \tag{4.1}$$

In general, images that are interesting from a modeling perspective contain large amounts of noise, which will ultimately lead to the inability of most processing algorithms to segment coherent image objects without also segmenting noisy regions or pixels not belonging to any objects. Thus, pre-processing images to remove noise is necessary before segmentation can proceed.

Two different test images were chosen for developing the segmentation and denoising techniques outlined in this chapter. The first is a "Shapes" test image consisting of five shapes with grey level 127 set against a homogeneous background of grey level

195, generated specifically for this work (this image is shown along with a 3-D rendering of it in Figure 4-1). The second is the IEEE standard test image “Lena,” chosen for its widespread use in the image processing literature and consequent provision of a good basis for comparison of results.

For algorithm testing purposes, each of the test images was made noisy with the application of one of two different levels of additive Gaussian white noise (a Gaussian noise distribution featuring a zero mean added randomly to the intensity signal [20, 71]), with standard deviation set to $\sigma = 30$ for a moderately noisy case, and $\sigma = 100$ for a highly noisy case (Figure 4-6). Figure 4-8 shows the results of attempting to segment noisy images without any pre-processing, with the 3-D rendering provided in Figure 4-9 clearly illustrating the nature of the difficulties encountered, even when a moderately noisy field with $\sigma = 30$ is applied to a 2-tone image containing simple shapes. Segmentation generally entails grouping pixels together according to their brightness or some property related to it. If some bright noisy pixels appear in an otherwise dark region of an image, those bright pixels will be regarded by a segmentation algorithm as “belonging to” a different group of pixels than the ones in their immediate neighborhood, resulting in discontinuous, noisy segmentation patterns. In the specific case of the Shapes test image, bright noisy pixels inside the boundaries of the relatively darker shapes will be considered part of the background, and dark noisy background pixels will be associated with the same group the shapes belong to. For this reason, segmentation is necessarily coupled with some sort of denoising facility when applied to real images, which will always contain some amount of noise.

Image denoising is an active research topic in the computer graphics community with a host of methods available; however, in the present framework the desired denoising facility must align with the need to segment images such that the result can be conveniently employed in flow computations. Therefore, an assessment of appropriate, state-of-the-art competing denoising techniques was made to determine the most suitable choice from the point of view of computational modeling.

The most modest denoising methods involve diffusive blurring, such as applying Gaussian filters and isotropic diffusion schemes to images in order to diminish the relative magnitude of outlying pixel values. While being effective in noise removal, isotropic diffusion methods suffer an inability to preserve object boundaries, as sharp edges are blurred just the same as isolated noisy pixels are, without any means for discriminating between the two. This is problematic for segmentation, which relies on the presence of strong gradients and high contrast in order to conform to object boundaries with fidelity. Therefore, based on the imaging modality, the type of system being imaged and the precision of the geometric features required to be captured, the choice of denoising technique may be different. In the following sections we examine some modern denoising techniques to determine their suitability for the types of applications targeted in the computational fluid dynamics community.

4.2.1 Anisotropic Diffusion

In order to address the problem of uniform diffusion destroying valuable image information, Perona and Malik proposed a nonlinear PDE for smoothing an image on a continuous domain, known as *anisotropic diffusion*, in which diffusion rates depend locally on image intensity gradients:

$$\begin{cases} \frac{\partial \boldsymbol{I}}{\partial \boldsymbol{t}} = \nabla \cdot [\boldsymbol{c}(|\nabla \boldsymbol{I}|) \cdot \nabla \boldsymbol{I}] \\ \boldsymbol{I}(\boldsymbol{t} = \mathbf{0}) = \boldsymbol{I_o} \end{cases} \quad \textbf{(4.2)}$$

In Equation 4.2, $\boldsymbol{\nabla}$ is the gradient operator, $(\boldsymbol{\nabla}\,\cdot)$ is the divergence operator, $c(x)$ is the diffusion coefficient, and I_o is the original image before processing. Two possible ways of setting the diffusion coefficients are suggested:

$$\boldsymbol{c}(\boldsymbol{x}) = \frac{\mathbf{1}}{\mathbf{1}+(\boldsymbol{x}/\boldsymbol{\kappa})^{\mathbf{2}}} \quad \textbf{(4.3)}$$

or

$$\boldsymbol{c}(\boldsymbol{x}) = \mathbf{exp}[-(\boldsymbol{x}/\boldsymbol{\kappa})^{\mathbf{2}}]\,. \quad \textbf{(4.4)}$$

In both cases, κ is an edge magnitude parameter with an optimal value that depends on image characteristics. The discrete form of the Perona/Malik PDE is written as

$$I_s^{t+\Delta t} = I_s^t + \frac{\Delta t}{|\bar{\eta}_s|} \sum_{p \in \bar{\eta}_s} c\left(\boldsymbol{\nabla} I_{s,p}^t\right) \boldsymbol{\nabla} I_{s,p}^t, \quad \textbf{(4.5)}$$

where I_s^t is the discretely sampled image, s denotes pixel position on a discrete grid, Δt is the time step size, $\bar{\eta}_s$ is the spatial neighborhood of pixel s, $|\bar{\eta}_s|$ is the number of pixels in the neighborhood window (usually the four cardinal neighbors in 2-D), and $\boldsymbol{\nabla} I_{s,p}^t = I_p^t - I_s^t \;\forall\, p \in \bar{\eta}_s$.

In the Perona/Malik formulation, if $|\boldsymbol{\nabla} I| \gg \kappa$, then $c(|\boldsymbol{\nabla} I|) \rightarrow 0$ (sometimes called an all-pass filter), and if $|\boldsymbol{\nabla} I| \ll \kappa$, then $c(|\boldsymbol{\nabla} I|) \rightarrow 1$ (isotropic diffusion, or Gaussian filtering). The idea is to set the diffusion coefficient as a variable that is dependent upon image brightness gradients, so that little diffusion occurs in regions having large gradients with respect to κ, (i.e. at edges) and higher diffusion rates (i.e. noise removal and smoothing) occur elsewhere. In this way, edges are hopefully preserved with greater fidelity than that offered by simple constant diffusion rate equations.

4.2.2 Speckle Reducing Anisotropic Diffusion (SRAD)

Image noise generally comes in two different forms: *additive* noise and *multiplicative* noise. Additive noise (Gaussian white noise) is the type referred to in the beginning of Section 4.2, whereby a normal distribution of noise having zero mean is superimposed onto the uncorrupted image signal $s(\boldsymbol{x}, t)$:

$$I(\boldsymbol{x}, t) = s(\boldsymbol{x}, t) + N_{add}(\boldsymbol{x}, t). \tag{4.6}$$

Multiplicative noise, also known as speckle, shows up in imaging modalities such as ultrasound and radar, and takes the form

$$I(\boldsymbol{x}, t) = s(\boldsymbol{x}, t) \cdot N_{mult}(\boldsymbol{x}, t). \tag{4.7}$$

As can be seen in the above equation, this speckle noise is directly proportional to the grey level of the original signal; it increases with signal intensity, and so arises most prominently in regions of high reflectivity where backscattering and random fluctuations in the return signal are most pronounced [71].

While the anisotropic diffusion method proposed by Perona and Malik offers an improvement over isotropic diffusion in the way of edge preservation in the presence of additive Gaussian noise, it does not perform well on image signals corrupted with granular speckle patterns which can occupy several pixels and possess gradients that are similar in magnitude to those of bona fide edges. It also was found to suffer an inability to remove all of the outlying noisy pixels in some of the images studied here without setting κ to a value which was sufficiently large to cause the diffusion scheme to approach isotropic behavior and over-smooth edges.

In an effort to address these problems, Yu and Acton introduced further improvements to the Perona-Malik algorithm by setting the diffusion rate to be dependent not only on intensity gradients, but also on the Laplacian of local intensity values. In their so-called *Speckle Reducing Anisotropic Diffusion* (SRAD) technique, the diffusion equation takes the familiar form

$$\begin{cases} \frac{\partial I(x,y;t)}{\partial t} = \nabla \cdot [c(q) \cdot \nabla I(x,y;t)] \\ I(x,y;0) = I_o(x,y), (\partial I(x,y;t)/\partial \boldsymbol{n})|_{\partial\Omega} = 0 \end{cases}, \qquad \textbf{(4.8)}$$

with $\partial\Omega$ denoting the border of Ω, $\boldsymbol{n}$ representing the outer normal vector to $\partial\Omega$, but with the diffusion rate now calculated as

$$c(q) = \frac{1}{1+[q^2(x,y;t)-q_o^2(t)]/[q_o^2(t)(1+q_o^2(t))]} \qquad \textbf{(4.9)}$$

or

$$c(q) = \exp\{-[q^2(x,y;t) - q_o^2(t)]/[q_o^2(t)(1+q_o^2(t))]\}, \qquad \textbf{(4.10)}$$

which has a dependence on a new variable, $q(x,y;t)$, called the instantaneous coefficient of variation:

$$q(x,y;t) = \sqrt{\frac{\left(\frac{1}{2}\right)\left(\frac{|\nabla I|}{I}\right)^2 - \left(\frac{1}{4^2}\right)\left(\frac{\nabla^2 I}{I}\right)^2}{\left[1+\left(\frac{1}{4}\right)\left(\frac{\nabla^2 I}{I}\right)\right]^2}}. \qquad \textbf{(4.11)}$$

The quantity $q(x,y;t)$ is large where there is a large gradient magnitude in the image being smoothed, leading to a small diffusion coefficient $c(q)$. In the presence of strong speckling, which is marked by a large Laplacian values, $q(x,y;t)$ becomes small and $c(q)$ increases. The quantity $q_o(t)$ is dubbed the speckle scale function, and is approximated as

$$q_o(t) \approx q_o \exp[-\rho t]\ . \tag{4.12}$$

In the speckle scale function, ρ is a constant and q_o is the speckle coefficient of variation in the observed image. In a manner analogous to the setting of κ in the anisotropic diffusion equation, setting q_o to larger values will generally yield more noise reduction, but at the expense of edge preservation. For fully developed speckle such as seen in ultrasound data, it is suggested to set $q_o = 1$; for partially correlated data, $q_o < 1$.

In numerical form, gradients are separated into right (forward difference) and left (backward difference) (R and L respectively) components so that their magnitude may be captured accurately, and discretized on a grid with mesh spacing h in the following fashion:

$$\nabla_{\mathrm{R}} I_{i,j}^n = \left[\frac{I_{i+1,j}^n - I_{i,j}^n}{h}, \frac{I_{i,j+1}^n - I_{i,j}^n}{h}\right], \tag{4.13}$$

$$\nabla_{\mathrm{L}} I_{i,j}^n = \left[\frac{I_{i,j}^n - I_{i-1,j}^n}{h}, \frac{I_{i,j}^n - I_{i,j-1}^n}{h}\right], \tag{4.14}$$

$$\nabla^2 I_{i,j}^n = \frac{I_{i+1,j}^n + I_{i-1,j}^n + I_{i,j+1}^n + I_{i,j-1}^n - 4I_{i,j}^n}{h^2}\ . \tag{4.15}$$

Symmetric boundary conditions are set on the edges of the image domain. The discrete diffusion coefficient is calculated according to

$$c_{i,j}^n = c\left[q\left(\frac{1}{I_{i,j}^n}\sqrt{\left|\nabla_{\mathrm{R}} I_{i,j}^n\right|^2 + \left|\nabla_{\mathrm{L}} I_{i,j}^n\right|^2}, \frac{1}{I_{i,j}^n}\nabla^2 I_{i,j}^n\right)\right], \tag{4.16}$$

and the divergence of $c \cdot \nabla I$ is calculated as

$$d_{i,j}^n = \frac{1}{h^2}\left[c_{i+1,j}^n\left(I_{i+1,j}^n - I_{i,j}^n\right) + c_{i-1,j}^n\left(I_{i-1,j}^n - I_{i,j}^n\right) + c_{i,j+1}^n\left(I_{i,j+1}^n - I_{i,j}^n\right) + c_{i,j-1}^n\left(I_{i,j-1}^n - I_{i,j}^n\right)\right]. \tag{4.17}$$

Again, symmetric boundary conditions are applied at domain edges. The image is then updated with the *SRAD update function* and evolved iteratively:

$$I_{i,j}^{n+1} = I_{i,j}^{n} + \frac{\Delta t}{4} d_{i,j}^{n} \tag{4.18}$$

4.2.3 3D Speckle Reducing Anisotropic Diffusion

In 2004, Acton et al. revisited their method and extended it to three dimensions for the purpose of being able to denoise 3-D ultrasound images. The method is nearly identical to the 2-D SRAD algorithm, but with $q(x, y, z; t)$ and the update equation scaled appropriately to reflect the addition of the third dimension. The PDE still takes the form

$$\begin{cases} \frac{\partial I(x,y,z;t)}{\partial t} = \mathbf{\nabla} \cdot [c(q) \cdot \mathbf{\nabla} I(x, y, z; t)] \\ I(x, y, z; 0) = I_o(x, y, z), (\partial I(x, y, z; t)/\partial \boldsymbol{n})|_{\partial\Omega} = 0 \end{cases}, \tag{4.19}$$

with

$$c(q) = \frac{1}{1+[q^2(x,y,z;t)-q_0^2(t)]/[q_0^2(t)(1+q_0^2(t))]} \tag{4.20}$$

and

$$q(x, y, z; t) = \sqrt{\frac{\left(\frac{1}{3}\right)\left(\frac{|\nabla I|}{I}\right)^2 - \left(\frac{1}{6^2}\right)\left(\frac{\nabla^2 I}{I}\right)^2}{\left[1+\left(\frac{1}{6}\right)\left(\frac{\nabla^2 I}{I}\right)\right]^2}}. \tag{4.21}$$

4.2.4 Discrete form of 3D Speckle Reducing Anisotropic Diffusion

For 3-D SRAD, derivatives are discretized in a fashion analogous to that in the 2-D method:

$$\mathbf{\nabla}_{\mathrm{R}} I_{i,j,k}^{n} = \left[\frac{I_{i+1,j,k}^{n} - I_{i,j,k}^{n}}{h}, \frac{I_{i,j+1,k}^{n} - I_{i,j,k}^{n}}{h}, \frac{I_{i,j,k+1}^{n} - I_{i,j,k}^{n}}{h}\right], \tag{4.22}$$

$$\nabla_{\mathrm{L}} I_{i,j,k}^{n} = \left[\frac{I_{i,j}^{n} - I_{i-1,j,k}^{n}}{h}, \frac{I_{i,j}^{n} - I_{i,j-1,k}^{n}}{h}, \frac{I_{i,j}^{n} - I_{i,j,k-1}^{n}}{h}\right], \tag{4.23}$$

$$\nabla^2 I_{i,j,k}^{n} = \frac{I_{i+1,j,k}^{n} + I_{i-1,j,k}^{n} + I_{i,j+1,k}^{n} + I_{i,j-1,k}^{n} + I_{i,j,k+1}^{n} + I_{i,j,k-1}^{n} - 6I_{i,j,k}^{n}}{h^2}; \tag{4.24}$$

with symmetric boundary conditions similarly applied at the boundaries of the 3-D image set. The diffusion coefficient is calculated according to

$$c_{i,j,k}^{n} = c\left[q\left(\frac{1}{I_{i,j,k}^{n}}\sqrt{\left|\nabla_{\mathrm{R}} I_{i,j,k}^{n}\right|^2 + \left|\nabla_{\mathrm{L}} I_{i,j,k}^{n}\right|^2}, \frac{1}{I_{i,j,k}^{n}}\nabla^2 I_{i,j,k}^{n}\right)\right], \tag{4.25}$$

and then the divergence of $c \cdot \nabla I$ is calculated as

$$d_{i,j,k}^{n} = \frac{1}{h^2}\left[c_{i+1,j,k}^{n}\left(I_{i+1,j,k}^{n} - I_{i,j,k}^{n}\right) + c_{i-1,j,k}^{n}\left(I_{i-1,j,k}^{n} - I_{i,j,k}^{n}\right) + c_{i,j+1,k}^{n}\left(I_{i,j+1,k}^{n} - I_{i,j,k}^{n}\right) + c_{i,j-1,k}^{n}\left(I_{i,j-1,k}^{n} - I_{i,j,k}^{n}\right) + c_{i,j,k+1}^{n}\left(I_{i,j,k+1}^{n} - I_{i,j,k}^{n}\right) + c_{i,j,k-1}^{n}\left(I_{i,j,k-1}^{n} - I_{i,j,k}^{n}\right)\right], \tag{4.26}$$

again, with symmetric boundary conditions applied. The image is then updated with the *3DSRAD update function*, with the divergence now applied to a volume:

$$I_{i,j,k}^{n+1} = I_{i,j,k}^{n} + \frac{\Delta t}{6} d_{i,j,k}^{n}. \tag{4.27}$$

4.2.5 Wavelet Based Multiscale Anisotropic Diffusion (WMSAD)

Recently, Zhong and Sun proposed to improve upon anisotropic diffusion even further by first casting the image to be denoised into wavelet space before applying diffusion. While anisotropic diffusion in image brightness space has been shown to be effective at reducing noise, Zhong and Sun claim that it is still too diffusive in the presence of large amounts of noise due to the lack of a reliable edge-stopping criterion, and that more reliable edge-stopping can be effected in the wavelet domain due to its multi-resolution properties. Starting with the Perona/Malik anisotropic diffusion formulation, with the source term scaled by a stability constant λ rather than by the time step size Δt [72]

$$(I_s^{t+\Delta t}) = (I_s^t) + \frac{\lambda}{|\overline{\eta}_s|}\sum_{p\in\overline{\eta}_s} c\left(\nabla\left(I_{s,t}^t\right)\right)\nabla\left(I_{s,t}^t\right), \quad \textbf{(4.28)}$$

Zhong and Sun make use of a new diffusion rate coefficient with more robust edge-stopping characteristics, defined by Black et al.:

$$c(|\nabla I|,\sigma) = \begin{cases} \frac{1}{2}\left[1-\left(\frac{|\nabla I|}{\sigma}\right)^2\right]^2, |\nabla I| \leq \sigma \\ 0,\ \text{otherwise} \end{cases} . \quad \textbf{(4.29)}$$

In Equation 4.29, the variable σ is called a “scale parameter” and represents some threshold about the image gradient magnitude $|\nabla I|$, and is set to

$$\sigma = \sqrt{5}\sigma_e, \quad \textbf{(4.30)}$$

where σ_e denotes a “robust statistical scale”

$$\sigma_e = 1.42826\ \text{MAD}(\nabla I)\ . \quad \textbf{(4.31)}$$

In Equation 4.31, MAD denotes the mean average deviation:

$$\text{MAD}(\nabla I) = \text{median}_i(|\nabla I - \text{median}_i(|\nabla I|)|)\ . \quad \textbf{(4.32)}$$

Once the robust scale σ_e and the scale parameter σ are known, the anisotropic diffusion stability coefficient setting the strength of the source term is calculated as

$$\lambda = \frac{1}{c(\sigma_e,\sigma)}\ . \quad \textbf{(4.33)}$$

In [73], it is claimed that the diffusion algorithm acting on the image transformed to image space can be made even more stationary – that is, made so that the image set’s mean and variance do not change with time or position – by smoothing the finest scale of wavelet coefficients with a minimum mean-squared error (MMSE) based filter before applying anisotropic diffusion. Writing a noisy image transformed into wavelet space as

$$w_{2^j}^k I_n(x,y) = w_{2^j}^k I(x,y) + w_{2^j}^k n(x,y), \tag{4.34}$$

with $w_{2^j}^k I(x,y)$ denoting the wavelet coefficient of a noise-free image at location (x,y) and scale 2^j in spatial direction k, $w_{2^j}^k I_n(x,y)$ representing the wavelet coefficient of the noisy image, and $w_{2^j}^k n(x,y)$ representing the wavelet coefficient of zero-mean and σ_n^2-variance additive Gaussian white noise, the local variance in the spatial neighborhood of noisy wavelet coefficients can be estimated as

$$\hat{\sigma}^2 = \mathrm{MAX}\left\{0, \frac{1}{|\bar{\eta}_s|} \sum_{p\in\bar{\eta}_s} \left[\left(w_2^k I_n(x,y)\right)^2 - \sigma_n^2\right]\right\}. \tag{4.35}$$

In Equation 4.35, the noise standard deviation is estimated, using robust statistics, as

$$\hat{\sigma}_n = \frac{\mathrm{median}(|w(I)|)}{0.6745}, \tag{4.36}$$

so that the noise-free wavelet coefficient values at the finest scale may be estimated as

$$\widehat{w}_2^k I(x,y) = \frac{\hat{\sigma}^2}{\hat{\sigma}^2 + \hat{\sigma}_n^2} w_2^k I_n(x,y). \tag{4.37}$$

With the wavelet coefficients filtered at the finest scale, anisotropic diffusion is applied to the wavelet coefficients at the four finest levels.

$$w_{2^j}^k(I_s^{t+\Delta t}) = w_{2^j}^k(I_s^t) + \frac{\lambda}{|\bar{\eta}_s|} \sum_{p\in\bar{\eta}_s} c\left(\nabla w_{2^j}^k\left(I_{s,p}^t\right), \sigma\right) \nabla w_{2^j}^k\left(I_{s,p}^t\right) \tag{4.38}$$

Finally, Zhong and Sun propose a method for adaptively setting the stability coefficient λ based on what they refer to as the interscale ratio, denoted γ, of the sum of absolute wavelet coefficients (SAWC):

$$\gamma = \frac{N_{2^{j+1}}^k w(x_o, y_o)}{N_{2^j}^k w(x_o, y_o)}, \tag{4.39}$$

where the SAWC, N, is an operator classifying wavelet coefficients at the same scale within a *cone of influence* (COI) of a point (x_o, y_o) (Figure 4-7):

$$N_{2^j}^{k} w(x_o, y_o) = \sum |w(I)|, (x, y) \in \mathrm{COI}(x_o, y_o). \quad \textbf{(4.40)}$$

This simply means that any feature that shows up as a large wavelet coefficient at level $j+1$ in wavelet space should also show up as a set of large wavelet coefficients within the $2^{dimension}$ pixels/voxels that correspond to the same imaged region at level j.

The stability coefficient of diffusion in wavelet space, λ, is set to a value that represents texture, edges, or regions otherwise determined to be desirable to keep in consideration for the final segment if γ is larger than some experimentally set threshold. Otherwise, λ is set to a value that reflects noise, so that the impact of the noisy pixel is minimized. In the paper describing this method of delineation, γ is experimentally set to a value of $\sim$1.8. However, little guidance is offered as to how the upper and lower values of λ should be set.

Later in this chapter the relative capabilities of the anisotropic diffusion, SRAD and WMSAD techniques will be examined in detail and compared according to their performance with different types of images. These methods are evaluated based on their ability to remove additive and multiplicative noise with different PSNRs (peak signal to noise ratios) while preserving physical features (signal) of interest in the images. In addition, the ability of each technique to provide accurate segmentation of the embedded features in an image will be assessed.

4.3 Segmentation using Level Sets

It was mentioned in the introductory chapter that segmentation is the process of grouping objects together in a manner which mimics the abilities of the human visual system, whereby objects are classified into groups. This allows for the consideration of an object as something distinct from that which surrounds it, and provides the basic information necessary for constructing an embedded surface and applying desired boundary conditions to it in the context of modeling and flow computations. In this section, we will examine the segmentation process in detail and expand its capabilities within our framework beyond the simple approaches taken in the previous chapter.

4.3.1 Creating Active Contours through Functional Minimization

In 1989, Mumford and Shah proposed using the calculus of variations to decompose an image I_o containing image domain Ω into two piecewise-smooth approximations, separated by a segmenting curve or surface C, through minimization of the following functional:

$$F_{MS}(I,C) = \mu L(C) + \lambda \int_{\Omega} (I_o - I)^2 \, d\Omega + \int_{\Omega \setminus C} |\nabla I|^2 \, d\Omega. \qquad \textbf{(4.41)}$$

In this Mumford-Shah functional, as it has come to be known, $L(C)$ denotes the length of curve C, and $\lambda > 0$ and $\mu \geq 0$ are fixed weighting parameters set to give what is judged to be a good result on a given image field. The first term controls the length and smoothness of the segmentation curve, while the other terms separate the image into distinct regions in a manner that allows for discontinuities along the edges of each region.

Vese and Chan later expanded upon the Mumford and Shah method by proposing an algorithm for decomposing an image into two approximately piecewise-constant regions, determined by each region's mean intensity value:

$$F_{VC}(c_1, c_2, C) = \mu L(C) + \nu A(\Omega_1)$$
$$+\lambda_1 \int_{\Omega_1} |I_o - c_1|^2 \, d\Omega_1 + \lambda_2 \int_{\Omega_2} |I_o - c_2|^2 \, d\Omega_2. \qquad \textbf{(4.42)}$$

In this Vese/Chan functional, Ω_1 represents the part of an image domain Ω that lies inside of a segmentation curve C, $\Omega_2 = \Omega \backslash \Omega_1$ (the region outside of curve C), $A(\Omega_1)$ denotes the area of the region Ω_1 inside the curve, and $\lambda_1 > 0$, $\lambda_2 > 0$, $\mu \geq 0$ and $\nu \geq 0$ are again fixed weighting parameters which are tuned based on image characteristics for optimal results. The variables c_1 and c_2 are generally taken to represent average intensity values inside and outside of the segmentation contour ($\bar{I}_1$ and $\bar{I}_2$), respectively, and thus depend on the location of the contour itself.

As a convenient way to track interfacial motion, a segmentation curve may be regarded as the zero-level of a level set function that is defined over the image domain Ω, allowing a curve's length and contained area to be defined as

$$L(C) = L(\varphi = 0) = \int_\Omega |\nabla H(\varphi)| \, d\Omega = \int_\Omega \delta(\varphi) |\nabla \varphi| \, d\Omega, \qquad \textbf{(4.43)}$$

$$A(C) = A(\varphi \leq 0) = \int_\Omega H(\varphi) \, d\Omega. \qquad \textbf{(4.44)}$$

In the curve length and area equations, $\varphi = \varphi(x, y)$ (or $\varphi(x, y, z)$ in 3-D), H represents a discrete Heaviside function $H(x) = \frac{1}{2}\left[1 + \frac{2}{\pi} \arctan\left(\frac{x}{h}\right)\right]$ on a Cartesian grid with mesh spacing h, and δ is a discrete Dirac delta function $\delta(x) = \frac{d}{dx} H(x) = \frac{1}{\pi} \frac{h}{x^2 + h^2}$. Cast in this level set formulation, the Vese-Chan functional becomes

$$F_{VC}(c_1, c_2, \varphi) = \mu \int_\Omega \delta(\varphi) |\nabla \varphi| \, d\Omega + \nu \int_\Omega H(\varphi) \, d\Omega$$

$$+\lambda_1 \int_\Omega |I_o - c_1|^2\, H(\varphi) d\Omega$$

$$+\lambda_2 \int_\Omega |I_o - c_2|^2\, \big(1 - H(\varphi)\big) d\Omega, \qquad \textbf{(4.45)}$$

with inner and outer region averages defined as

$$c_1 = \frac{\int_\Omega I_o H(\varphi) d\Omega}{\int_\Omega H(\varphi) d\Omega},\ c_2 = \frac{\int_\Omega I_o \big(1 - H(\varphi)\big) d\Omega}{\int_\Omega \big(1 - H(\varphi)\big) d\Omega}. \qquad \textbf{(4.46)}$$

Euler-Lagrange minimization of the Vese/Chan functional leads to the level set evolution equation

$$\frac{\partial\varphi}{\partial t} = \delta(\varphi)\left[\mu\boldsymbol{\nabla}\cdot\left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}\right) - \lambda_1(I_0 - c_1)^2 + \lambda_2(I_0 - c_2)^2\right], \qquad \textbf{(4.47)}$$

which evolves an initial level set curve of arbitrary shape to convergence at some steady state result conforming to the boundary of the object being segmented. The main disadvantage of the Chan-Vese approach for segmenting the image is that it is rather time-consuming to evolve the level sets via the diffusion equation shown above. To address this issue, a modification to the approach that provides rather rapid segmentations was proposed by Gibou and Fedkiw.

4.3.2 Revisiting k-Means Clustering

Gibou and Fedkiw proposed to make the active contour models just described more direct by ignoring the first regularization term penalizing curve length, and setting the Dirac delta function $\delta = 1$ in order to extend level set information uniformly into the image domain, leading to the ODE introduced in the last chapter:

$$\frac{d\varphi}{dt} = V_{LS} = \lambda_1(I_0 - c_1)^2 - \lambda_2(I_0 - c_2)^2\,. \qquad \textbf{(4.48)}$$

As formulated, Equation 4.48 may be evolved in small steps just as the PDE proposed by Vese and Chan, with the level set field illustrated as a hypersurface evolving through the 2-D image space (Figure 4-2 and Figure 4-3). However, it is pointed out by Gibou and Fedkiw that the accuracy of level set curve evolution is not of concern to the process of segmentation as long as a desirable steady-state result is achieved at convergence. Thus, large time steps may be taken in updating the level set evolution equation (4.48). In fact, for the images segmented as part of this work, convergence was generally reached within 1-3 iterative steps using the simple update scheme

$$\varphi^{n+1} = \varphi^{n} + V_{LS}^{n+1}. \tag{4.49}$$

In addition to its speed, the method was also found to possess a nice combination of simplicity, robustness, and ability to yield consistently good results for our purposes, and so it was adopted for all of the segmentation work pertaining to this thesis.

The “level set” field that results from the Gibou/Fedkiw method is actually nothing but a scaled and shifted version of the original intensity field in the image domain; a pixel cast in level set space takes on a value that is maximally of the order of the square of the difference between its corresponding local intensity value and the average intensity of the region in which it is classified at a particular time, with its sign depending upon which region it belongs to. As the level set update equation approaches convergence, a pixel belonging to set Ω_1 (conventionally considered to be inside the segmentation curve of an object) will be closer in value to the inner average brightness c_1 than to the outer average brightness c_2, resulting in a negative level set value. Similarly, pixels belonging to the set Ω_2 will possess intensity values closer to c_2 than to c_1, giving $\varphi > 0$. (This matches the standard convention for level set fields employed throughout this work, in which negative level set values are considered to be “inside” an interface and positive level set values are considered to be “outside.”) The amount of zero-level curve bias toward the inside or outside of an imaged object is determined by the relative values of the weighting coefficients λ_i (Figure 4-5), which are generally tuned by the user to achieve results to their liking.

In their description of the algorithm, Gibou and Fedkiw suggest setting $|\varphi| = 1$, which leads to a piecewise binary level set field having values of ± 1 on which constants c_i may be efficiently computed as

$$c_1 = \frac{\sum I_o(\varphi+1)}{\sum(\varphi+1)},\ c_2 = \frac{\sum I_o(\varphi-1)}{\sum(\varphi-1)}\,. \tag{4.50}$$

This is the strategy originally adopted Chapter 3. Unfortunately, the information necessary for calculating the zero-level curve position at sub-pixel resolutions is not preserved with such a pixel "painting" scheme, resulting in all of the original image boundary smoothness being removed. Indeed, the Lagrangian method outlined in Chapter 3 required a considerable amount of point smoothing after segmentation in order to produce a smooth interface, largely for this very reason.

4.3.3 Interface Construction with Cell Cuts

One of the main points of this work is to dispense with Lagrangian interface treatments fraught with tedious point placement and smoothing algorithms, so we wish to obtain smooth object contours, without further manipulation, directly from the segmentation result. To achieve this, level set values are simply updated to convergence using Equations 4.45 and 4.46, then, rather than updating inside and outside coefficients using the binary scheme of Equation 4.47, cell cuts are used as a means to calculate the position of the zero-level curve generated with the Gibou/Fedkiw algorithm. This allows for the subsequent creation of a signed distance level set field within a narrow band defined about the imaged object's interface by using the fast marching method outlined in [24]. The cell-cut redistancing algorithm proceeds as follows:

1. Each grid point in the image domain is swept over to check whether it is an interfacial point. This is easily accomplished by multiplying the level set value of any grid point by that of any of its four cardinal (i.e. north, south, east, and west) neighbors; if the resulting number is negative, then an interfacial crossing must have occurred and the grid point in question lies adjacent to a boundary.
2. Each interfacial point's spatial location (x_p, y_p) is stored.
3. Considering (x_p, y_p) as cell-center locations, the number of cuts in each interfacial grid cell is determined by probing its four cardinal neighbors. The number of cuts is recorded and the distance from (x_p, y_p) to the zero-level interface is calculated using the appropriate method for the number of cuts found:
    a. Case 1- two cell cuts. This is the most straightforward case, as the two cell cut locations can be used directly to calculate the level set value at (x_p, y_p).
        i. First, the cell cut locations are calculated and stored:

$$x_1 = x_p + |x_p - x_{nbr1}| \frac{\varphi_p}{|\varphi_p| + |\varphi_{nbr1}|} h \,, \qquad \textbf{(4.51)}$$

$$y_1 = y_p + |y_p - y_{nbr1}| \frac{\varphi_p}{|\varphi_p| + |\varphi_{nbr1}|} h \,, \qquad \textbf{(4.52)}$$

$$x_2 = x_p + |x_p - x_{nbr2}| \frac{\varphi_p}{|\varphi_p| + |\varphi_{nbr2}|} h \,, \qquad \textbf{(4.53)}$$

$$y_2 = y_p + |y_p - y_{nbr2}| \frac{\varphi_p}{|\varphi_p| + |\varphi_{nbr2}|} h \,. \qquad \textbf{(4.54)}$$

        ii. Cell cut locations (x_1, y_1) and (x_2, y_2) are used to calculate an interpolation distance α:

$$\alpha = \frac{(x_p - x_1)(x_2 - x_1) + (y_p - y_1)(y_2 - y_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2} . \quad \textbf{(4.55)}$$

iii. The newly calculated interpolation distance is then used to find (x_n, y_n), the address of the points at the foot of the normal vector pointing from the interface to (x_p, y_p):

$$x_n = x_1 + \alpha(x_2 - x_1),\ \ y = y_1 + \alpha(y_2 - y_1) . \quad \textbf{(4.56)}$$

iv. Finally, the magnitude of the normal vector pointing from the interface to each of its adjacent points (x_p, y_p) is used to calculate an updated level set value $\hat{\varphi}$:

$$|\boldsymbol{n}| = \sqrt{(x_p - x_n)^2 + (y_p - y_n)^2} , \quad \textbf{(4.57)}$$

$$\hat{\varphi} = \mathrm{sign}(\varphi)|\boldsymbol{n}| . \quad \textbf{(4.58)}$$

b. Case 2- three cell cuts. If three sides of an interfacial cell are found to be cut, the closest (x_1, y_1) and (x_2, y_2) pair to (x_p, y_p) are used to find interfacial distance in the algorithm, which otherwise proceeds as above.

c. Case 3- one cell cut. In this case, the neighborhood window being probed for neighbor information is expanded to include the 4 points located in diagonal directions (i.e. northeast, southeast, northwest, and southwest) from (x_p, y_p). (x_1, y_1) are stored as in the other two cases, but (x_2, y_2) are calculated slightly differently to include the increase in distance brought by their diagonal positioning with respect to (x_p, y_p):

$$x_2 = x_p + |x_p - x_{nbr2}| \frac{\varphi_p}{|\varphi_p| + |\varphi_{nbr2}|} \sqrt{2}h , \quad \textbf{(4.59)}$$

$$y_2 = y_p + |y_p - y_{nbr2}| \frac{\varphi_p}{|\varphi_p| + |\varphi_{nbr2}|} \sqrt{2}h . \quad \textbf{(4.60)}$$

4. Once level set values are determined at all of the interfacial points, fast marching is used to extend the signed distance level set field from the interfacial points outward into a narrow band surrounding the segmentation curve (Figure 4-4). Complete details about the fast marching method can be found in e.g. [24, 30].

Up to this point, nearly the entire discussion about image segmentation has been limited to cases involving 2-D images. However, it is appropriate to note here that all of the aforementioned techniques have successfully been extended to three dimensions as part of this work; results will be shown in section 4.4 below. The 3-D approach is exactly the same, but with voxels located at (x_p, y_p, z_p) replacing pixels at (x_p, y_p), level set surfaces $\varphi(x, y, z)$ replacing level set curves $\varphi(x, y)$, and cell cuts being performed along faces of cells having 6 cardinal and 20 diagonal neighbors rather than through edges of cells having 4 cardinal and 4 diagonal neighbors.

4.4 Test Image Results

Each of the three primary denoising techniques outlined above were tested on the "Shapes" and "Lena" images with both added Gaussian white noise levels (four image cases). Results were qualified based on segmentability of the Shapes image, and quantified using the measure of *peak signal-to-noise ratio* for both:

$$PSNR = \frac{P_{MAX}}{P_{RMS}} \qquad \textbf{(4.61)}$$

In Equation 4.61, P_{MAX} refers to the maximum signal power possible in an image domain (255) and P_{RMS} is the root mean square signal power measured in the image. For reference, the $PSNR$ of the noisy "shapes" image is 18.3 for $\sigma = 30$ and 10.3 for

$\sigma = 100$, while the $PSNR$ for the "Lena" image is 18.7 for $\sigma = 30$ and 10.1 for $\sigma = 100$, before the introduction of any diffusive processing.

4.4.1 SRAD

The first denoising algorithm evaluated was 2-D Speckle Reducing Anisotropic Diffusion method of Yu and Acton. For each of the four noisy test images, q_o was set to both 1.0 and 0.25 in order to demonstrate its effect on the final result. In addition, the maximum number of iterations was set to both 100 and 500 for each q_o value on each image (convergence was set to 10^{-3}, and was often reached before the 500th iteration for the latter cases) in order to assess sensitivity to diffusion time, as well as the effectiveness of the edge-stopping criteria intrinsic to the method.

The results of applying SRAD with $q_o = 1.0$ (500 max iterations) are shown in Figure 4-10, with a 3-D rendering of the noisy Shapes images after smoothing shown in Figure 4-11. Visually, most of the speckle appears to have been removed from all four test images. $PSNR$ results on the Lena image also indicate successful denoising, with values increasing from 18.7 to 20.8 for $\sigma = 30$, and from 10.1 to 20.9 for $\sigma = 100$. However, $PSNR$ results indicate an actual degradation of image quality for the Shapes image with $\sigma = 30$, from 18.3 to 11.7, and a marginal increase for Shapes" with $\sigma = 100$, from 10.3 to 15.0.

Setting q_o to ¼ of this value (0.25) yielded much different results both qualitatively and quantitatively. For example, $PSNR$ for Shapes with $\sigma = 30$ increased significantly from 18.3 to 24.4, with marginal increases for the other 3 image cases. Visually, however, the improvement is scarcely detectable for any of them.

4.4.2 Regular Anisotropic Diffusion

The original anisotropic diffusion algorithm of Perona and Malik was tested next as a way to assess the relative improvements offered by SRAD and WMSAD. The diffusion equation was again allowed to run for a maximum of 500 diffusive time steps, with a convergence tolerance set to 10^{-3}. The edge magnitude parameter κ was set to both 30 and 100 for each of the images having a noise $\sigma = 30$ in order to evaluate its effect.

Interestingly, the original anisotropic diffusion gave some of the best results of any method tested, at least when comparing only the measures of peak signal-to-noise ratio. With $\kappa = 30$, $PSNR$ was calculated to be 19.6 for both the Lena and Shapes images, while $PSNR$ was measured to be 28.9 and 25.3 when $\kappa = 100$ for Shapes and Lena, respectively (Figure 4-13). Of course, it should be pointed out that these test images were corrupted with additive noise rather than speckle, so anisotropic diffusion should not be expected to perform poorly. Nonetheless, the addition of the curvature-dependent term in SRAD was found to enable the rapid diffusion of isolated noisy pixels while preserving edges defined by large gradients.

4.4.3 Wavelet Based Multiscale Anisotropic Diffusion

The final denoising algorithm tested on the Shapes and Lena images was the WMSAD method of Zhong and Sun. For these tests, several different tunable parameters had to be selected due to the inherent complexity of the method. Recall that the sum of absolute wavelet coefficients (SAWC), computed within the cone of influence of several different wavelet coefficient levels, is used to calculate an interscale ratio γ, which is in turn used to set the stability coefficient λ which scales the diffusion rate of the denoising PDE acting on the image in wavelet space. Experimentally, γ was given a few different threshold values, including 1.8, the one suggested by the authors. The resulting high and low values for λ were also varied, with some of the results summarized in Table 4-1 and Table 4-2. However, it was found that results were nearly identical when the stability coefficient was simply calculated as in Equation 4.30, obviating the need to find an interscale ratio or tune its threshold to select between two different stability coefficient values (which are also themselves tunable to image properties). The greatest difference was actually made through wavelet selection. Daubechies-20 wavelets gave the best quantitative results in this investigation, but it is notable that both Daub20 and Daub4 wavelet types produced good qualitative image denoising (Figure 4-14). Pertinent results from all of the test image denoising cases are summarized in Table 4-1 and Table 4-2.

Although the $PSNR$ results give an indication of perceived visual quality, it was found that they predict segmentability of a denoised image quite poorly. Figure 4-15 shows some of the segmentation results of the denoised Shapes image. SRAD with $q_o = 1.0$ gave some of the lowest $PSNR$ values out of any of the test cases, yet it also produced the only acceptable segmentation results. All other methods retained noise in the final segments, making them unusable for modeling purposes.

Figure 4-16 and Figure 4-17 help to illustrate this point, comparing SRAD to regular anisotropic diffusion and WMSAD in 3-D renderings. WMSAD clearly retains a great deal of noise, which would require some sort of further processing to remove and get the image into a state suitable for segmentation.

Table 4-1.PSNR results of denoising methods applied to the Lena test image.

Image	Method	$Iters_{max}$	ISR	λ_1	λ_2	Wavelet	σ	q_o	κ	PSNR
Lena	SRAD	100	-	-	-	-	30	1	-	22.2
							100			21.7
							30	0.25		21.4
							100			10.6
		500					30	1		20.8
							100			20.9
							30	0.25		21.4
							100			10.6
	WM-SAD	100	1.8	5	0.1	Daub4	30	-	-	25.5
				5			100			17.2
				0.25			30			25.7
			0.1	5			30			25.7
			1							25.7
			10							25.7
			auto	-	-	Daub4	30			25.7
						Daub20	30			26.3
						Daub4	100			17.1
	RAD	100	-	-	-	-	30	-	30	19.2
									100	25.3
		500							30	19.6

The original anisotropic diffusion scheme actually provides very good results overall, but was unable to remove all of the outlying "speckles" (seen as spikes in Figure 4-17), illustrating the usefulness of Yu and Acton's Speckle Reducing Anisotropic Diffusion method.

Table 4-2. PSNR results of denoising methods applied to the Shapes test image.

Image	Method	Iters$_{max}$	ISR	λ_1	λ_2	Wavelet	σ	q$_o$	κ	PSNR
Shapes	SRAD	100	-	-	-	-	30	1	-	12.4
Shapes	SRAD	100	-	-	-	-	100	1	-	16.5
Shapes	SRAD	100	-	-	-	-	30	0.25	-	24.4
Shapes	SRAD	100	-	-	-	-	100	0.25	-	11.0
Shapes	SRAD	500	-	-	-	-	30	1	-	11.7
Shapes	SRAD	500	-	-	-	-	100	1	-	15.0
Shapes	SRAD	500	-	-	-	-	30	0.25	-	24.4
Shapes	SRAD	500	-	-	-	-	100	0.25	-	11.0
Shapes	WM-SAD	100	1.8	5	0.1	Daub4	30	-	-	18.6
Shapes	WM-SAD	100	1.8	5	0.1	Daub4	100	-	-	16.8
Shapes	WM-SAD	100	1.8	0.25	0.1	Daub4	30	-	-	18.7
Shapes	WM-SAD	100	0.1	5	0.1	Daub4	30	-	-	18.7
Shapes	WM-SAD	100	1	5	0.1	Daub4	30	-	-	18.7
Shapes	WM-SAD	100	10	5	0.1	Daub4	30	-	-	18.6
Shapes	WM-SAD	100	auto	-	-	Daub4	30	-	-	18.7
Shapes	WM-SAD	100	auto	-	-	Daub20	30	-	-	17.5
Shapes	WM-SAD	100	auto	-	-	Daub4	100	-	-	16.7
Shapes	RAD	100	-	-	-	-	30	-	30	19.0
Shapes	RAD	100	-	-	-	-	30	-	100	28.9
Shapes	RAD	500	-	-	-	-	30	-	30	19.6

So far, SRAD appears to be the superior method of those tested, at least for the purpose of segmenting the simple test image generated here. In addition to removing speckle, it also possesses an inbuilt contrast enhancement feature as regions away from edges continue to diffuse toward a homogeneous steady state. This results in low *PSNR* values, but it also results in crisp edges that are favorable to reliable segmentation.

4.5 Real Image Results

Test images such as those discussed to this point were quite useful in the development and preliminary testing of our image processing algorithms, but actual images in two and three dimensions are this work's real impetus. Thus, this chapter concludes with an evaluation of SRAD and WMSAD applied to two different data sets: The first is a set of CT slices acquired of a cylindrical concrete sample (provided by personnel at Eglin Air Force base), and the second is an ultrasound image of a liver, taken from Philips' ultrasound image library available online [74].

4.5.1 CT Image Slices of Concrete

This first image set, a series of 679 CT slices of concrete taken at 1 mm intervals, with dimensions of 897 $\times$ 832 pixels, was acquired as part of an ongoing research effort for the USAF (one slice is shown in Figure 4-18 as an example), and denoised and segmented using either SRAD with $q_o = 0.25$ / $q_o = 1.0$, or WMSAD with Daubechies-4 wavelets and $\lambda = \frac{1}{c(\sigma_e, \sigma)}$. It proved to be an excellent developmental tool, as it challenged denoising methods with large amounts of noise that had as much variation as there was contrast in the baseline image intensity, and it provided a means of testing the methods being developed as they were extended into 3-D.

All of the denoising methods were first tested on a single 2-D slice before moving on to smoothing and segmenting the entire volume. A small code was written in Matlab to crop the original images so that only a square region would remain about the center of the concrete sample, removing the empty black space outside the cylindrical specimen.

The cropped image was also re-scaled to have dimensions of 256×256 pixels, for the purpose of eliminating excessive computation time and data storage during testing.

Figure 4-19 (a) shows the cropped and re-scaled original concrete slice before denoising, while Figure 4-19 (b-c) show the results of denoising with SRAD $q_o = 1.0$, SRAD $q_o = 0.25$, and WMSAD Daub4, respectively. It can be seen that all of the methods visually appear to have increased contrast and diminished noise compared to the original, with SRAD $q_o = 0.25$ and WMSAD appearing to blur edges the least and retain most of the original visual information. However, subsequent segmentation results (Figure 4-20) reveal that WMSAD was unable to remove a sufficient amount of noise to produce an acceptable segment.

In Figure 4-20 the segmentation curves are shown superimposed onto the original image slice. Both SRAD cases produced usable segments, but the $q_o = 0.25$ case clearly generated curves that follow the original object contours with the greatest amount of fidelity; the $q_o = 1.0$ SRAD case diffused the boundaries too much to be segmented accurately. While the WMSAD case did a better job than SRAD $q_o = 1.0$ at preserving crisp edges, it also retained too much noise and thus produced unusable segmentation results. This is once again further elucidated with 3-D renderings of the level set segmentation fields, which are provided in Figure 4-21. While offering a significant reduction of noise over the original concrete slice image, WMSAD clearly does not possess the characteristic smoothness of either SRAD result.

4.5.2 Extension to Three Dimensions

For segmenting and denoising the concrete image data set in 3-D, the 2-D slices were cropped to have dimensions of 128×128 pixels, again for the purpose of saving computation time. In addition, only 128 slices were kept out of the original 679, by selecting every 5th slice from a set of 640. The result was a cubic image of dimension $128 \times 128 \times 128$ voxels. SRAD $q_o = 1.0$, SRAD $q_o = 0.25$, and WMSAD Daub4 were once again used to denoise the image set and test their capabilities, but this time using 3-D versions of each.

Figure 4-22 (a) shows an example slice (45 of 128) before processing, with Figure 4-22 (b-d) showing the different segmentation results for the same slice. Immediately notable is the fact that SRAD appears to have become considerably more diffusive with the addition of gradient information in the z-direction (image slice normal direction); with SRAD $q_o = 1.0$, image intensity has become sufficiently diffused to eliminate most of the edges delineating objects from their surroundings, making the objects unrecognizable. SRAD $q_o = 0.25$ still performs well, but with perhaps less fidelity than in the 2-D case. WMSAD seems to yield much better results when applied to a 3-D volume compared with its 2-D performance, but still lacks in boundary smoothness in the segment.

Figure 4-23 (b-d) shows the final 3-D segmentation result in its entirety for each of the three test cases, with Figure 4-23 (a) illustrating isosurfaces in the original unprocessed image set. The axes have been limited to range from 20 to 100 in each direction for visual clarity. SRAD with $q_o = 1.0$ is clearly much too diffuse to be usable in this particular image set, with SRAD $q_o = 0.25$ and WMSAD giving the best results qualitatively. WMSAD actually appears to give superior results with respect to keeping all of the imaged objects separate from each other – SRAD with $q_o = 0.25$ has allowed some object surfaces to merge – but still requires additional smoothing to make it amenable to modeling.

4.5.3 Ultrasound Images

Ultrasound imaging holds a great deal of promise for image-based modeling, as ultrasound machines are portable, they do not emit radiation, and they possess the ability to rapidly acquire data in real time. Unfortunately, the images they produce are also of low visual quality compared to other modalities, and so denoising becomes an especially important challenge. Still, they may hold the key to 4-D modeling in the near term. Therefore, as a final test of the methods discussed herein, an ultrasound image was acquired from Philips' online library of ultrasound images, and denoised with SRAD and WMSAD prior to segmentation. The ultrasound test image chosen is of a normal adult liver, and is illustrated in Figure 4-24.

Figure 4-25 shows the denoising and segmentation results, with SRAD results in Figure 4-25 (a) and (b), and WMSAD results in Figure 4-25 (c) and (d). The best SRAD result on this sonograph was obtained by setting $q_o = 1.0$, even though this relatively high value for the speckle coefficient proved to effect too much diffusion on the CT images examined previously. This could be partly due to the fact that the CT images actually do not contain speckle, but rather are dominated by additive Gaussian noise. Ultrasound images, on the other hand, contain fully developed speckle noise. The liver sonograph in particular features objects of interest that have low intensity values, so their edges were well preserved while rapid diffusion toward homogeneity occurred elsewhere. While some of the smaller features were diffused out of the image and thus missed by the segmentation algorithm, the large containing vessel that features prominently in the sonograph was essentially unaltered. Figure 4-26 illustrates the human liver along with some of its vascular and biliary structures. Examining this figure, it appears likely that the less prominent dark regions in the sonograph correspond to cross-sectional views of other vessels oriented at least partially out-of-plane; such structures may be retained in a 3-D sonograph with the introduction of gradient and curvature terms in the third direction.

As with other cases, WBAD with a Daubechies-4 wavelet did denoise the image considerably, and perhaps gave results superior to those given by SRAD from a pure human visualization standpoint, but its level of noise removal was not sufficient to be useful for segmentation purposes.

4.6 Conclusions

A variety of segmentation and denoising techniques were discussed, each possessing their own advantages within the field of image analysis. The rapid k-means segmentation approach outlined by Gibou and Fedkiw (modified with cell cuts to give smooth contours) was adopted for this work because of its computational efficiency and reliability on smooth image fields; we were able to get consistently good results for each of the test cases investigated, and quickly. However, it should be kept in mind that neglecting the regularization terms in the full evolution equations of Vese and Chan may have been partially responsible for this method’s lack of performance on noisy images, thus causing it to rely heavily upon the success of preprocessing algorithms.

For the purposes of image-based modeling, that is, the ability to generate usable segments from noisy images, the Speckle Reducing Anisotropic Diffusion methods of Yu and Acton in 2-D, and of Acton et al. in 3-D, proved to be the most consistently reliable from the standpoint of producing segmentable images that could easily be used as modeling bases. However, wavelet-based methods still hold a great deal of promise and deserve further investigation, especially in light of the 3-D concrete results discussed in the previous section. For instance, only Daubechies wavelets were considered in this work; other wavelet types may be better suited to the task of image noise removal, especially where smooth boundaries without speckle are desired. Filtering wavelet coefficients at different scales was also considered only briefly, so much future work still remains in that regard.

With denoising and smooth segmentation facilities established, the focus of our attention now turns to embedding the results of these facilities into a computational domain for modeling purposes. Segmentation simply defines contours; the next order of business is to go one step further and quantify how those contours are behaving, so that we may define a complete set of boundary conditions. For this, we turn to optical flow, the subject of the next two chapters.

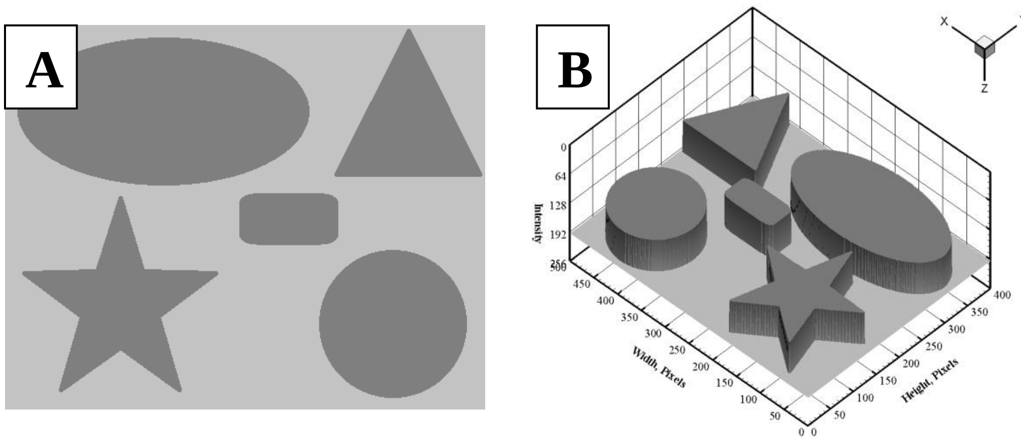


Figure 4-1. “Shapes” test image (A), and a 3-D rendering of it (B). The intensity scale in (B) is reversed for visual clarity.

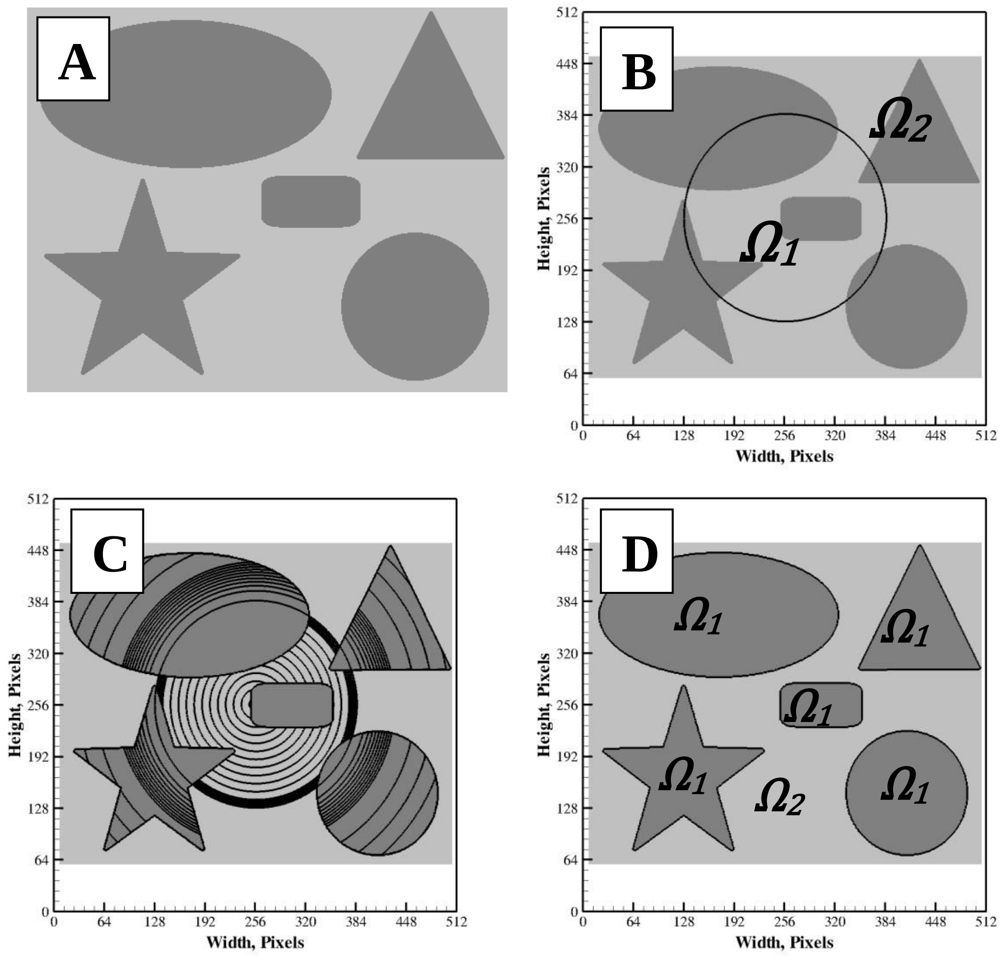


Figure 4-2. Segmentation of the Shapes image (A) is performed by initializing an arbitrary level set field in the image space (B), and evolving it (C) until the features of interest are segmented (D). (Zero-levels are shown for clarity.) Segmentation was achieved in 24 adaptive pseudo time steps for illustration here, but in reality, the method requires very few (1-3) steps of arbitrary size.

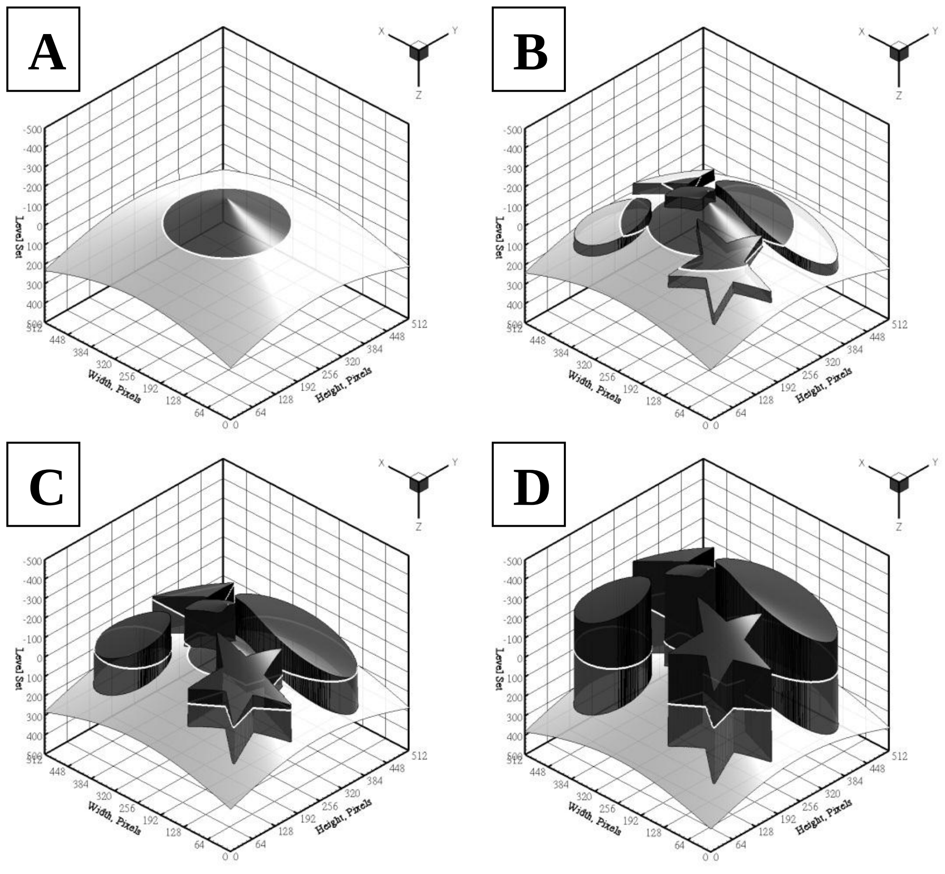


Figure 4-3. 3D representation of the k-Means segmentation process: (A) Initial level set; (B) level set after evolving for 10 pseudo time steps; (C) after 15 pseudo time steps; (D) after 24 pseudo time steps, with segmentation complete. (Zero-levels are illustrated with a white line for clarity.) Time step size was adapted based on maximum rate of change of the level set field, with $\Delta T \sim O(10^{-3})$. In reality, segmentation can be achieved in very few (1-3) pseudo time steps, with arbitrarily large time step sizes.

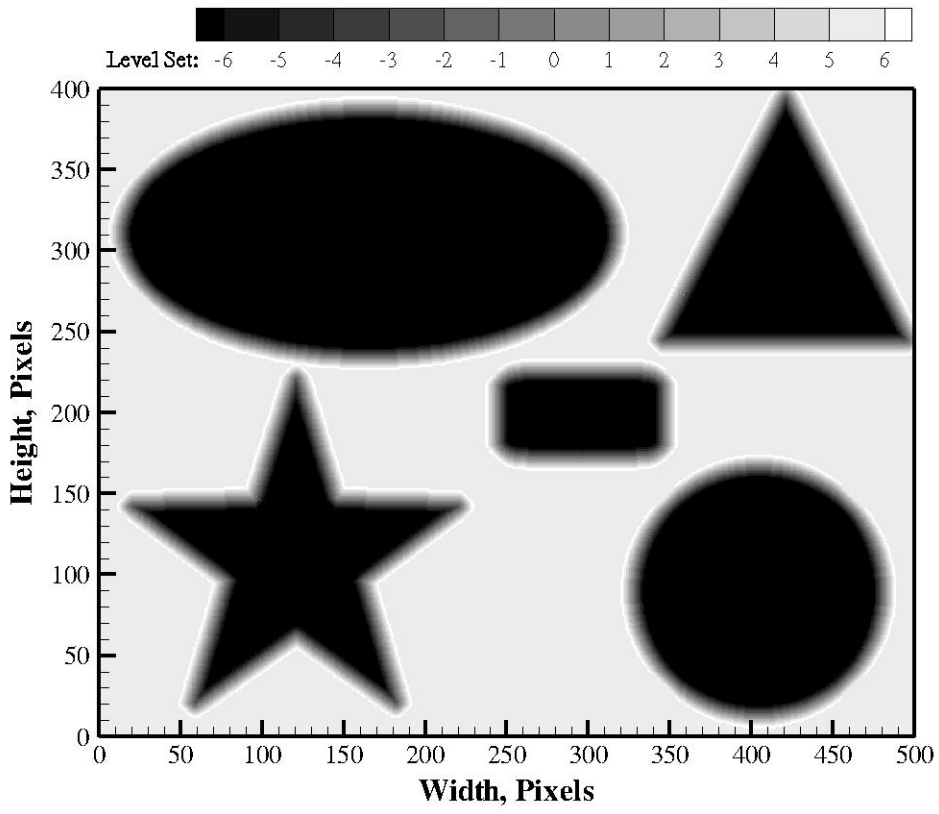


Figure 4-4. After segmentation, a narrow band level set field is constructed about the segmentation curve (zero-level). Inside the narrow band, the level set function is a signed distance function extending 6 units length from either side of the interface. The rest of the field outside the narrow band is assigned a value of +/- 6 depending on which side of the interface it lies on.

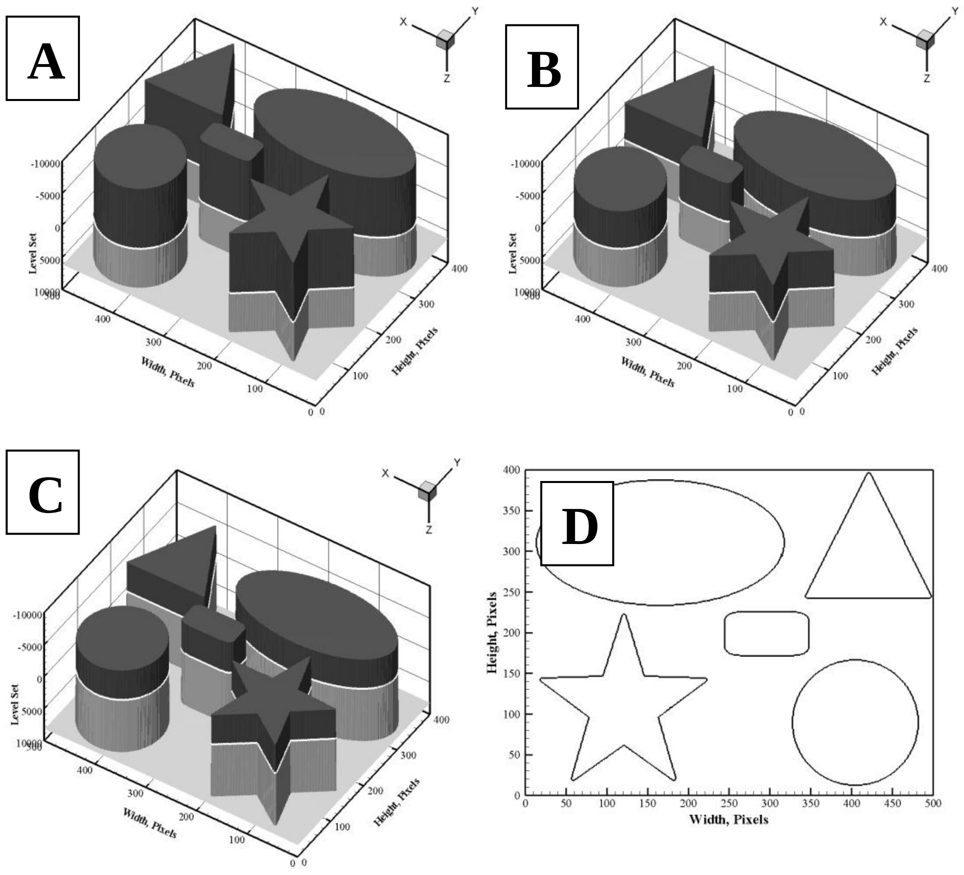


Figure 4-5. “Shapes” test image represented as level sets using the method of Gibou and Fedkiw (level set axes reversed for visual clarity): (A) $\lambda_1 = 1.0, \lambda_2 = 1.5$; (B) $\lambda_1 = 1.0, \lambda_2 = 1.0$; (C) $\lambda_1 = 1.5, \lambda_2 = 1.0$. Given the idealized contrast in the test image, all three cases lead to the same segmentation result (D).

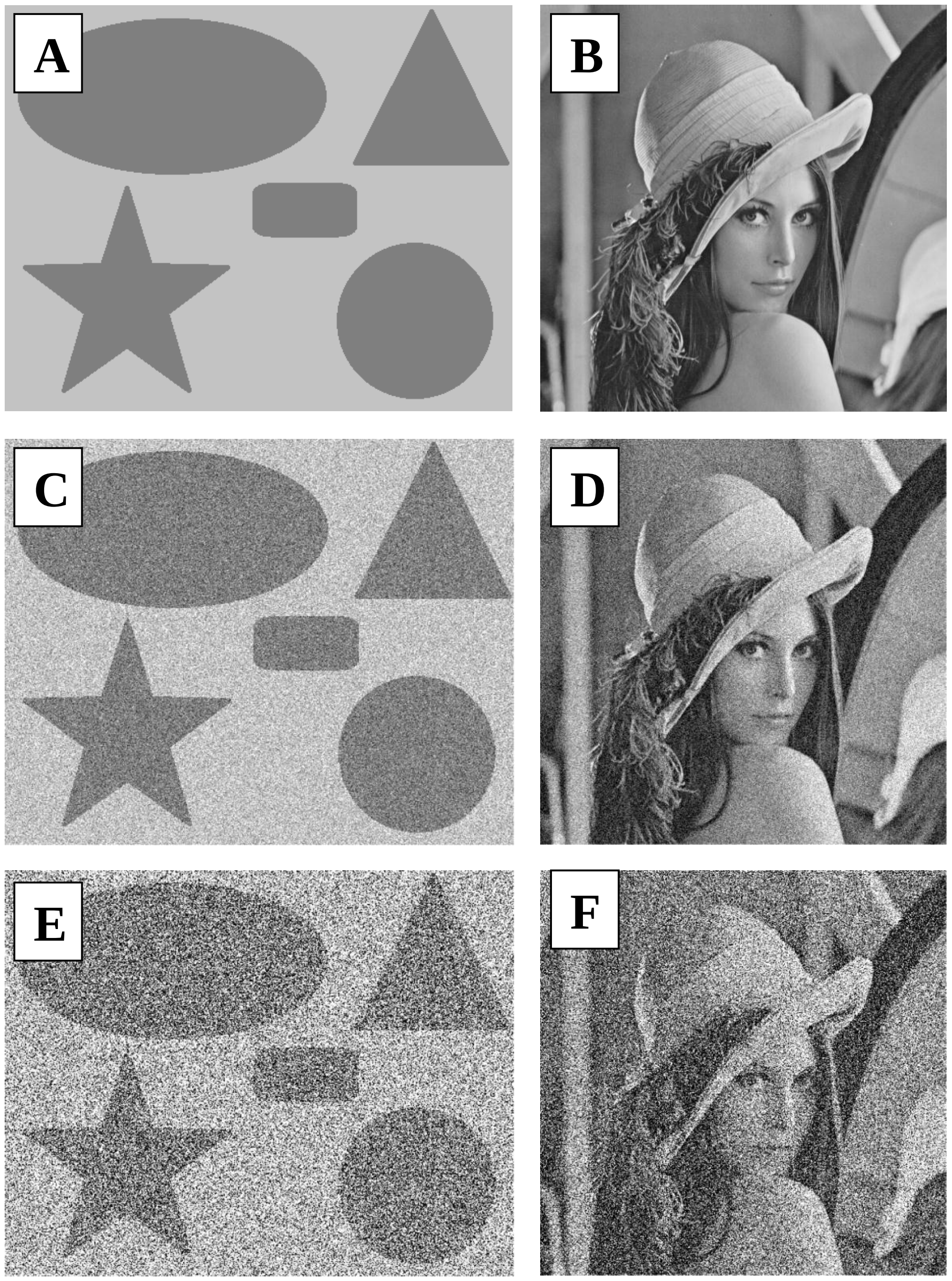


Figure 4-6. Gaussian white noise was added to two test images in order to evaluate the effectiveness of various denoising methods: (A) “Shapes,” a set of simple test shapes composed of grey values 127 and 195 created specifically for this work; (B) “Lena” standard test image; (C) Shapes with noise $\sigma = 30$, $PSNR = 18.7$; (D) Lena with noise $\sigma = 30$, $PSNR = 18.7$; (E) Shapes with noise $\sigma = 100$, $PSNR = 10.3$; (F) Lena with noise $\sigma = 100$, $PSNR = 10.1$.

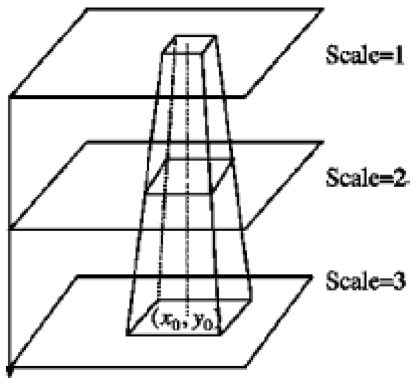


Figure 4-7. “Cone of Influence,” taken directly from [73].

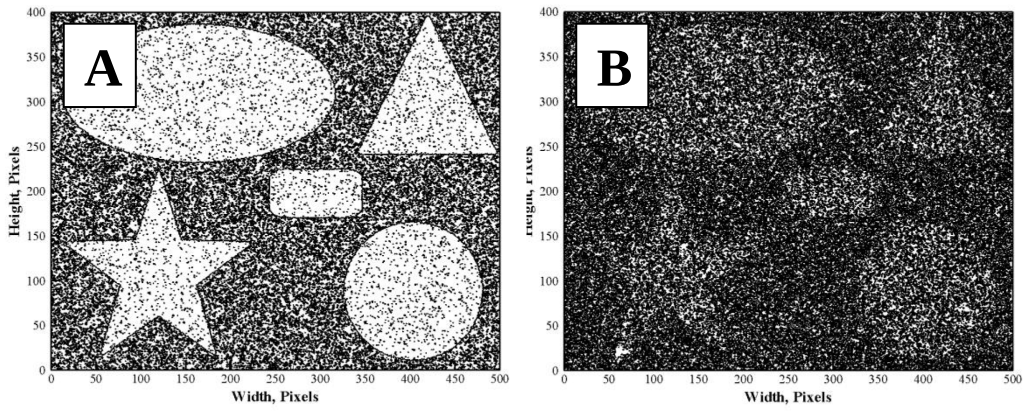


Figure 4-8. Segmenting the noisy Shapes image is not possible without pre-processing: Segmentation result for (A) $\sigma = 30$ and (B) $\sigma = 100$.

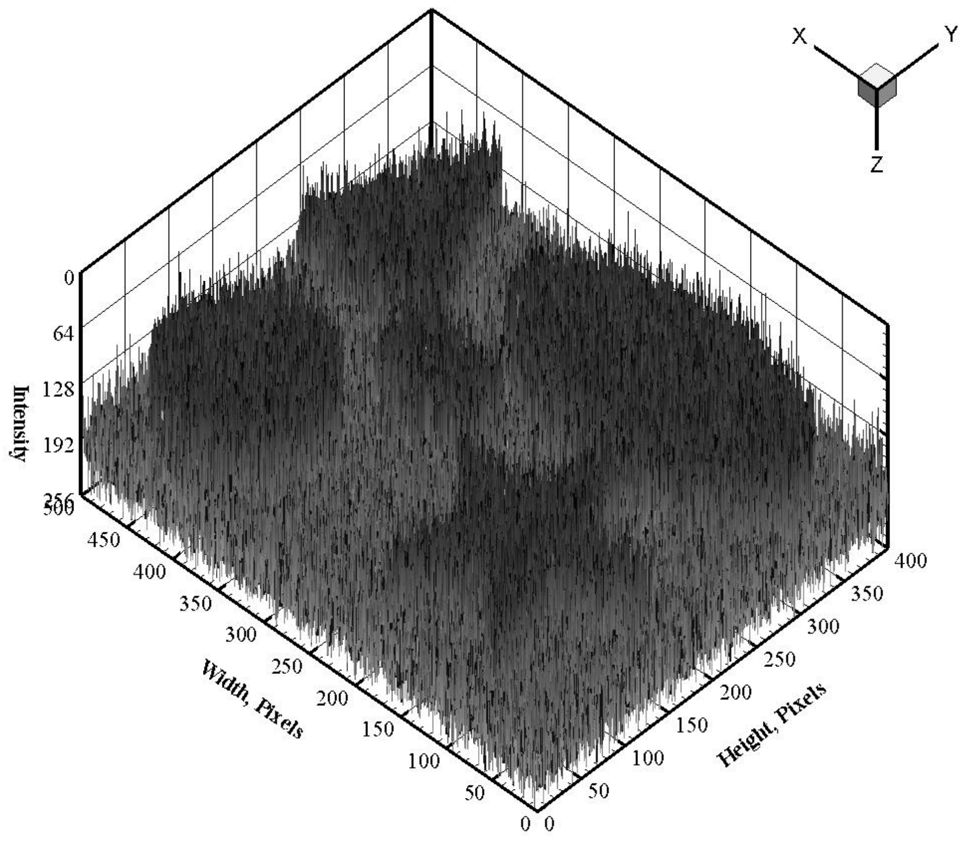


Figure 4-9. 3-D rendering of brightness intensity in the noisy Shapes test image clearly illustrates the difficulty that noise presents to level-based segmentation methods ($\sigma = 30$).

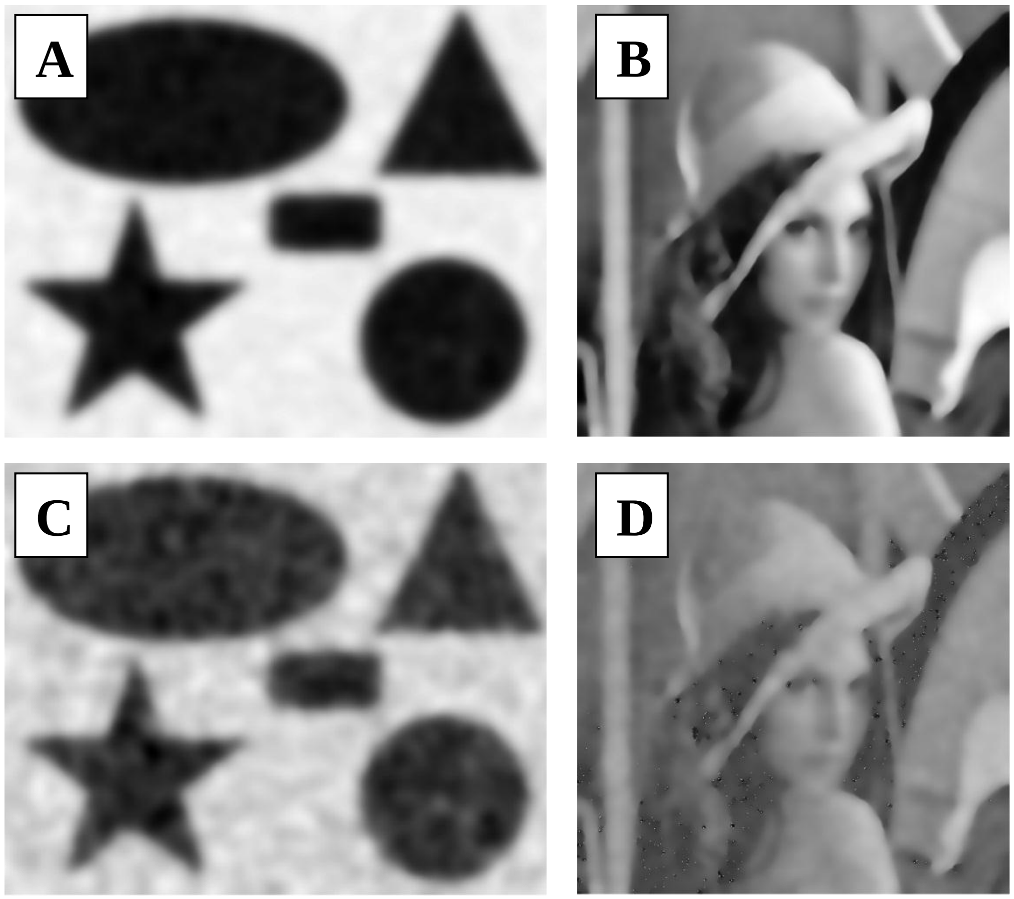


Figure 4-10. Test images smoothed using SRAD with $q_o = 1.0$, $iters_{max} = 500$: (A) $\sigma = 30$, $PSNR = 11.7$; (B) $\sigma = 30, PSNR = 20.8$; (C) $\sigma = 100$, $PSNR = 15.0$; (D) $\sigma = 100, PSNR = 20.9$.

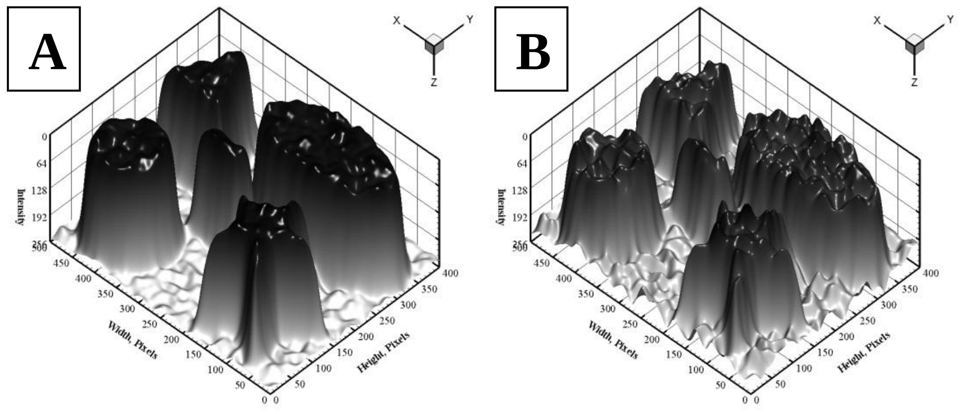


Figure 4-11. 3-D rendering of Test images smoothed using SRAD with $q_o = 1.0$, $iters_{max} = 500$: (A) $\sigma = 30$, $PSNR = 11.7$; (B) $\sigma = 100$, $PSNR = 15.0$.

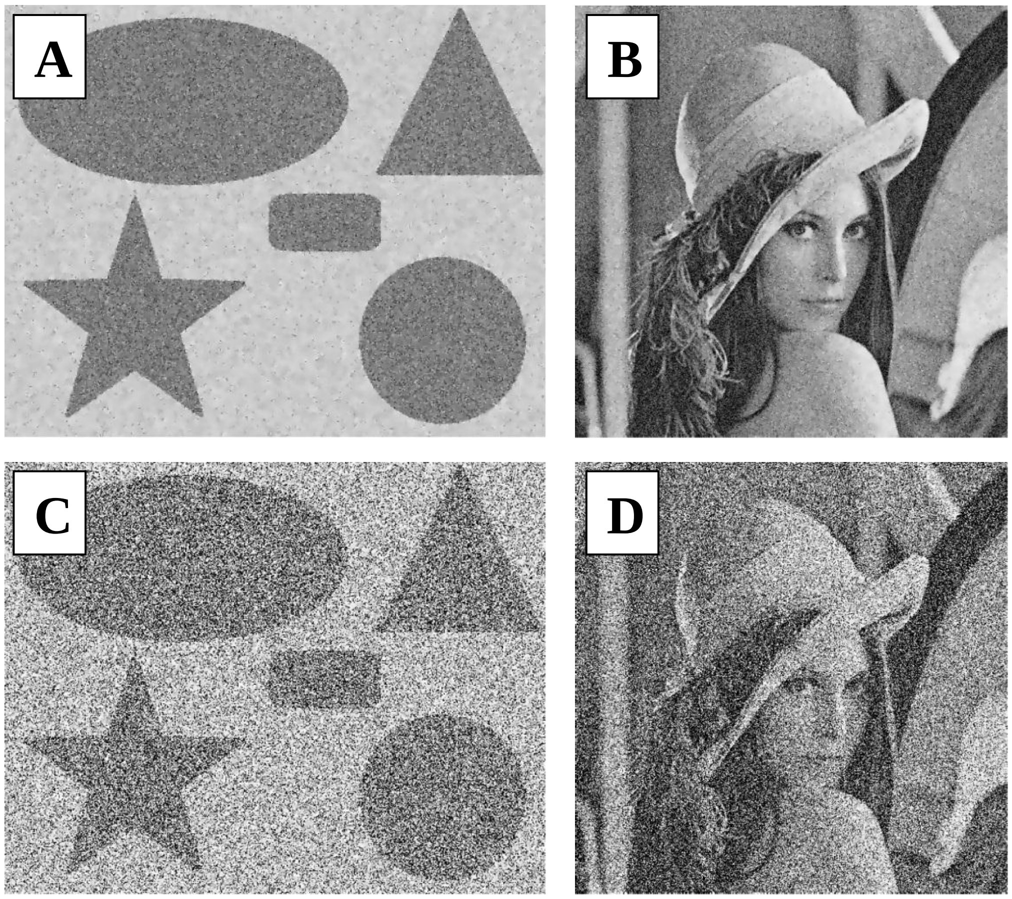


Figure 4-12. Test images smoothed using SRAD with $q_o = 0.25$, $iters_{max} = 500$: (A) $\sigma = 30$, $PSNR = 24.4$; (B) $\sigma = 30, PSNR = 21.4$; (C) $\sigma = 100$, $PSNR = 11.0$; (D) $\sigma = 100, PSNR = 10.6$.

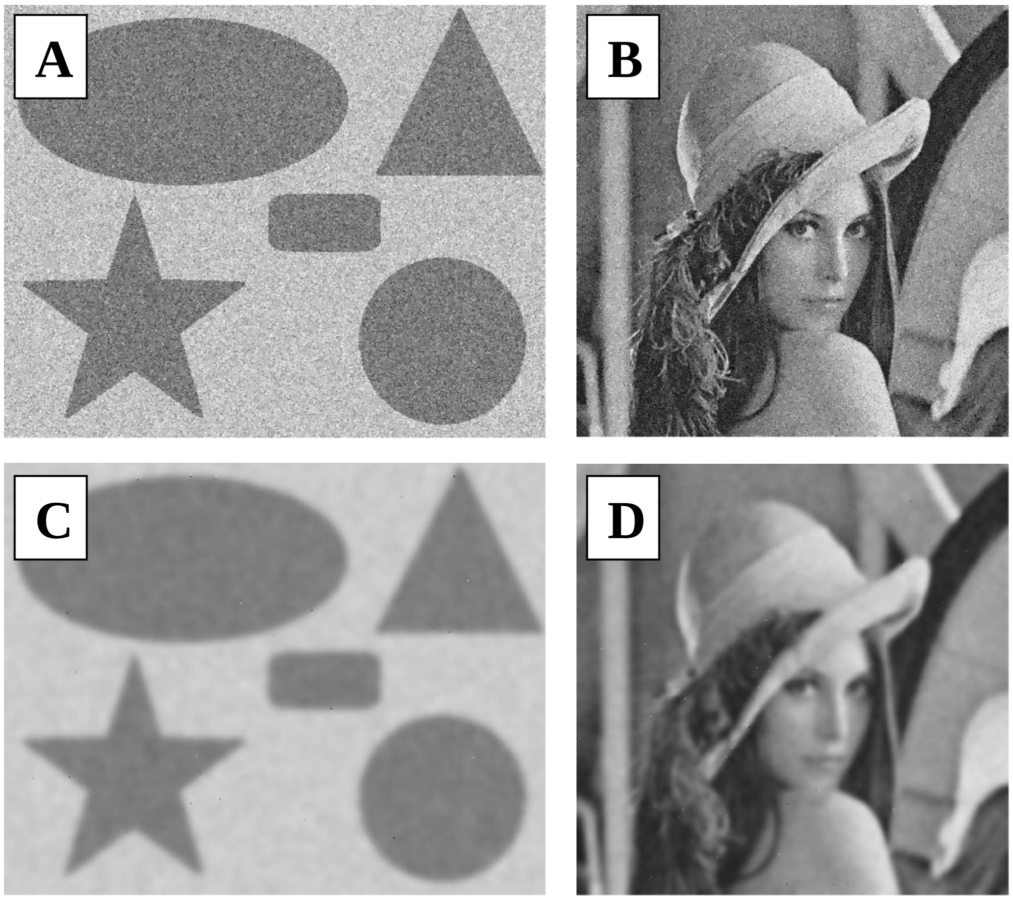


Figure 4-13. Test images smoothed using anisotropic diffusion; noise $\sigma = 30$ and $iters_{max} = 500$: (A) $\kappa = 30$, $PSNR = 19.6$; (B) $\kappa = 30$, $PSNR = 19.6$; (C) $\kappa = 100$, $PSNR = 28.9$; (D) $\kappa = 100$, $PSNR = 25.3$.

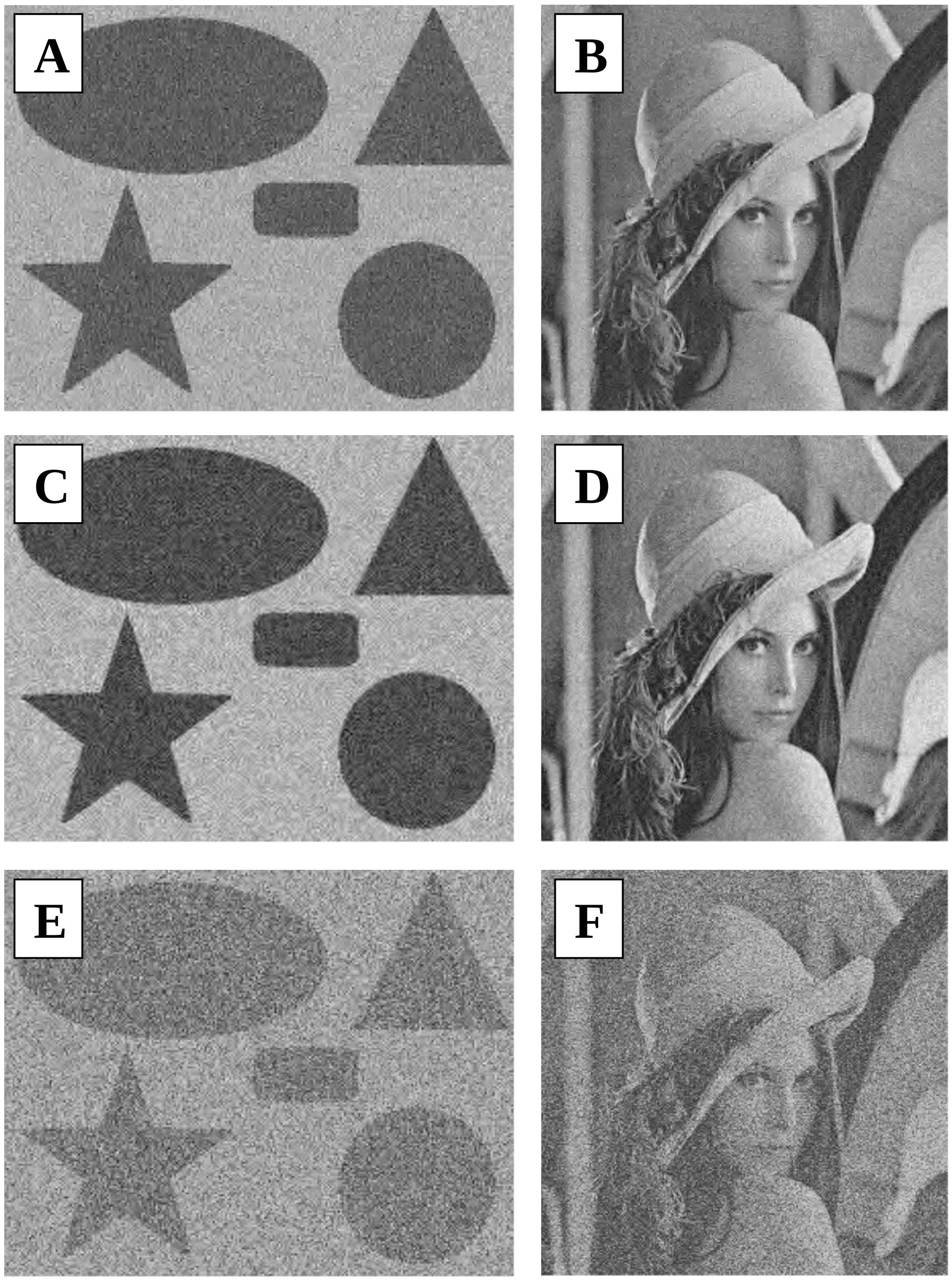


Figure 4-14. Test images smoothed using WMSAD: (A) Daub4, $\sigma = 30$, $PSNR = 18.7$; (B) Daub4, $\sigma = 30$, $PSNR = 25.7$; (C) Daub20, $\sigma = 30$, $PSNR = 17.5$; (D) Daub20, $\sigma = 30$, $PSNR = 26.3$; (E) Daub4, $\sigma = 100$, $PSNR = 16.7$; (F) Daub4, $\sigma = 100$, $PSNR = 17.4$.

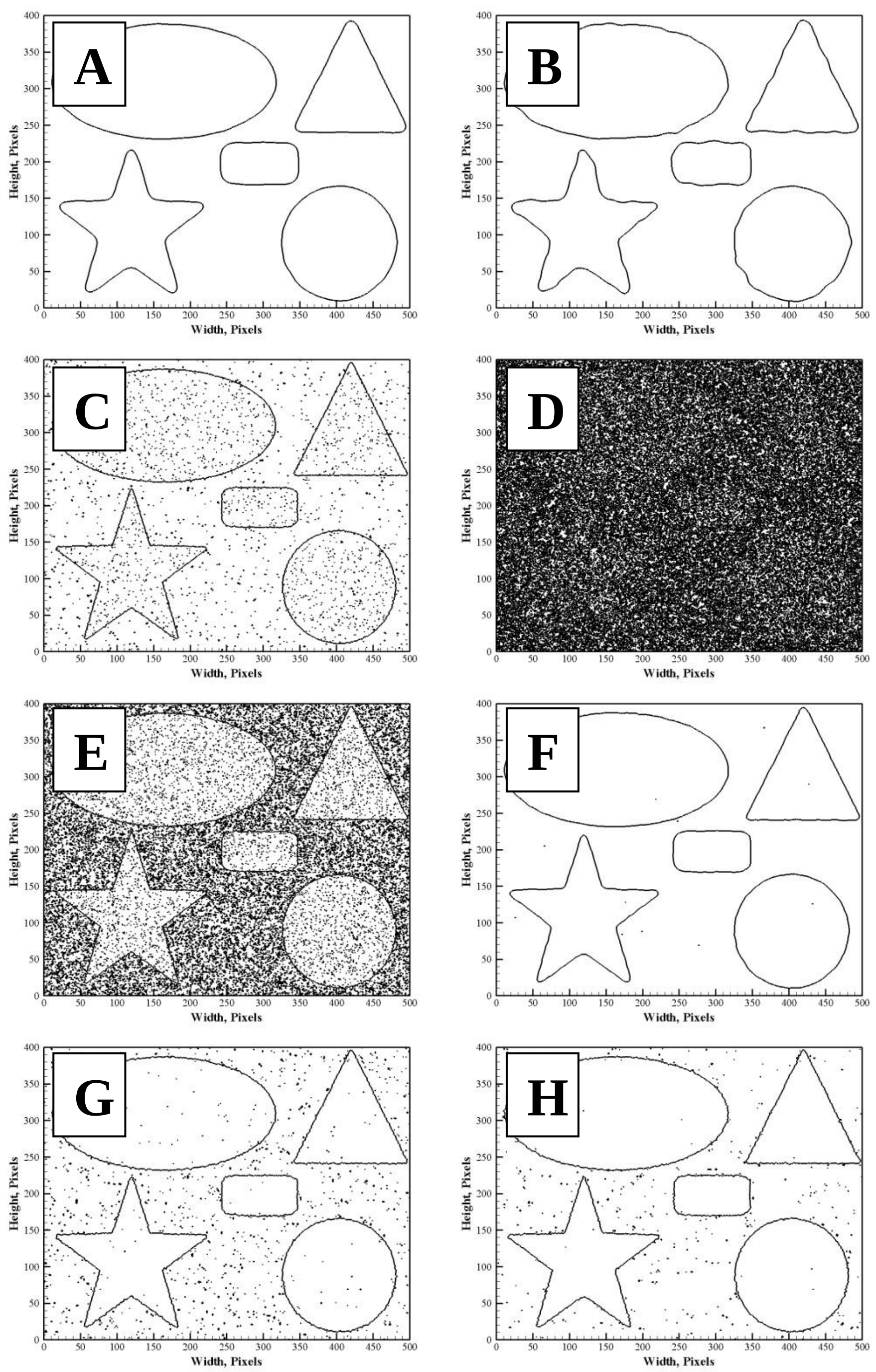


Figure 4-15. Segmentation of shapes test image after applying various denoising methods: (A) SRAD with $q_o = 1.0$, $\sigma = 30$, $PSNR = 11.7$; (B) SRAD with $q_o = 1.0$, $\sigma = 100$, $PSNR = 15.0$; (C) SRAD with $q_o = 0.25$, $\sigma = 30$, $PSNR = 24.4$; (D) SRAD with $q_o = 0.25$, $\sigma = 100$, $PSNR = 11.0$; (E) RAD with $\kappa = 30$, $\sigma = 100$, $PSNR = 11.0$; (F) RAD with $\kappa = 100$, $\sigma = 100$, $PSNR = 28.9$; (G) Daub4 WMSAD with $\sigma = 30$, $PSNR = 18.7$; (H) Daub20 WMSAD with $\sigma = 30$, $PSNR = 17.5$.

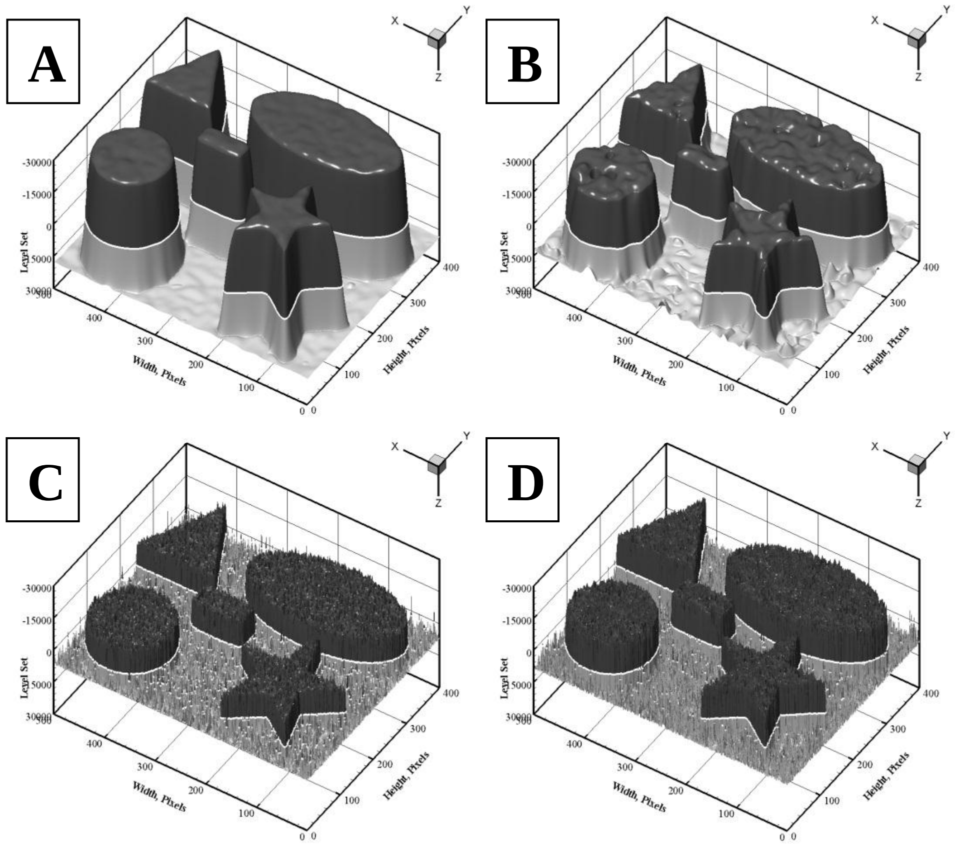


Figure 4-16. 3-D rendering of the level set field lends insight into segmentation effectiveness: $\sigma = 30$: (A) SRAD with $q_o = 1.0$, $\sigma = 30$; (B) SRAD with $q_o = 1.0$, $\sigma = 100$; (C) WMSAD Daub4, $\sigma = 30$; (D) WMSAD Daub20, $\sigma = 30$.

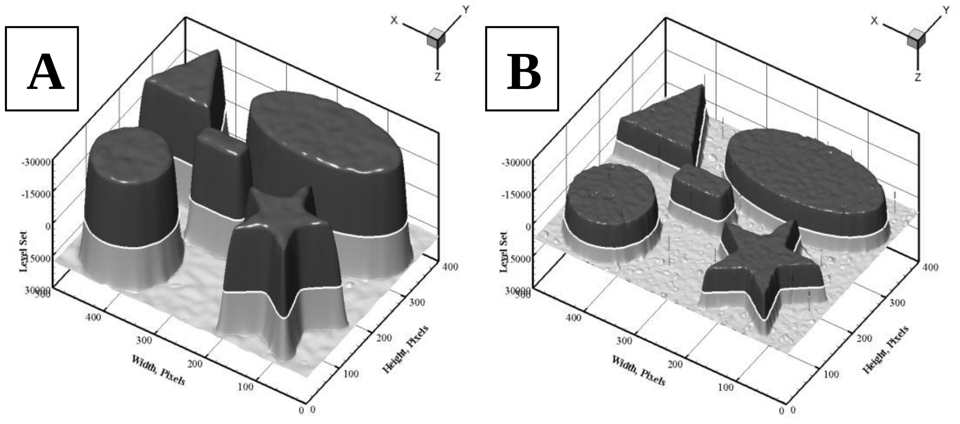


Figure 4-17. SRAD (A) improves upon the original anisotropic diffusion method (B) by increasing contrast and eliminating outlying noise “speckles.” (Shapes test image, $\sigma = 30$.)

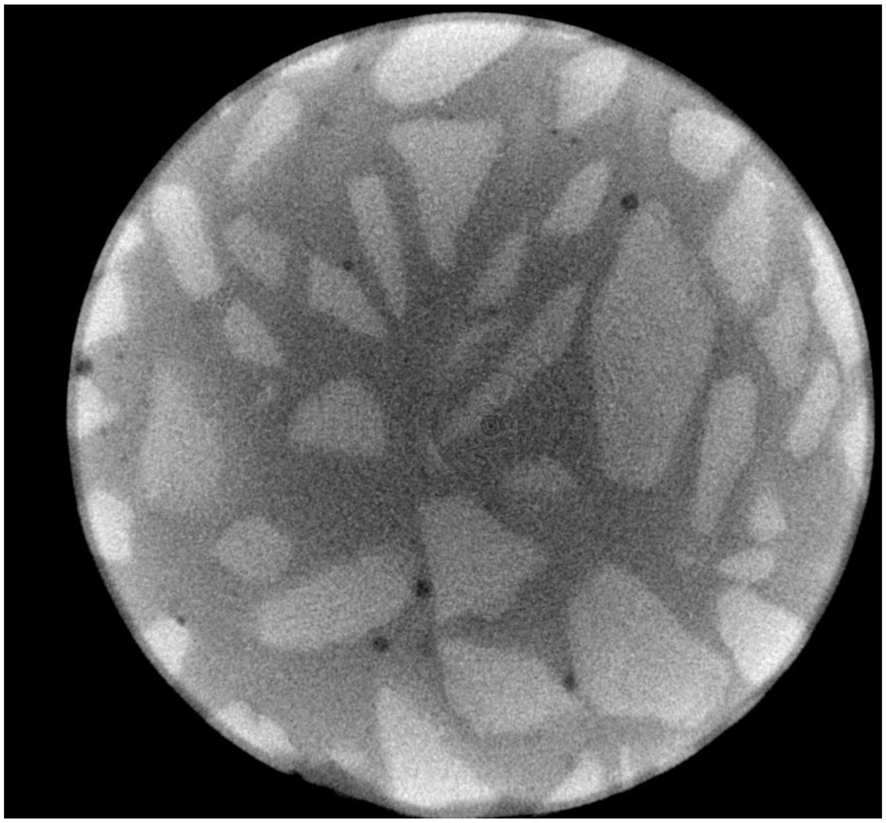


Figure 4-18. CT slice of a concrete sample. Original image dimensions 897 × 832 in 679 slices.

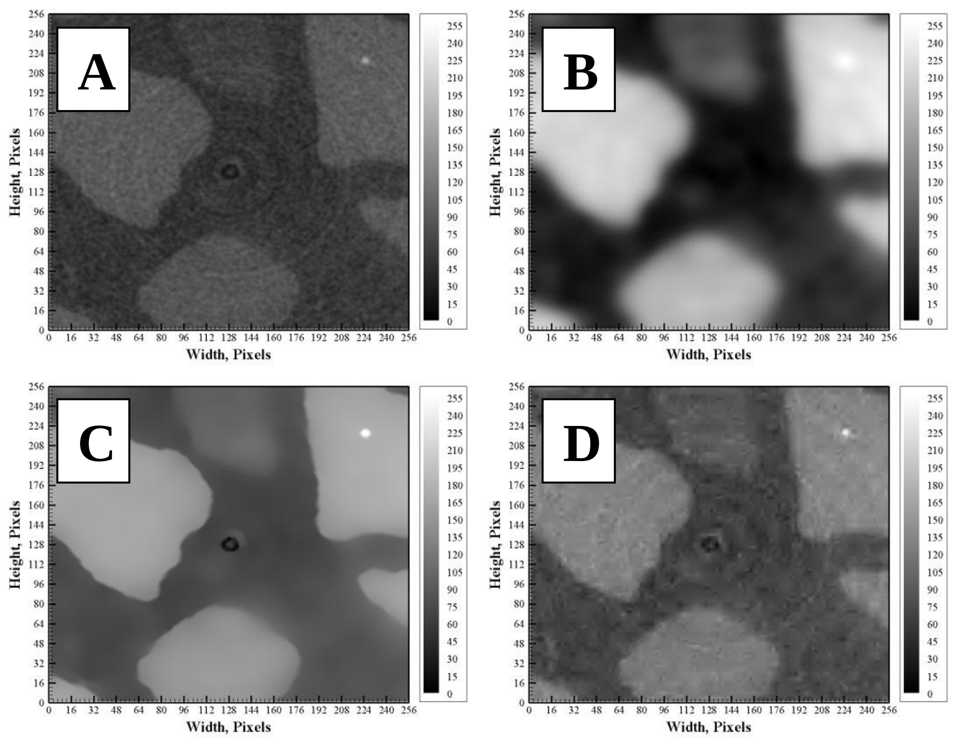


Figure 4-19. Section of concrete image slice, cropped for processing: (A) Original image; (B) SRAD with $q_o = 1.0$; (C) SRAD with $q_o = 0.25$; (D) WMSAD Daub4.

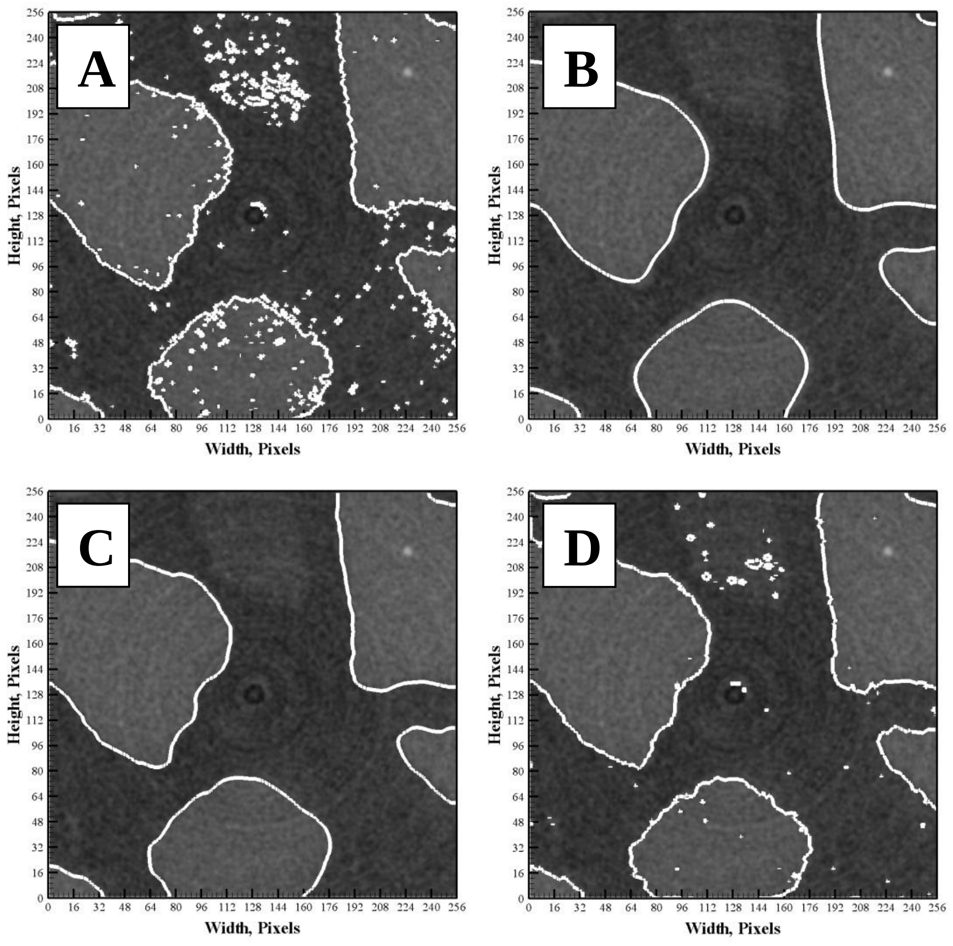


Figure 4-20. Concrete segmentation results, overlaid on the original image slice for visual clarity: (A) Original image; (B) SRAD with $q_o = 1.0$; (C) SRAD with $q_o = 0.25$; (D) WMSAD Daub4.

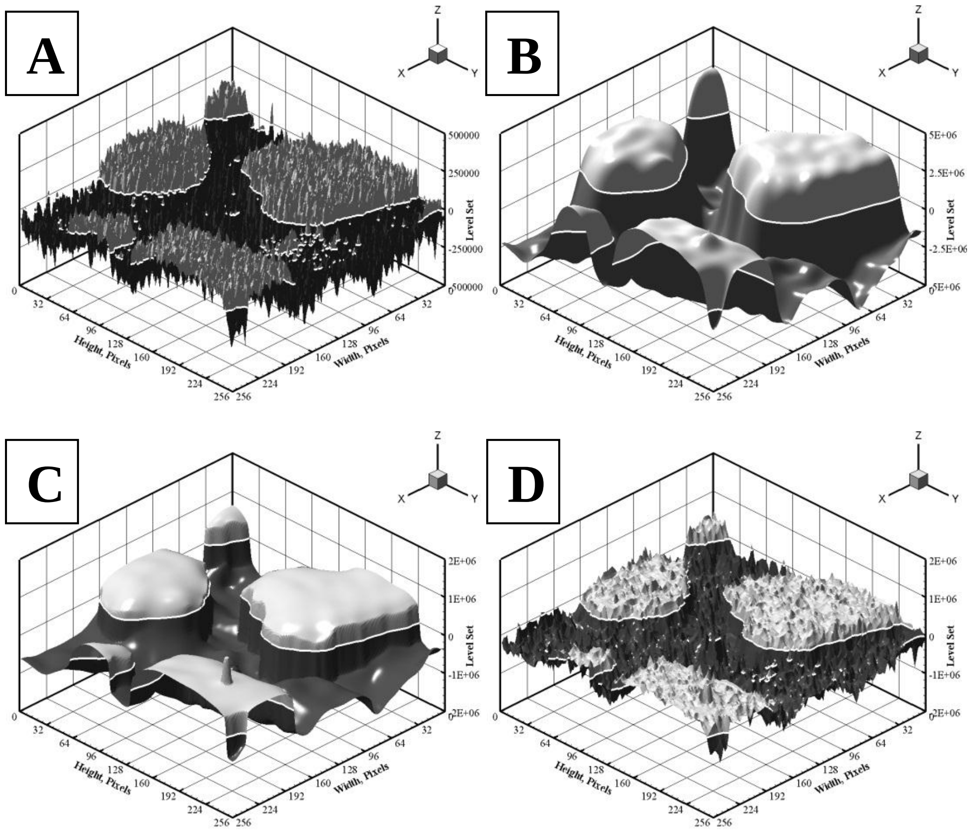


Figure 4-21. 3-D rendering of concrete image slice sections: (A) Original image; (B) SRAD with $q_o = 1.0$; (C) SRAD with $q_o = 0.25$; (D) WMSAD Daub4.

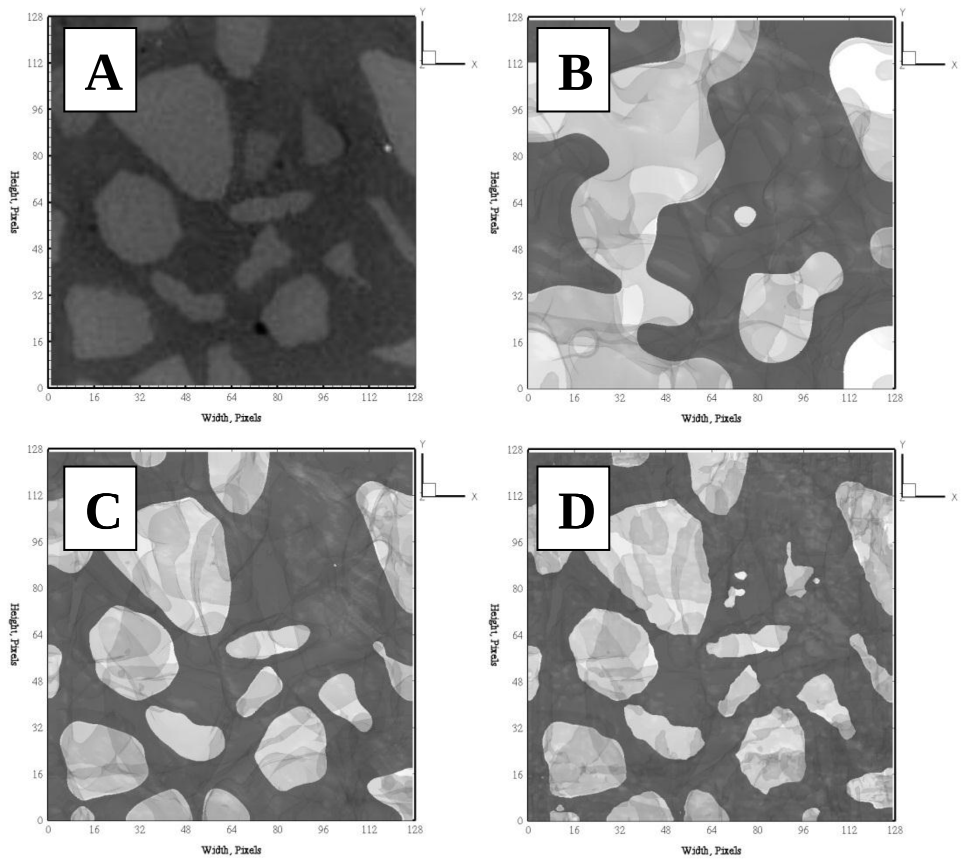


Figure 4-22. Single slice (45 of 128) of the 3-D concrete data set: (A) original image; (B) segmented surface after denoising with 3DSRAD, $q_o = 1.0$; (C) segmented surface after denoising with 3DSRAD, $q_o = 0.25$; (D) segmented surface after denoising with WMSAD, Daub4 wavelets.

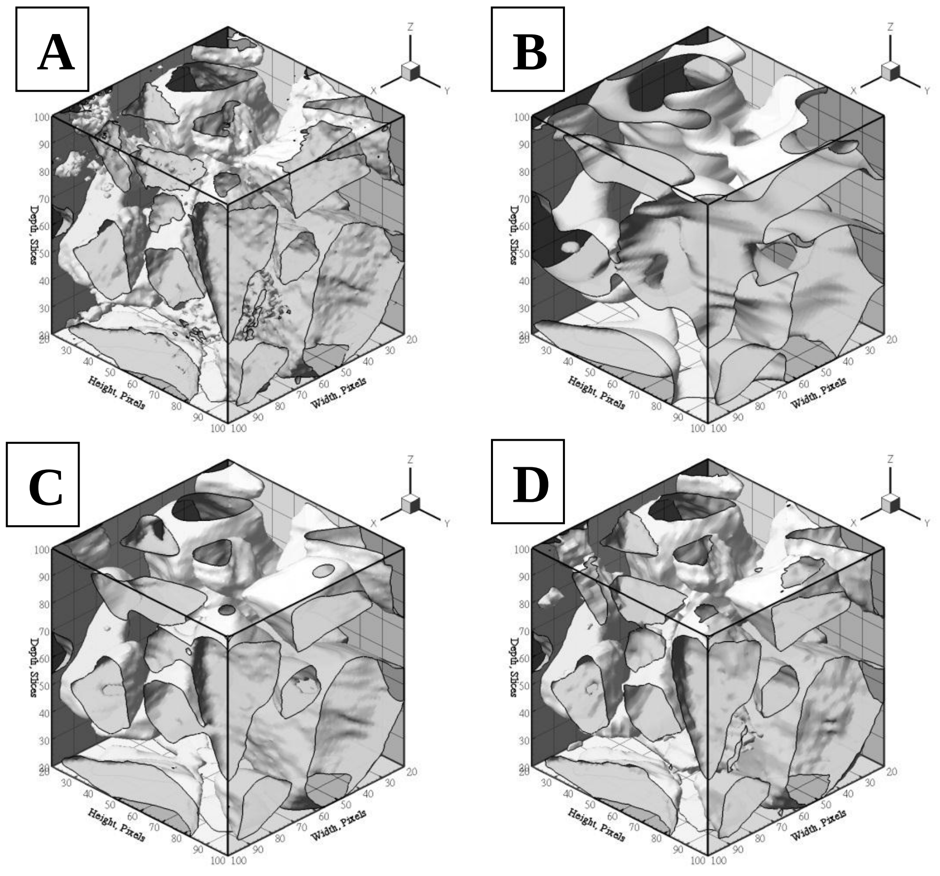


Figure 4-23. 3-D concrete data set: (A) original image, isocontours of grey level 70; (B) segmented surface after denoising with 3DSRAD, $q_o = 1.0$; (C) segmented surface after denoising with 3DSRAD, $q_o = 0.25$; (D) segmented surface after denoising with WMSAD, Daub4 wavelets.

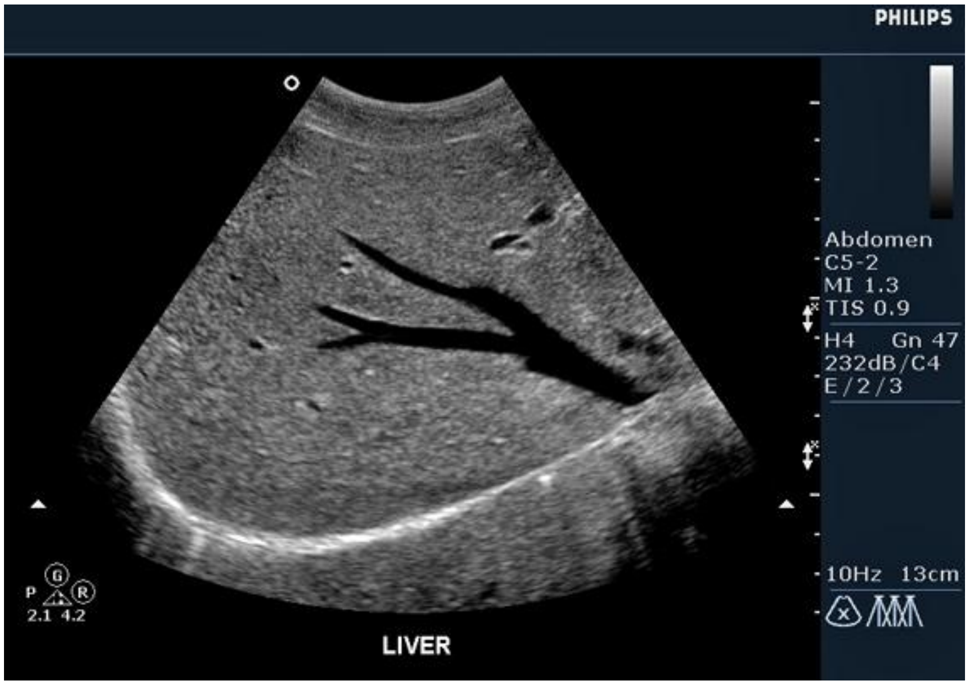


Figure 4-24. Ultrasound image of the liver, from the Philips ultrasound web site at http://www3.medical.philips.com/en-us/secure/images_site/largeImage.asp?size=blowup&classcode=03&appcode=a&imagename=0084-HD11-C5-2-ABD&systemcode=c&div=ultra.

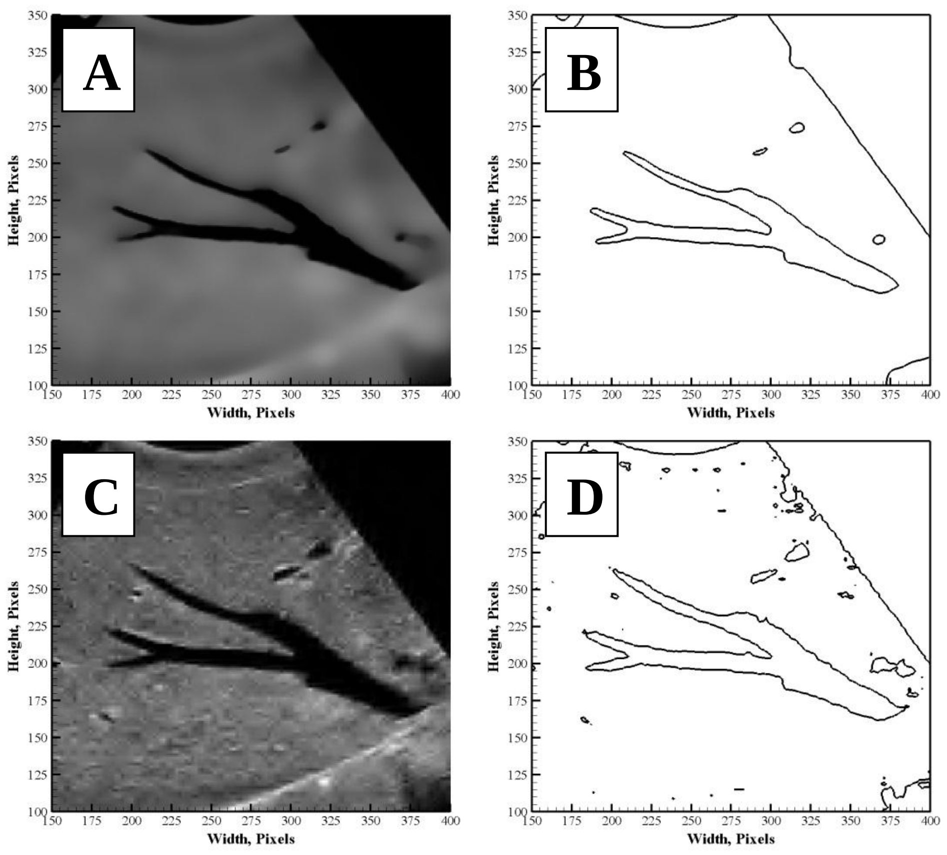


Figure 4-25. Denoised image and segmentation results for a cropped section of the Philips ultrasound liver image: (A, B) SRAD, $q_o = 1.0$; (C, D) WMSAD, Daub4 wavelets.

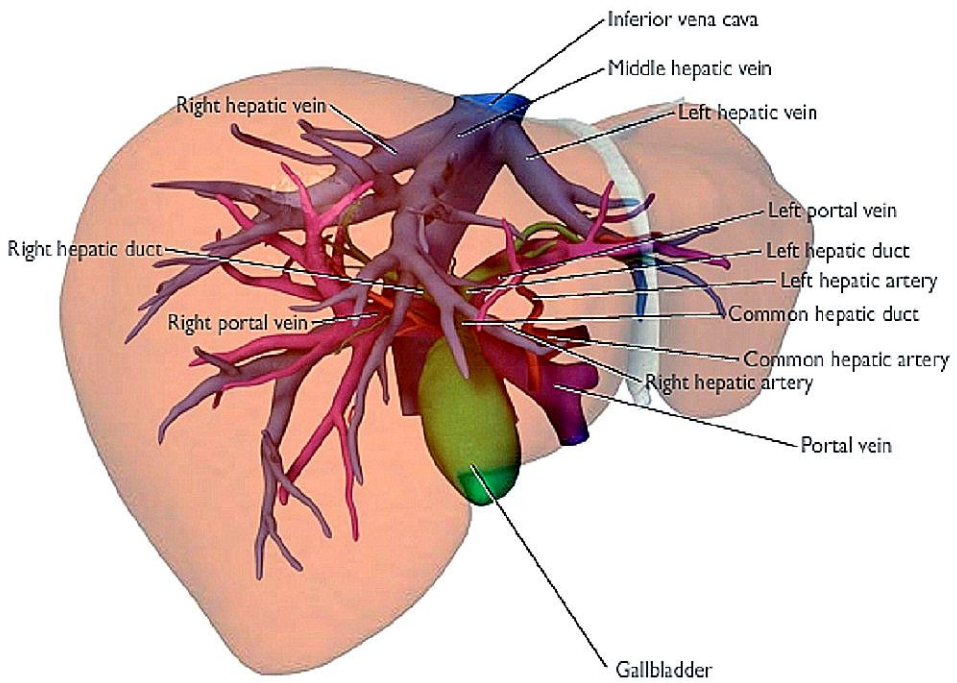


Figure 4-26. Screenshot of liver anatomy taken from http://pie.med.utoronto.ca/VLiver/VLiver_content/VLiver_interactiveLiver.html.

CHAPTER 5

TOWARD TRULY EULERIAN IMAGE BASED MODELING

5.1 Introduction

A recurring theme in this work has been the difficulty involved with meshing the surfaces of complex moving boundaries in a Lagrangian sense. Meanwhile, the whole point of the image-based modeling framework being developed in this thesis is to greatly simplify the process of creating complex moving geometries in fluid flow calculations. While the algorithm laid out in Chapter 3 was shown to succeed in enabling accurate representation of complex motions that would be difficult or nearly impossible to define functionally, particularly in the case of modeling duodenal peristaltic contractions, it still falls short of attaining the desired level of simplicity.

The method outlined in Chapter 3 employed a modified rapid k-means method to create a binary segmentation field for object tracking, which was stepwise in nature. Because pixels could only take on one of two possible values with this approach, the interface between an object and its surroundings was necessarily stepwise as well, disallowing for smooth segmentation contours. Most importantly, there existed no mechanism for calculating the velocity of an object with this binary representation. Thus, object interfaces had to be populated with Lagrangian points in order to provide a complete surface description so that boundary conditions could be assigned to imaged objects. Points were necessary because they provided a set of physical addresses that could be used for interface smoothing and, combined with knowledge of the elapsed time between image frames, allowed for the construction of interface velocities. Of course, the one-to-one point correspondence required for velocity calculations was in general not given directly when populating the binary segmentation field; an identical number of interfacial pixels were not usually found upon segmentation from one frame to the next, and so further manipulation had to proceed in order to ensure that there were an equal number of surface points between image frames being considered, and that the points were equally spaced to avoid unphysical jumps in velocity or lack of grid connectivity with the underlying modeling mesh.

The tedium involved with these point population and smoothing operations, combined with a lack of generality in surface detection, impelled the development of a more robust and reliable method that does not rely on surface point tracking. The flow solver and level set methods used in our CFD code operate in an Eulerian sense on a fixed Cartesian mesh, and so it seemed most natural to segment images for the purpose of obtaining level set boundaries in the same way. In essence, we desired to develop a method which can directly generate Eulerian level set fields on a series of images so that the resultant level sets may be employed in supplying boundary conditions straight away, without the need for any intermediate Lagrangian steps.

The denoising and improved segmentation developments outlined in Chapter 4 were important first steps away from the Lagrangian method, as they gave us the ability to generate smooth contours of imaged objects without the use of points. However, this leaves us without the information that was afforded by those points – the ability to predict intermediate positions of interfaces between image frame (fiducial) time steps so that the imaged object may be allowed to evolve with the temporal resolution demanded by the flow solver, and the ability to set boundary conditions using one-to-one correspondence and knowledge of point displacements through a series of passing image frames.

One promising way of supplying the information previously supplied through the use of Lagrangian point tracking lies with a paradigm known as *optical flow*. Optical flow was first invented for the purpose of tracking objects in the context of computer vision, and as such offers a mechanism for generating velocity vector fields that describe motions in an imaged scene. It is essentially an Eulerian representation of movement as it is perceived visually, lending a global perspective that fits well with present goals aimed toward simplicity and seamless integration with the established Cartesian flow solver code. Recent developments in optical flow and its use in conjunction with active contour segmentation techniques make it attractive as a potential means for accomplishing the goals of this thesis project, namely, enabling a completely Eulerian representation of complex moving geometries and their interaction with surrounding fluid media without the need for any information about modeled objects outside of how they appear visually. A literature survey describing the evolution of the field of optical flow is given as background in the following sections.

5.2 The Origins of Optical Flow

Optical flow has been an active area of research in the field of computer vision since it was first introduced by Berthold Horn and Brian Schunck in 1981. In their seminal publication *Determining Optical Flow*, Horn and Schunck define optical flow as a distribution of apparent velocities of the movements of brightness patterns in an image, and put forth a novel means for computing it. The term "brightness pattern" is synonymous with the distribution of irradiance intensity in an image, and is defined as $I(\boldsymbol{x}, t)$, where $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ in a 2-D image sequence [15].

The purpose of optical flow is to provide information about the rate of change of the spatial arrangement of objects that appear in a series of image frames. As an extension to already existing edge detection methods, Horn and Schunck made the portentous realization that discontinuities in an optical flow field could be useful in marking segmentation boundaries, and thus could help identify the locations and the movements of objects in a scene simultaneously [15]. Because optical flow is a paradigm based upon visual perception, they made several assumptions that match what we already intuitively know about our own vision:

i) The apparent velocity of brightness patterns can be directly identified with surface motions of objects in an image scene, however, an apparent velocity exists only if it can be detected visually. As a counterexample, a perfect uniform cylinder rotating about its axis viewed from a fixed location with a fixed light source will not have any apparent velocity, because there is nothing visually indicating that it is in fact a moving object. Thus, optical flow is limited to detecting changes in scale and position of regions of pixels relative to each other.

ii) The surface being imaged is considered to be a flat object, i.e. variations in brightness caused by shading effects are ignored. This simply means that 3-D objects become 2-D projections when imaged, which is again visually intuitive.

iii) Incident illumination is uniform across a surface, such that brightness is proportional solely to the reflectance of the surface at the corresponding point on an object:

$$I(\boldsymbol{x}, t) \propto \rho(\boldsymbol{x}, t). \tag{5.1}$$

This is really just an extension of assumption (ii), stating that the light source remains fixed in the 2-D projection being considered.

iv) Reflectance varies smoothly and has no spatial discontinuities. Thus, image brightness is differentiable.

These assumptions (i) through (iv) lead to the first of two modeling constraints imposed by Horn and Schunck in generating an optical flow field: the brightness of any material object in the image is constant, so that

$$\frac{dI}{dt} = 0. \tag{5.2}$$

From an Eulerian perspective, this is written as

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0. \tag{5.3}$$

If we let $\frac{dx}{dt} = u$ and $\frac{dy}{dt} = v$ be components of velocity, then $\boldsymbol{u} = \begin{bmatrix} u \\ v \end{bmatrix}$ and Equation 5.3 can be re-written as

$$\frac{\partial I}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} I = 0. \tag{5.4}$$

Alternatively, Equation 5.3 can be written as

$$I_x u + I_y v + I_t = 0, \tag{5.5}$$

where any subscript $(*)$ denotes $\frac{\partial}{\partial *}$ [15]. This will be the notation used for the duration of the chapter.

Because intensity is a scalar value measured locally in each pixel comprising an image, another constraint is needed in order to provide an optical flow velocity, which of course has two components in a 2-D image sequence. Horn and Schunck note that Equations 5.3 through 5.5 provide the component of motion in the direction of the image intensity gradient (which is $-\frac{I_t}{\sqrt{I_x^2+I_y^2}}$ by inspection), but the component of optical flow velocity tangent to the intensity gradient cannot be found without a second constraint [15]. To provide the missing information, then, they make the additional assumption that objects in an image will always undergo continuous deformation or rigid body motion, and therefore that adjacent object regions must have similar velocities; the image intensity velocity field must vary smoothly everywhere except at object boundaries, where there can be discontinuities. This introduces a smoothness constraint, the second constraint needed to construct a 2-component velocity field, which is proposed to be accomplished by minimizing the magnitude of the gradient of the optical flow field velocity $\sqrt{\boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} u} = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2}$ [15].

The overall aim is to approximately satisfy the constraints by minimizing simultaneously [15]:

a) the sum of the errors in image brightness

$$\mathcal{E}_b = I_x u + I_y v + I_t \tag{5.6}$$

b) the departure from flow field smoothness

$$\mathcal{E}_s^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{5.7}$$

Minimizing these two error equations together is necessary and sufficient to obtain a complete flow field [15].

A weighting factor α^2 is defined for the purpose of deciding the relative importance of errors in velocity values and field smoothness. Since it is desired to determine the optical flow field over the entire image domain, the problem is posed as an integral functional, which represents a total error, or energy, to be minimized:

$$E^2 = \iint \mathcal{F} dxdy, \tag{5.8}$$

where

$$\mathcal{F} = \alpha^2 \varepsilon_s^2 + \varepsilon_b^2. \tag{5.9}$$

Thus, the goal here is to find a mapping $\mathcal{F}: \mathbb{R}^2 \to \mathbb{R}$ that minimizes the energy E^2 characterizing image space $\Omega \subset \mathbb{R}^2$.

This type of minimization problem belongs to the Calculus of Variations, and the mapping $\mathcal{F}$ is given by the solution to a pair of Euler-Lagrange equations [15, 75-76] (Appendix A):

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \nabla^2 u = 0 \tag{5.10}$$

$$I_y(I_x u + I_y v + I_t) - \alpha^2 \nabla^2 v = 0 \tag{5.11}$$

Horn and Schunck used 3-D forward differencing throughout their numerical formulation to estimate spatial and temporal derivatives (Appendix B), arguing that they have found formulae with larger support to give equivalent results when applied to smooth images. Of course, if the image is not smooth, it may be useful to apply higher-order discretizations, possibly even central differencing, so that noisy pixels and other spurious discontinuities may not be so amplified in their effect. Laplacians of flow velocities were estimated as (Figure 5-1)

$$\nabla^2 u \approx \bar{u}_{i,j,k} - u_{i,j,k} \tag{5.12}$$

$$\nabla^2 v \approx \bar{v}_{i,j,k} - v_{i,j,k}, \tag{5.13}$$

where $\bar{u}$ and $\bar{v}$ are local average velocity components, defined in Appendix B.

Introducing the Horn and Schunck approximation for the Laplacians in Equations 5.12 and 5.13 gives the equations

$$I_x\left(I_x u + I_y v + I_t\right) - \alpha^2(u - \bar{u}) = 0 \quad \textbf{(5.14)}$$

$$I_y\left(I_x u + I_y v + I_t\right) - \alpha^2(v - \bar{v}) = 0, \quad \textbf{(5.15)}$$

which, cast in terms of the optical flow field components, become (Appendix C)

$$u = \bar{u} - I_x\left(I_x\bar{u} + I_y\bar{v} + I_t\right)/\left(\alpha^2 + I_x^2 + I_y^2\right) \quad \textbf{(5.16)}$$

$$v = \bar{v} - I_y\left(I_x\bar{u} + I_y\bar{v} + I_t\right)/\left(\alpha^2 + I_x^2 + I_y^2\right). \quad \textbf{(5.17)}$$

In this first work, the equations for *u* and *v* are iteratively solved explicitly:

$$u^{n+1} = \bar{u}^n - I_x\left(I_x\bar{u}^n + I_y\bar{v}^n + I_t\right)/\left(\alpha^2 + I_x^2 + I_y^2\right) \quad \textbf{(5.18)}$$

$$v^{n+1} = \bar{v}^n - I_y\left(I_x\bar{u}^n + I_y\bar{v}^n + I_t\right)/\left(\alpha^2 + I_x^2 + I_y^2\right) \quad \textbf{(5.19)}$$

The total number of iterations n_{iter} required for convergence of Equations 5.18 and 5.19 is determined by the size of the image domain [15]. In regions of the image where the gradient $\boldsymbol{\nabla} I$ is small, such as a group of pixels representing the body of an object in the domain, local optical flow velocity estimates will simply be the average of the neighboring velocity estimates; Horn and Schunck note that velocity information will propagate inwards from object boundaries, in a manner analogous to the evolution of the heat rate equation where change in temperature is proportional to the Laplacian. Thus, it is recommended to set $n_{iter,max}$ to the maximum side length of the image pixel grid as a conservative estimate that allows information to propagate fully through the image field [15].

$$n_{iter,max} = \mathrm{MAX}\left(n_{x,image}, n_{y,image}\right) \quad \textbf{(5.20)}$$

One of the benefits of a variational formulation is that natural boundary conditions are always intrinsically supplied by the imposed variational displacement [76]. Horn and Schunck point out that the natural boundary conditions for their formulation turn out to be homogeneous Neumann conditions (zero normal derivatives), and so boundary pixels may be treated simply by copying their values to a layer of pixels padding the original image domain.

In [15], no guidance is given with regard to setting the relative weighting α^2 between brightness constancy and smoothness constraints. However, a 2008 paper by Jhunjhunwala and Rajagopalan titled *Optical Flow based Volumetric Spatio-Temporal Interpolation* offers as its primary contribution a means of estimating a value for α that (it is hoped) will prevent the problem from becoming over- or under-constrained. Empirically, Jhunjhunwala and Rajagopalan found that the best results were achieved on their test images by setting α equal to the gradient magnitude calculated at a given pixel, for every pixel in the domain [16].

$$\alpha_i = \|\nabla I_i\| \tag{5.21}$$

Overall, the variational formulation proposed in this first optical flow publication by Horn and Schunck was found by them to work nicely for smooth sample images, but suffered large errors near discontinuities in some of the test images analyzed. In terms of this thesis project, that is a rather unfortunate consequence because the primary goal in making use of the optical flow method is to effectively model moving boundaries with accurate velocity conditions in a sharp fashion using level sets. As it turns out, however, the field of optical flow has progressed considerably since 1981, in a direction favorable to present developments.

5.3 Combining Optical Flow with Active Contours

In a 2004 article titled *Active Contours and Optical Flow for Automatic Tracking of Flying Vehicles*, the authors (Ha et al.) employ the optical flow formulation of Horn and Schunck as a means for determining where to initialize active segmentation contours so that target object interfaces may be found and revealed more quickly and reliably [14]. The optical flow velocity field of an image sequence is used for contour initialization on each image frame supplied, and then the contour is evolved from this predicted position until a suitable segmentation is achieved.

Active contours are defined in [14] as deformable continuity splines which are generally evolved by energy minimization methods under the influence of image-dependent forces, internal forces, and user-defined constraints. Ha et al. propose a level set active contour evolution approach based on

i) Euclidean curve shortening: A gradient direction is defined in which a given curve shrinks as rapidly as possible relative to Euclidean arc length.

ii) Theory of conformal metrics: Euclidian arc length is multiplied by a conformal factor that is defined by features to be extracted, i.e. regions of high gradient, then the corresponding gradient evolution equations are computed.

Casting the image domain in this framework generates a potential well at the bottom of which lie features of interest to be extracted. The initial contour then simply flows into the well as it evolves, and segmentation is complete [14].

They start with a level set formulation of a Euclidian curve-shortening equation

$$\frac{\partial\varphi}{\partial t} = \psi(x,y)\|\nabla\varphi\|\left[\nabla\cdot\frac{\nabla\varphi}{\|\nabla\varphi\|}+\nu\right], \qquad \textbf{(5.22)}$$

where φ is a level set field, $\frac{\nabla\varphi}{\|\nabla\varphi\|}$ is the outward unit normal $\boldsymbol{n}$ to a given level set contour, and the divergence of the normal $\boldsymbol{\nabla}\cdot\boldsymbol{n}$ defines contour curvature κ. The term v is a so-called "inflation" term defined to keep the level set contours flowing in the proper direction [14].

$\psi(x, y)$ is a "stopping term," included to prevent curve evolution from continuing forth in regions where an edge is detected, hopefully resulting in the contour ceasing motion once the boundaries of an object are segmented. Thus, it is desirable for ψ to be small near edges. A common definition for ψ in image processing is

$$\psi(x,y) = (1 + \|\nabla G_\sigma(x,y) * I(x,y)\|^2)^{-1}, \qquad \textbf{(5.23)}$$

where G_σ is a 2D Gaussian smoothing filter, based on some tunable parameter σ, that is convoluted with pixel intensity (Equation 5.23) [14].

$$G_\sigma(x,y) = \sigma^{-1/2} e^{-|x^2+y^2|/4\sigma} \qquad \textbf{(5.24)}$$

For a typical curve $C = [x(p), y(p)]^{\mathrm{T}}$ parameterized by some parameter p, arc length is given by

$$ds = \sqrt{x_p^2 + y_p^2}\,dp, \qquad \textbf{(5.25)}$$

where subscripts denote derivatives in p in the usual convention defined earlier in this chapter. Ha et al. define a new arc length function dependent upon $\psi(x, y)$, so that the curve shortening portion of the level set formulation $\|\boldsymbol{\nabla}\varphi\|\kappa$ will be dependent upon a new metric ds_ψ (Equation 5.26) which becomes small near edges.

$$ds_\psi = \sqrt{x_p^2 + y_p^2}\,\psi dp \qquad \textbf{(5.26)}$$

In other words, the initialized level set curve is designed to shrink most quickly where it is desired to, in uniform regions of the image space, while evolving slowly or completely ceasing to shrink where there exists an edge to be segmented.

At this point, the authors introduce an artificial time parameter *t* to evolve curve $C = [x(p,t), y(p,t)]^{\mathrm{T}}$ according to a length function

$$L_\psi(t) = \int_0^1 \left\| \frac{\partial C}{\partial p} \right\| \psi \, dp. \tag{5.27}$$

The first variation of Equation 5.27 leads to [17]

$$L'_\psi(t) = -\int_0^{L_\psi(t)} \langle \frac{\partial C}{\partial t}, \psi\kappa\hat{\boldsymbol{n}} - (\boldsymbol{\nabla}\psi \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}} \rangle \, ds. \tag{5.28}$$

Since $L'_\psi(t)$ is a scalar and the integrand is composed of the inner product of vectors, it follows that the inner product is simply a dot product (projection) of the two vectors, which is clearly maximal when they lie in the same direction. Thus, the curve perimeter L_ψ shrinks most rapidly when $\frac{\partial C}{\partial t} = [\psi\kappa - (\boldsymbol{\nabla}\psi \cdot \hat{\boldsymbol{n}})]\hat{\boldsymbol{n}}$. Casting this relationship in a level set framework yields

$$\frac{\partial \varphi}{\partial t} = \psi \|\boldsymbol{\nabla}\varphi\| \left[\boldsymbol{\nabla} \cdot \frac{\boldsymbol{\nabla}\varphi}{\|\boldsymbol{\nabla}\varphi\|} \right] + \boldsymbol{\nabla}\psi \cdot \boldsymbol{\nabla}\varphi, \tag{5.29}$$

and adding the constant inflation term as before gives the complete level set curve evolution formulation [14]

$$\frac{\partial \varphi}{\partial t} = \psi \|\boldsymbol{\nabla}\varphi\| \left[\boldsymbol{\nabla} \cdot \frac{\boldsymbol{\nabla}\varphi}{\|\boldsymbol{\nabla}\varphi\|} + \nu \right] + \boldsymbol{\nabla}\psi \cdot \boldsymbol{\nabla}\varphi. \tag{5.30}$$

In this context, optical flow is utilized solely to provide information about where to initialize the segmentation curve, and so must precede level set contour evolution. Ha et al. use directly the formulation of Horn and Schunck, applying central differencing (Appendix D) for spatial derivatives in the image domain and one-sided differencing between image frames. An explicit gradient descent method was chosen as their iteration scheme to solve for velocities *u* and *v*:

$$\tilde{u}^{n+1} = \tilde{u}^n - \gamma\left[\alpha^2\left(I_x u^n + I_y v^n + I_t\right)I_x - \nabla^2 u^n\right] \tag{5.31}$$

$$\tilde{v}^{n+1} = \tilde{v}^n - \gamma\left[\alpha^2\left(I_x u^n + I_y v^n + I_t\right)I_y - \nabla^2 v^n\right]. \tag{5.32}$$

In Equations 5.31 and 5.32, γ is a perturbation parameter that is tuned based on the image sequence at hand, and is referenced elsewhere in [14] without supplying further details.

Because their work was directed toward real-time image-based tracking of flying vehicles, the primary aim was to maximize computational efficiency. Thus, the authors of [14] made another important contribution by casting the optical flow methodology in a multigrid framework. Multigrid methods essentially increase the speed at which matrix solutions can be found by first solving a given set of equations on a coarse mesh, then using this coarse mesh solution as the initial guess on a finer mesh, and so on recursively until a solution to the original dense problem is realized (Appendix E). This approach also carries with it the distinct advantage of being more likely to find global minima of elliptic equations on the coarse level, reducing the likelihood of a solution being "trapped" in a local minimum when solved strictly on the original dense mesh [12, 14]. Ha et al. found that convergence sped up by an order of magnitude by employing multiple grids in this way.

5.4 Preserving Nonlinearity: Improving the Ability to Track Discontinuities

A notable breakthrough was made in 2004 when Brox et al. reported in *High Accuracy Optical Flow Estimation Based on a Theory for Warping* that a significant contribution was possible in the field with the realization that the multigrid approach offered earlier facilitates the preservation of nonlinearity in the energy functional to be minimized. All works up to this point utilized the optical flow brightness constraint originally derived by Horn and Schunck, which was always linearized by the assumption that (I_x, I_y) is known, and leads to a fixed point iteration on (u, v). In Brox et al., however, it is noted that linearization is only valid if an image changes linearly along a displacement, which is generally not the case. The validity of this assumption becomes particularly tenuous if displacements are large [12]. Thus, grey value constancy is assumed as in previous works, but left in the form

$$I(x, y, t) = I(x + u, y + v, t + 1) \tag{5.33}$$

for $I: \Omega \subset \mathbb{R}^3 \rightarrow \mathbb{R}$.

If $\boldsymbol{x} \coloneqq [x, y, t]^{\mathrm{T}}$ and $\boldsymbol{w} \coloneqq [u, v, 1]^{\mathrm{T}}$, then 5.33 can be written more compactly as [9]

$$I(\boldsymbol{x}) = I(\boldsymbol{x} + \boldsymbol{w}). \tag{5.34}$$

This is an important distinction from simply writing $\frac{dI}{dt} = 0$, which leads to the previous linear formulations of the minimization problem.

In addition to assuming brightness constancy, Brox et al. introduce an additional gradient constancy assumption which states that the relative brightness of an object and its surroundings stays the same regardless of an object's location or the time at which it is being viewed, effectively reducing sensitivity to overall changes in image brightness [12]:

$$\boldsymbol{\nabla} I(\boldsymbol{x}) = \boldsymbol{\nabla} I(\boldsymbol{x} + \boldsymbol{w}). \tag{5.35}$$

With the new constraint imposed on the optical flow field, the brightness error is changed from the standard Horn and Schunck formulation to

$$\varepsilon_b^2 = |I(\boldsymbol{x} + \boldsymbol{w}) - I(\boldsymbol{x})|^2 + \gamma |\boldsymbol{\nabla} I(\boldsymbol{x} + \boldsymbol{w}) - \boldsymbol{\nabla} I(\boldsymbol{x})|^2, \tag{5.36}$$

where γ is a relative weighting between brightness constancy and brightness gradient constancy.

To further reduce sensitivity to violations of the brightness constancy assumption, Equation 5.36 is tempered with an increasing concave function in order to lessen the impact of outliers in the image domain. Based on work by Black and Anandan [10], Brox et al. chose to achieve this sensitivity reduction using a modified L^1 norm function $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$, where ϵ is some small value that prevents problems associated with a machine's attempt to numerically approximate the square root of zero (set to 10^{-3} in their work):

$$\varepsilon_b^2 = \Psi(|I(\boldsymbol{x}+\boldsymbol{w}) - I(\boldsymbol{x})|^2 + \gamma|\nabla I(\boldsymbol{x}+\boldsymbol{w}) - \nabla I(\boldsymbol{x})|^2). \qquad \textbf{(5.37)}$$

In this way, ε_b^2 cannot vary linearly about noisy outlier pixels, but varies less abruptly as the square root of its original value.

The smoothness constraint is also used as in previous works, but is enforced over the entire spatio-temporal domain and is modified in the same way as the brightness constraint to lessen the unwanted effects of noise:

$$\varepsilon_s^2 = \Psi(|\nabla u|^2 + |\nabla v|^2). \qquad \textbf{(5.38)}$$

In Equation 5.38, the divergence operator acts spatially and temporally so that $\nabla := [\partial_x, \partial_y, \partial_t]$. Introducing the usual regularization coefficient α^2, the energy functional to be minimized takes on the familiar form

$$E^2(x,y) = \iiint(\varepsilon_b^2 + \alpha^2\varepsilon_s^2)d\boldsymbol{x}. \qquad \textbf{(5.39)}$$

As with the method of Horn and Schunck before, the goal is to find the functions *u* and *v* that minimize $E^2(x,y)$. The variational formulation leads to a set of Euler-Lagrange equations, which now have the form [12]

$$\Psi'(I_z^2 + \gamma I_{xz}^2 + \gamma I_{yz}^2) \cdot [I_x I_z + \gamma I_{xx} I_{xz} + \gamma I_{xy} I_{yz}]$$
$$-\alpha^2 \nabla \cdot [\Psi'(|\nabla u|^2 + |\nabla v|^2)\nabla u] = 0 \qquad \textbf{(5.40)}$$

$$\Psi'\left(I_z^2 + \gamma I_{xz}^2 + \gamma I_{yz}^2\right) \cdot \left[I_y I_z + \gamma I_{yy} I_{yz} + \gamma I_{xy} I_{xz}\right]$$
$$-\alpha^2 \nabla \cdot [\Psi'(|\nabla u|^2 + |\nabla v|^2)\nabla v] = 0, \qquad \textbf{(5.41)}$$

where $\Psi'(s^2) = \frac{1}{2\sqrt{s^2+\epsilon^2}}$, and the authors have used the following shorthand for clearer representation:

$$I_x \coloneqq \partial_{\mathrm{x}} I(\boldsymbol{x} + \boldsymbol{w})$$
$$I_y \coloneqq \partial_{\mathrm{y}} I(\boldsymbol{x} + \boldsymbol{w})$$
$$I_z \coloneqq I(\boldsymbol{x} + \boldsymbol{w}) - I(\boldsymbol{x})$$
$$I_{xx} \coloneqq \partial_{\mathrm{xx}} I(\boldsymbol{x} + \boldsymbol{w})$$
$$I_{yy} \coloneqq \partial_{\mathrm{yy}} I(\boldsymbol{x} + \boldsymbol{w})$$
$$I_{xy} \coloneqq \partial_{\mathrm{xy}} I(\boldsymbol{x} + \boldsymbol{w})$$
$$I_{xz} \coloneqq \partial_{\mathrm{x}} I(\boldsymbol{x} + \boldsymbol{w}) - \partial_{\mathrm{x}} I(\boldsymbol{x})$$
$$I_{yz} \coloneqq \partial_{\mathrm{y}} I(\boldsymbol{x} + \boldsymbol{w}) - \partial_{\mathrm{y}} I(\boldsymbol{x}).$$

This is the point where novelty is introduced: the Euler-Lagrange equations defined in this way are nonlinear in $\boldsymbol{w}$, so nested fixed-point iterations are performed to linearize the system in terms of velocities and their incremental changes. Starting out, if $\boldsymbol{w}^k \coloneqq [u^k, v^k, 1]^{\mathrm{T}}$ with initialization $\boldsymbol{w}^o = [0,0,1]^{\mathrm{T}}$ at the coarsest grid level, then $\boldsymbol{w}^{k+1}$ is the solution of [12]

$$\Psi'\left([I_z^{k+1}]^2 + \gamma[I_{xz}^{k+1}]^2 + \gamma\left[I_{yz}^{k+1}\right]^2\right) \cdot \left[I_x^k I_z^{k+1} + \gamma I_{xx}^k I_{xz}^{k+1} + \gamma I_{xy}^k I_{yz}^{k+1}\right]$$
$$-\alpha^2 \nabla \cdot [\Psi'(|\nabla u^{k+1}|^2 + |\nabla v^{k+1}|^2)\nabla u^{k+1}] = 0 \qquad \textbf{(5.42)}$$

$$\Psi'\left([I_z^{k+1}]^2 + \gamma[I_{xz}^{k+1}]^2 + \gamma\left[I_{yz}^{k+1}\right]^2\right) \cdot \left[I_y^k I_z^{k+1} + \gamma I_{yy}^k I_{yz}^{k+1} + \gamma I_{xy}^k I_{xz}^{k+1}\right]$$
$$-\alpha^2 \nabla \cdot [\Psi'(|\nabla u^{k+1}|^2 + |\nabla v^{k+1}|^2)\nabla v^{k+1}] = 0. \qquad \textbf{(5.43)}$$

Once a fixed-point solution in $\boldsymbol{w}^k$ is reached, the solution grid is refined and the coarse grid solution will be used as an initialization at the new scale.

However, Ψ' and I_*^{k+1} remain nonlinear, and thus must first be linearized before a solution for the optical flow field may be obtained and used to initialize a refined pixel domain. To accomplish this, 1st order Taylor expansions linearize I_*^{k+1} [12]:

$$I_z^{k+1} \approx I_z^k + I_x^k du^k + I_y^k dv^k \qquad \textbf{(5.44)}$$

$$I_{xz}^{k+1} \approx I_{xz}^k + I_{xx}^k du^k + I_{xy}^k dv^k \qquad \textbf{(5.45)}$$

$$I_{yz}^{k+1} \approx I_{yz}^k + I_{xy}^k du^k + I_{yy}^k dv^k, \qquad \textbf{(5.46)}$$

where

$$u^{k+1} = u^k + du^k \qquad \textbf{(5.47)}$$

$$v^{k+1} = v^k + dv^k. \qquad \textbf{(5.48)}$$

For shorthand, Brox et al. define

$$\begin{aligned}(\Psi')_b^k \coloneqq\ & \Psi'\Big(\big[I_z^k + I_x^k du^k + I_y^k dv^k\big]^2 \\ & +\gamma\big[I_{xz}^k + I_{xx}^k du^k + I_{xy}^k dv^k\big]^2 \\ & +\gamma\big[I_{yz}^k + I_{xy}^k du^k + I_{yy}^k dv^k\big]^2\Big)\end{aligned} \qquad \textbf{(5.49)}$$

and

$$(\Psi')_s^k \coloneqq\ \Psi'(|\boldsymbol{\nabla}(u^k + du^k)|^2 + |\boldsymbol{\nabla}(v^k + dv^k)|^2). \qquad \textbf{(5.50)}$$

They term Equation 5.49 “data term robustness” and Equation 5.50 “smoothness diffusivity,” and with these new shorthand definitions write the Euler-Lagrange equations once again as

$$(\Psi')_b^k \cdot \{I_x^k[I_z^k + I_x^k du^k + I_y^k dv^k]\}$$
$$+\gamma(\Psi')_b^k\{I_{xx}^k[I_{xz}^k + I_{xx}^k du^k + I_{xy}^k dv^k]$$
$$+I_{xy}^k[I_{yz}^k + I_{xy}^k du^k + I_{yy}^k dv^k]\}$$
$$-\alpha^2\nabla \cdot \{(\Psi')_s^k \nabla(u^k + du^k)\} = 0 \qquad \textbf{(5.51)}$$
$$(\Psi')_b^k \cdot \{I_y^k[I_z^k + I_x^k du^k + I_y^k dv^k]\}$$
$$+\gamma(\Psi')_b^k\{I_{yy}^k[I_{yz}^k + I_{xy}^k du^k + I_{yy}^k dv^k]$$
$$+I_{xy}^k[I_{xz}^k + I_{xx}^k du^k + I_{xy}^k dv^k]\}$$
$$-\alpha^2\nabla \cdot \{(\Psi')_s^k \nabla(v^k + dv^k)\} = 0. \qquad \textbf{(5.52)}$$

It is noted that this is still a nonlinear system of equations for fixed point *k*, but now in increments of du^k and dv^k [12]. The remaining nonlinearity in Ψ' is removed using a second inner fixed point iteration loop over a new step *l*, where $du^{k,l}$ and $dv^{k,l}$ denote the variables iterated on, with $du^{k,0} \coloneqq 0$ and $dv^{k,0} \coloneqq 0$ [12]. Now the Euler-Lagrange equations take on their final forms:

$$(\Psi')_b^{k,l} \cdot \{I_x^k[I_z^k + I_x^k du^{k,l+1} + I_y^k dv^{k,l+1}]\}$$
$$+\gamma(\Psi')_b^{k,l}\{I_{xx}^k[I_{xz}^k + I_{xx}^k du^{k,l+1} + I_{xy}^k dv^{k,l+1}]$$
$$+I_{xy}^k[I_{yz}^k + I_{xy}^k du^{k,l+1} + I_{yy}^k dv^{k,l+1}]\}$$
$$-\alpha^2\nabla \cdot \left\{(\Psi')_s^{k,l} \nabla(u^k + du^{k,l+1})\right\} = 0 \qquad \textbf{(5.53)}$$
$$(\Psi')_b^{k,l} \cdot \{I_y^k[I_z^k + I_x^k du^{k,l+1} + I_y^k dv^{k,l+1}]\}$$
$$+\gamma(\Psi')_b^{k,l}\{I_{yy}^k[I_{yz}^k + I_{xy}^k du^{k,l+1} + I_{yy}^k dv^{k,l+1}]$$
$$+I_{xy}^k[I_{xz}^k + I_{xx}^k du^{k,l+1} + I_{xy}^k dv^{k,l+1}]\}$$
$$-\alpha^2\nabla \cdot \left\{(\Psi')_s^{k,l} \nabla(v^k + dv^{k,l+1})\right\} = 0. \qquad \textbf{(5.54)}$$

Applying their method to a set of test images, Brox et al. found the resultant errors in their optical flow fields to be up to 5 times smaller than in those of all previous methods which do not preserve nonlinearity.

5.5 Coupling Image Segmentation and Optical Flow

The idea of preserving nonlinearity was clearly a significant contribution to the original optical flow formulation. However, in their 2006 paper *Piecewise-Smooth Dense Optical Flow via Level Sets*, Amiaz and Kiryati report that even the method of Brox et al. may potentially fail along strong discontinuities in the optical flow field, i.e. where object contours exist [9]. In an attempt to improve upon this situation, they offer yet a further advancement of the field, directly coupling optical flow estimation with active contour segmentation techniques.

Unlike the work of Ha et al., in which optical flow was employed as a means for initializing segmentation contours with little attention given to the accuracy of the flow field itself, Amiaz and Kiryati seek to produce optical flow fields and segmentation contours which concertedly act to represent imaged objects and their motion with fidelity. To this end, they embed the functional developed by Brox et al. directly with an active contour segmentation model. In this way, the optical flow problem is restated as one of determining two piecewise smooth flow fields separated by a contour, simultaneously minimizing contour curve length and the optical flow functional within the segregated smooth regions [9]. The level set method of Chan and Vese for minimizing the Mumford-Shah energy functional was ultimately chosen by Amiaz and Kiryati, because of its ability to produce sharp interfaces.

Recall from Chapter 4, in the section on image segmentation, that Chan and Vese reformulated the Mumford-Shah functional

$$\mathcal{F}_{MS}(I) = \int_{\Omega} f(I)d\boldsymbol{x} + \mu \int_{\Omega \backslash C} s(I)d\boldsymbol{x} + \nu|C| \quad \textbf{(5.55)}$$

so that it would act on a segmentation field possessing two distinct regions; one inside of a segmentation curve (I^-) and one outside (I^+). The segmentation curve separating the two regions represents the zero-level of an evolving level set function φ,

$$\begin{aligned}\mathcal{F}_{CV} = &\int_{\Omega}(I^+ - I_o)^2 H(\varphi)d\boldsymbol{x} + \int_{\Omega}(I^- - I_o)^2 H(-\varphi)d\boldsymbol{x}\\ &+\mu \int_{\Omega}|\boldsymbol{\nabla} I^+|^2 H(\varphi)d\boldsymbol{x} + \mu \int_{\Omega}|\boldsymbol{\nabla} I^-|^2 H(-\varphi)d\boldsymbol{x}\\ &+\nu \int_{\Omega}|\boldsymbol{\nabla} H(\varphi)|^2 d\boldsymbol{x}, \end{aligned} \quad \textbf{(5.56)}$$

$(H(\varphi)$ is the Heaviside function of φ, demarcating I^+ and I^-, and contour length $|C|$ is now represented equivalently by $|\boldsymbol{\nabla} H(\varphi)|)$, and the functional $\mathcal{F}_{CV}$ is minimized by iteratively solving the Euler-Lagrange equations for I^+, I^-, and φ. With a zero level set in place, the formulation of Amiaz and Kiryati follows almost exactly that of Brox et al., only now the optical flow field is solved for an image domain that is defined as being composed of two neighboring regions. Two optical flow fields, $\boldsymbol{w}^+ \coloneqq [u^+, v^+, 1]$ and $\boldsymbol{w}^- \coloneqq [u^-, v^-, 1]$, are defined so that the optical flow functional of Brox et al. takes the form

$$\begin{aligned}&E^2(u^+, v^+, u^-, v^-, \varphi) =\\ &\iiint \Psi(|I(\boldsymbol{x}+\boldsymbol{w}^+) - I(\boldsymbol{x})|^2 + \gamma|\boldsymbol{\nabla} I(\boldsymbol{x}+\boldsymbol{w}^+) - \boldsymbol{\nabla} I(\boldsymbol{x})|^2)\, H(\kappa\varphi)d\boldsymbol{x}\\ &+ \iiint \Psi(|I(\boldsymbol{x}+\boldsymbol{w}^-) - I(\boldsymbol{x})|^2 + \gamma|\boldsymbol{\nabla} I(\boldsymbol{x}+\boldsymbol{w}^-) - \boldsymbol{\nabla} I(\boldsymbol{x})|^2)\, H(-\kappa\varphi)d\boldsymbol{x}\\ &\qquad +\mu \iiint \Psi(|\boldsymbol{\nabla} u^+|^2 + |\boldsymbol{\nabla} v^+|^2)\, H(\varphi)d\boldsymbol{x}\\ &\qquad +\mu \iiint \Psi(|\boldsymbol{\nabla} u^-|^2 + |\boldsymbol{\nabla} v^-|^2)\, H(-\varphi)d\boldsymbol{x}\\ &\qquad +\nu \iiint |\boldsymbol{\nabla} H(\varphi)|d\boldsymbol{x}. \end{aligned} \quad \textbf{(5.57)}$$

The Heaviside function is scaled by parameter $\kappa < 1$ in order to stress variations in the flow near discontinuities. Following the formulation of Chan and Vese, the relative weight between image brightness and smoothness constraints is denoted by μ rather than α^2, but the two are in fact equivalent.

Defining derivatives inside and outside of the zero level set contour,

$$I_x^\pm \coloneqq \partial_{\mathrm{x}} I(\boldsymbol{x} + \boldsymbol{w}^\pm)$$

$$I_y^\pm \coloneqq \partial_{\mathrm{y}} I(\boldsymbol{x} + \boldsymbol{w}^\pm)$$

$$I_z^\pm \coloneqq I(\boldsymbol{x} + \boldsymbol{w}^\pm) - I(\boldsymbol{x})$$

$$I_{xx}^\pm \coloneqq \partial_{\mathrm{xx}} I(\boldsymbol{x} + \boldsymbol{w}^\pm)$$

$$I_{yy}^\pm \coloneqq \partial_{\mathrm{yy}} I(\boldsymbol{x} + \boldsymbol{w}^\pm)$$

$$I_{xy}^\pm \coloneqq \partial_{\mathrm{xy}} I(\boldsymbol{x} + \boldsymbol{w}^\pm)$$

$$I_{xz}^\pm \coloneqq \partial_{\mathrm{x}} I(\boldsymbol{x} + \boldsymbol{w}^\pm) - \partial_{\mathrm{x}} I(\boldsymbol{x})$$

$$I_{yz}^\pm \coloneqq \partial_{\mathrm{y}} I(\boldsymbol{x} + \boldsymbol{w}^\pm) - \partial_{\mathrm{y}} I(\boldsymbol{x}),$$

the data (brightness) constraint equations for $\boldsymbol{w}^+$ at $\varphi > 0$ and $\boldsymbol{w}^-$ at $\varphi < 0$ are written as

$$H(\pm\kappa\varphi)\Psi'\left({I_z^\pm}^2 + \gamma {I_{xz}^\pm}^2 + \gamma {I_{yz}^\pm}^2\right) \cdot \left[I_x^\pm I_z^\pm + \gamma I_{xx}^\pm I_{xz}^\pm + \gamma I_{xy}^\pm I_{yz}^\pm\right]$$
$$-\mu \nabla \cdot [H(\pm\varphi)\Psi'(|\nabla u^\pm|^2 + |\nabla v^\pm|^2)\nabla u^\pm] = 0 \qquad \textbf{(5.58)}$$

$$H(\pm\kappa\varphi)\Psi'\left({I_z^\pm}^2 + \gamma {I_{xz}^\pm}^2 + \gamma {I_{yz}^\pm}^2\right) \cdot \left[I_y^\pm I_z^\pm + \gamma I_{yy}^\pm I_{yz}^\pm + \gamma I_{xy}^\pm I_{xz}^\pm\right]$$
$$-\mu \nabla \cdot [H(\pm\varphi)\Psi'(|\nabla u^\pm|^2 + |\nabla v^\pm|^2)\nabla v^\pm] = 0, \qquad \textbf{(5.59)}$$

and the smoothness constraint equations for $\boldsymbol{w}^+$ at $\varphi < 0$ and $\boldsymbol{w}^-$ at $\varphi > 0$ are

$$\nabla \cdot [\Psi'(|\nabla u^\pm|^2 + |\nabla v^\pm|^2)\nabla u^\pm] = 0 \qquad \textbf{(5.60)}$$

$$\nabla \cdot [\Psi'(|\nabla u^\pm|^2 + |\nabla v^\pm|^2)\nabla v^\pm] = 0. \qquad \textbf{(5.61)}$$

Linearization and numerical discretization (Appendix F) of these equations leads to the discrete two-phase Euler-Lagrange equations for optical flow field velocity:

$$
\begin{aligned}
&H_\Delta(\pm\kappa\varphi^l)(\Psi')_b^{k,l,\pm}\cdot\left\{I_x^{k,\pm}\left[I_z^{k,\pm}+I_x^{k,\pm}du^{k,l+1,\pm}+I_y^{k,\pm}dv^{k,l+1,\pm}\right]\right\}\\
&+\gamma H_\Delta(\pm\kappa\varphi^l)(\Psi')_b^{k,l,\pm}\left\{I_{xx}^{k,\pm}\left[I_{xz}^{k,\pm}+I_{xx}^{k,\pm}du^{k,l+1,\pm}+I_{xy}^{k,\pm}dv^{k,l+1,\pm}\right]\right.\\
&\left.+I_{xy}^{k,\pm}\left[I_{yz}^{k,\pm}+I_{xy}^{k,\pm}du^{k,l+1,\pm}+I_{yy}^{k,\pm}dv^{k,l+1,\pm}\right]\right\}\\
&-\mu\boldsymbol{\nabla}\cdot\left\{H_\Delta(\pm\kappa\varphi^l)(\Psi')_s^{k,l,\pm}\,\boldsymbol{\nabla}(u^{k,\pm}+du^{k,l+1,\pm})\right\}=0 \qquad \textbf{(5.62)}
\end{aligned}
$$

and

$$
\begin{aligned}
&H_\Delta(\pm\kappa\varphi^l)(\Psi')_b^{k,l,\pm}\cdot\left\{I_y^{k,\pm}\left[I_z^{k,\pm}+I_x^{k,\pm}du^{k,l+1,\pm}+I_y^{k,\pm}dv^{k,l+1,\pm}\right]\right\}\\
&\quad+\gamma H_\Delta(\pm\kappa\varphi^l)(\Psi')_b^{k,l,\pm}\left\{I_{yy}^{k,\pm}\left[I_{yz}^{k,\pm}+I_{xy}^{k,\pm}du^{k,l+1,\pm}+I_{yy}^{k,\pm}dv^{k,l+1,\pm}\right]\right.\\
&\quad\left.+I_{xy}^{k,\pm}\left[I_{xz}^{k,\pm}+I_{xx}^{k,\pm}du^{k,l+1,\pm}+I_{xy}^{k,\pm}dv^{k,l+1,\pm}\right]\right\}\\
&\quad-\mu\boldsymbol{\nabla}\cdot\left\{H_\Delta(\pm\kappa\varphi^l)(\Psi')_s^{k,l,\pm}\,\boldsymbol{\nabla}(v^{k,\pm}+dv^{k,l+1,\pm})\right\}=0. \qquad \textbf{(5.63)}
\end{aligned}
$$

Like the authors of previous works, Amiaz and Kiryati use a gradient descent equation for level set evolution:

$$
\begin{aligned}
&\frac{\partial\varphi}{\partial t}=\nu\delta_\Delta(\varphi)\boldsymbol{\nabla}\cdot\left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}\right)\\
&-\mu\delta_\Delta(\varphi)\left[\Psi(|\boldsymbol{\nabla}u^+|^2+|\boldsymbol{\nabla}v^+|^2)-\Psi(|\boldsymbol{\nabla}u^-|^2+|\boldsymbol{\nabla}v^-|^2)\right]\\
&-\kappa\delta_\Delta(\varphi)\left[\Psi(|I(\boldsymbol{x}+\boldsymbol{w}^+)-I(\boldsymbol{x})|^2+\gamma|\boldsymbol{\nabla}I(\boldsymbol{x}+\boldsymbol{w}^+)-\boldsymbol{\nabla}I(\boldsymbol{x})|^2)\right.\\
&\left.-\Psi(|I(\boldsymbol{x}+\boldsymbol{w}^-)-I(\boldsymbol{x})|^2+\gamma|\boldsymbol{\nabla}I(\boldsymbol{x}+\boldsymbol{w}^-)-\boldsymbol{\nabla}I(\boldsymbol{x})|^2)\right], \qquad \textbf{(5.64)}
\end{aligned}
$$

or, more concisely, using the brightness and smoothness errors defined by Brox et al.,

$$
\begin{aligned}
&\frac{\partial\varphi}{\partial t}=\nu\delta_\Delta(\varphi)\boldsymbol{\nabla}\cdot\left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}\right)\\
&-\mu\delta_\Delta(\varphi)[\Psi(\varepsilon_s^2)^+-\Psi(\varepsilon_s^2)^-]\\
&-\kappa\delta_\Delta(\varphi)[\Psi(\varepsilon_b^2)^+-\Psi(\varepsilon_b^2)^-]. \qquad \textbf{(5.65)}
\end{aligned}
$$

In Equations 5.64 and 5.65, δ_Δ is the numerical approximation to the delta function $\delta(\varphi) = H'(\varphi)$:

$$\delta_\Delta(x) = H'_\Delta(x) = \frac{1}{\pi}\frac{\Delta}{\Delta^2+x^2}. \quad \textbf{(5.66)}$$

Finally, the discrete version of the two-phase level set equation is (Appendix G)

$$\varphi_{i,j}^{l+1} = \frac{1}{C}\left[\varphi_{i,j}^{l} + m\left(C_1\varphi_{i+1,j}^{l} + C_2\varphi_{i-1,j}^{l} + C_3\varphi_{i,j+1}^{l} + C_4\varphi_{i,j-1}^{l}\right)\right.$$
$$-\Delta t\mu\delta_\Delta\left(\varphi_{i,j}^{l}\right)\cdot\left(g_{i,j}^{l,+} - g_{i,j}^{l,-}\right)$$
$$\left.-\Delta t\kappa\delta_\Delta\left(\kappa\varphi_{i,j}^{l}\right)\cdot\left(f_{i,j}^{l,+} - f_{i,j}^{l,-}\right)\right. . \quad \textbf{(5.67)}$$

The Euler-Lagrange equations for optical flow and the level set evolution equations are iteratively solved in alternation to convergence, giving a final two-phase optical flow field of

$$\boldsymbol{w} = \begin{cases} \boldsymbol{w}^{+}, & \varphi > 0 \\ \boldsymbol{w}^{-}, & otherwise \end{cases}, \quad \textbf{(5.68)}$$

separated by a level set isocontour. Thus, in the Amiaz and Kiryati formulation, segmentation contours that distinguish the boundaries of moving objects are coupled with optical flow results, and those optical flow results are in turn updated with each reconfiguration of the segmentation contours in a two-way coupling relationship. However, one drawback of maintaining two-way coupling in this manner is that any large displacements in an image field from one frame to the next can result in a high computational cost, since the optical flow field and the segmentation contours must be iteratively evolved together with each new image frame.

5.6 Conclusions

Each improvement upon the original methodology of Horn and Schunck has led the field of optical flow stepwise toward usefulness as a means of quantifying the necessary conditions on moving boundaries for the purpose of performing CFD simulations. Arguably the most notable contribution among those surveyed was the preservation of nonlinearity introduced by Brox et al., which they showed to be considerably more accurate than the original linear method when applied to test images with known solutions [12].

The two-phase method of Amiaz and Kiryati promises some improvement over the single-phase approach of Brox et al., but implementing it requires evolving a segmentation contour iteratively over potentially many time steps; we already adopted in the previous chapter a smooth version of the rapid k-means approach of Gibou and Fedkiw precisely to avoid this incremental level set evolution process, and it demonstrated an ability to produce good segmentation results for our purposes (following a simplified version of Vese and Chan’s formulation) while only requiring a few iterations to reach convergence. For this reason, we have opted for now to retain the segmentation methods already in place in our modeling approach, and to use single-phase nonlinear optical flow, the method of Brox et al., as a means for setting boundary conditions on segmentation contours in an Eulerian sense, obviating the need for the previous point tracking method and its associated assumptions and tedium.

Interestingly, nobody has yet incorporated the optical flow and segmentation methods described here into a complete framework for constructing CFD models. Thus, the adaptation of optical flow to this context stands as a novel development in the way of enabling on-the-fly quantification of complex moving boundaries and their influence on a surrounding fluid environment. In the next chapter, the development and testing of optical flow algorithms is discussed, first using a series of test images, and then finally with some real images of interest to modeling.

1/12	**1/6**	**1/12**
1/6	**-1**	**1/6**
1/12	**1/6**	**1/12**

Figure 5-1. Visual representation of the Laplacian approximation [15].

CHAPTER 6

DEVELOPMENT, TESTING, AND APPLICATION OF OPTICAL FLOW

6.1 Introduction

In the previous chapter, optical flow was introduced as a method of tracking the motions of brightness patterns between image frames. Horn and Schunck [15] introduced the method in 1981, proposing image brightness as a conserved quantity in a moving image field. By placing a second constraint of smoothness on image pattern motion, they formulated a functional which, when minimized, leads to a set of coupled, linear Euler-Lagrange equations.

In 2004, Brox et al. improved upon Horn and Schunck's optical flow method by preserving nonlinearity in their formulation and introducing the gradient of image brightness as a quantity to be conserved along with the brightness, itself [12]. The result was a new pair of coupled, nonlinear equations that must be solved using nested fixed-point iteration schemes as described in Chapter 5.

In our ongoing effort to move away from the use of Lagrangian points in image based models, a complete optical flow package has been developed, including both the Horn-Schunck formulation (developed as a basis for comparison, and because of its continued persistence in the modern literature) and that of Brox et al. (developed because of its promise of marked improvement over linear formulations). This chapter outlines the development process, and is organized into five distinct sections: Section 6.2 briefly discusses the development of the optical flow algorithms, themselves. Section 6.3 covers optical flow testing, in which both the methods of Horn and Schunck and of Brox et al. were thoroughly evaluated by applying them to a set of synthetically generated test images. Section 6.4 moves toward application of optical flow to real image sequences, where the swimming American eel video is used to demonstrate the abilities of each optical flow method. Section 6.5 also applies optical flow to a video sequence, but this time it is one that captures red blood cells flowing through an experimental channel setup, so that optical flow is demonstrated on a set of image frames featuring many small objects rather than one large object (as has been the case up to now, i.e. the eel video). Finally, Section 6.6 applies the optical flow method once again to images of small particles, but now within the context of PIV image pairs, acting as a gauge of performance in situations where imaged objects are very small—on the order of a couple pixels—and the intensity gradients are quite high.

6.2 Development of the Algorithms

A complete optical flow code was written in Fortran 90, and included the option of solving the optical flow field using either the linear formulation of Horn and Schunck or the nonlinear method of Brox et al. so that the two could be easily compared. The discretized Horn and Schunck equations were modified somewhat, so that they could be left in matrix form and solved using the same successive over-relaxation algorithm employed in solving the nonlinear equations. Rather than estimating Laplacians as

$$\nabla^2 u = \bar{u} - u \qquad \textbf{(6.1)}$$

and

$$\nabla^2 v = \bar{v} - v \qquad \textbf{(6.2)}$$

with $\bar{u}$ and $\bar{v}$ representing velocity components averaged over the 9-point stencil suggested by Horn and Schunck and supplied in the previous chapter's Figure 5-1, a standard 5-point stencil was used (which, curiously, turned out to give better results than the original formulation). So the Laplacians of velocity are now estimated as

$$\nabla^2 u = u_E + u_W + u_N + u_S - 4u_P \tag{6.3}$$

and

$$\nabla^2 v = v_E + v_W + v_N + v_S - 4v_P, \tag{6.4}$$

with $E, W, N,$ and S subscripts indicating pixel-center values in the four cardinal directions, and subscript P denoting the pixel-center value being solved for. Rewriting the Horn and Schunck equations (Equations 5.10 and 5.11 in the previous chapter) in terms of u-velocity for the first one and v-velocity for the second one,

$$\alpha^2\nabla^2 u - I_x^2 u = I_x\left(I_y v + I_t\right) \tag{6.5}$$

$$\alpha^2\nabla^2 v - I_y^2 u = I_y(I_x u + I_t), \tag{6.6}$$

and using the Laplacian estimates given by Equations 6.3 and 6.4 leads to the new set of coupled Euler-Lagrange equations

$$u_E + u_W + u_N + u_S - \left(4 + \frac{I_x^2}{\alpha^2}\right)u_P = \frac{I_x}{\alpha^2}\left(I_y v_P + I_t\right) \tag{6.7}$$

and

$$v_E + v_W + v_N + v_S - \left(4 + \frac{I_y^2}{\alpha^2}\right)v_P = \frac{I_y}{\alpha^2}(I_x u_P + I_t). \tag{6.8}$$

These optical flow equations are now cast in a convenient form to be solved using point successive over-relaxation (PSOR):

$$u_P^{n+1} = (1-\omega)u_P^n + \frac{\omega}{A_P}(b - A_N u_N^n - A_E u_E^n - A_S u_S^{n+1} - A_W u_W^{n+1}) \tag{6.9}$$

$$v_P^{n+1} = (1-\omega)v_P^n + \frac{\omega}{A_P}(b - A_N v_N^n - A_E v_E^n - A_S v_S^{n+1} - A_W v_W^{n+1}). \tag{6.10}$$

Here, ω is an over-relaxation parameter to help speed up convergence (set to 1.99 for all cases in this thesis) and neighbor coefficient matrix entries $A_N \ldots A_W = 1$. In the u-velocity equation (Equation 6.9), $A_P = -4 - I_x^2/\alpha^2$ and $b = (I_x/\alpha^2)\left(I_y v_P^n + I_t\right)$, while the v-velocity equation (Equation 6.10) has $A_P = -4 - I_y^2/\alpha^2$ and $b = \left(I_y/\alpha^2\right)(I_x u_P^{n+1} + I_t)$.

Obviously, the nonlinear optical flow equations of Brox et al. are considerably more complicated, but they are still set up in exactly the same way. Recall from Chapter 5 that the Euler-Lagrange equation for incremental u-velocity update du was

$$\begin{aligned}
&\frac{(\Psi')_b^{k,l}}{\alpha^2} \cdot \left\{I_x^k\left[I_z^k + I_x^k du^{k,l+1} + I_y^k dv^{k,l+1}\right]\right\} \\
&+\gamma(\Psi')_b^{k,l}\{I_{xx}^k\left[I_{xz}^k + I_{xx}^k du^{k,l+1} + I_{xy}^k dv^{k,l+1}\right] \\
&+I_{xy}^k\left[I_{yz}^k + I_{xy}^k du^{k,l+1} + I_{yy}^k dv^{k,l+1}\right]\} \\
&= \nabla \cdot \left\{(\Psi')_s^{k,l}\, \nabla(u^k + du^{k,l+1})\right\}. \qquad \textbf{(6.11)}
\end{aligned}$$

We can rewrite the divergence term in discrete form as

$$\begin{aligned}
\nabla \cdot \left\{(\Psi')_s^{k,l}\, \nabla(u^k + du^{k,l+1})\right\} = {} & (\Psi')_{s,n}^{k,l}\left[\left(u_N^k + du_N^{k,l,m+1}\right) - \left(u_P^k + du_P^{k,l,m+1}\right)\right] \\
&+(\Psi')_{s,e}^{k,l}\left[\left(u_E^k + du_E^{k,l,m+1}\right) - \left(u_P^k + du_P^{k,l,m+1}\right)\right] \\
&+(\Psi')_{s,s}^{k,l}\left[\left(u_S^k + du_S^{k,l,m+1}\right) - \left(u_P^k + du_P^{k,l,m+1}\right)\right] \\
&+(\Psi')_{s,w}^{k,l}\left[\left(u_W^k + du_W^{k,l,m+1}\right) - \left(u_P^k + du_P^{k,l,m+1}\right)\right], \qquad \textbf{(6.12)}
\end{aligned}$$

where uppercase letter subscripts are pixel center values and lowercase letter subscripts are pixel face values in the four cardinal directions. This then allows the Euler-Lagrange equation for du to be written in PSOR form as

$$du_P^{k,l,m+1} = (1-\omega)du_P^{k,l,m} + \frac{\omega}{A_P}\left[b - A_N^{k,l}\left(u_P^k - u_N^k - du_N^{k,l,m}\right)\right.$$

$$-A_E^{k,l}\left(u_P^k - u_E^k - du_E^{k,l,m}\right) - A_N^{k,l}\left(u_P^k - u_N^k - du_N^{k,l,m}\right)$$

$$-A_W^{k,l}\left(u_P^k - u_W^k - du_W^{k,l,m+1}\right) - A_S^{k,l}\left(u_P^k - u_S^k - du_S^{k,l,m+1}\right), \qquad \textbf{(6.13)}$$

with

$$A_N = (\Psi')_{s,n}^{k,l},$$

$$A_S = (\Psi')_{s,s}^{k,l},$$

$$A_E = (\Psi')_{s,e}^{k,l},$$

$$A_W = (\Psi')_{s,w}^{k,l},$$

$$A_P = A_N + A_S + A_E + A_W + \frac{(\Psi')_{b,P}^{k,l}}{\alpha^2}\left[I_x^2 + \gamma\left(I_{xy}^2 + I_{xx}^2\right)\right],$$

and

$$b = -\frac{(\Psi')_{b,P}^{k,l}}{\alpha^2}\left\{I_x\left(I_y dv_P^{k,l,m} + I_t\right) + \gamma\left[I_{xx}\left(I_{xy} dv_P^{k,l,m} + I_{xt}\right) + I_{xy}\left(I_{yy} dv_P^{k,l,m} + I_{yt}\right)\right]\right\}.$$

The PSOR equation for dv follows similarly, with

$$A_P = A_N + A_S + A_E + A_W + \frac{(\Psi')_{b,P}^{k,l}}{\alpha^2}\left[I_y^2 + \gamma\left(I_{xy}^2 + I_{yy}^2\right)\right]$$

and

$$b = -\frac{(\Psi')_{b,P}^{k,l}}{\alpha^2}$$

$$\cdot\left\{I_y\left(I_x du_P^{k,l,m+1} + I_t\right) + \gamma\left[I_{xy}\left(I_{xx} du_P^{k,l,m+1} + I_{xt}\right) + I_{yy}\left(I_{xy} du_P^{k,l,m+1} + I_{yt}\right)\right]\right\}.$$

Intensity derivatives are computed using a central difference scheme, with one-sided differencing applied at domain boundaries (Appendix H).

Including the PSOR solver, the method of Brox et al. actually requires *two* nested iteration loops: the matrix equations for du and dv are each iterated to convergence in the PSOR solver in loop m; PSOR is called inside loop l, which converges on values of

du and dv for fixed velocity values u and v; and u and v are themselves updated to convergence in loop k according to

$$u^{k+1} = u^k + du^k \tag{6.14}$$

and

$$v^{k+1} = v^k + dv^k. \tag{6.15}$$

The data term robustness and smoothness diffusivity terms are updated in the k loop with each update of the velocity components u and v.

As suggested in Brox et al., calculating optical flow fields in a multi-resolution manner on grids of increasing levels of refinement helps to reduce the chances of an optical flow solution becoming trapped in a local minimum that does not satisfy global minimization of the constraint equations and speeds convergence [12]. Thus, for both the Horn-Schunck and Brox methods, optical flow was solved on 3 different grid levels of side length 32, 64, and 128 pixels, respectively, with each coarse solution providing the initial conditions interpolated onto the grid at the next level of refinement. Although this strategy was not proposed by Horn and Schunck in their work, it has been adopted here for both of the optical flow methods in order to provide a fair basis for comparison.

6.3 Optical Flow Testing

Both the Horn-Schunck and Brox et al. methods were tested extensively on a set of test images (Figure 6-1) before later applying them to real image sequences, in a process intended to highlight the abilities and limitations of each. Test images were all of dimension 128×128 pixels, and generated by a Fortran 90 code that was designed so that various shapes and types of motion between test image frames could be prescribed.

This way, the amount of error contained in a particular optical flow solution could easily be calculated by directly comparing the optical flow result with the prescribed motion.

In all of the test image cases, the smoothness weight α^2 in the Euler-Lagrange equations was set to 10.0, with the gradient weight (the weighting of the nonlinear terms) set to $\gamma = 1.0$ in the Brox equations. A set of three test frames was generated for each case, and the optical flow fields computed between frames 2 and 3 were used for error analysis (the Euler-Lagrange equations derived by Brox et al. contain temporal derivatives in the optical flow field, itself, that rely upon the existence of a prior optical flow solution; the temporal derivative is typically neglected for the first image pair, but this yields a slightly less accurate result [12]).

The images used for testing that are illustrated in Figure 6-1 are presented in increasing order of complexity with regard to geometry and motion, or a combination thereof, in an effort to systematically identify specific situations that lead to the success or failure of each optical flow algorithm. The optical flow field results computed in select cases are supplied in Table 6-1 and Table 6-2 at the end of this section, with Table 6-1 giving vector magnitude and angular errors averaged over the entire image domain, and Table 6-2 showing the same quantities averaged only over pixels containing boundaries (i.e. the zero level of the signed distance field test image, the central contour of the smooth Heaviside functions, etc).

6.3.1 Translational Motion of a Circular Level Set Field

The first of the test image sets contains a circular signed distance field, defined by the function

$$I(r) = r - R, \qquad \textbf{(6.17)}$$

with r representing radial distance from a defined central location $\boldsymbol{x}$, and R denoting a constant radius at which $I(R) = 0$. The field was prescribed with a translational motion (by defining a centroid velocity $\dot{\boldsymbol{x}}$) that remained constant and equal in the horizontal and vertical directions ($U = V = \text{const}$). This was initially regarded to be the simplest case, not only because of its smooth geometry and constant linear motion, but because the brightness gradient is radially constant and defined everywhere in the image domain.

Both optical flow methods were tested first with a small field displacement (whereby U and V were both set to 1.0 $pixel/frame$ so that the brightness field would be displaced 1 pixel in the x-direction and 1 pixel in the y-direction on each image pair), and then with a large displacement (U and V both set to 5.0 $pixels/frame$). Figure 6-2 and Figure 6-3 illustrate the results generated by each of the methods when given large field displacements, and Table 6-1 and Table 6-2 quantify the average errors. In each figure, the exact solution is shown in (A), along with the zero-level contour of the image field being translated. Figure 6-4 and Figure 6-5 (B) show the optical flow results of each method, with the velocity magnitude error (as a percentage of the exact velocity magnitude) shown in (C) and the angular error of the optical flow vectors (measured in degrees) given in (D). Immediately obvious from the figures and the tables is the fact that the method of Brox et al. produces significantly lower errors in both vector magnitude and direction, particularly along the boundaries of the zero-level. The velocity error magnitude is ~1.5% near the zero-level contour, compared with almost 7% using the Horn-Schunck method. In the small displacement case, both methods perform better, but the errors produced by the nonlinear method are still around an order of magnitude smaller. As will be shown later, this becomes particularly important when applying optical flow to imaged object boundary motion, as it is obviously of interest to supply the most accurate boundary conditions possible when coupling interfacial motion with a CFD solver.

6.3.2 Translational Motion of a Circular Heaviside Function

This case was set up identically to the previous one, but with a circular numerical Heaviside function defined as

$$I(r) = s\,\mathcal{H}(r - R), \qquad \textbf{(6.18)}$$

where s is a scaling constant (set to 10.0), comprising the image field rather than a signed distance level set function. Here the level of difficulty has increased, because strong gradients are now only defined within the contours of the circle near its prescribed radius of 32 pixels, with much smaller intensity gradients presenting near the edges of the image domain. Even with a small displacement of $U = V = 1.0\,pixel/frame$, the Horn-Schunck method is now beginning to show signs of difficulty, with magnitude errors nearly 5 times higher than they were in the previous image pair with the signed distance function (Figure 6-6 and Figure 6-7). Given the relatively large displacement magnitude of $U = V = 5.0\,pixels/frame$, the Horn-Schunck method began to produce large errors in its solution (Figure 6-4) that would simply be unusable for modeling, with velocity magnitude errors falling around 75%. Angular error was much smaller, averaging $< 2°$, but the result is still unacceptable for use in setting boundary conditions.

The method of Brox et al., on the other hand, consistently produced velocity magnitude errors well under 0.5% with angular errors of less than 0.1°; one of the benefits of preserving nonlinearity as reported in [12] is that it gives the method a greater ability to handle large object displacements in the absence of strong gradients over large portions of the image domain. This appears to be the case here, as well, with the Brox method showing a high level of accuracy even while the Horn-Schunck method was found to underestimate the velocity by more than half when faced with a large displacement field.

6.3.3 Translational Motion of a Discrete Circular Shape

Next, the demands placed on each optical flow calculation method were increased once again, by providing a discrete circular shape of constant zero intensity (black) against a white uniform background (intensity level of 255) as the translating object, in place of a circle or circular field with smooth gradients:

$$I(r) = \begin{cases} 255, r > R \\ 0, r \leq R \end{cases}. \tag{6.19}$$

In this case, even with $U = V = 1.0\, pixel/frame$, the Horn-Schunck method began to show considerable difficulty reaching an accurate result; this is easily visible comparing the exact solution of Figure 6-8 (A) with the Horn and Schunck optical flow solution illustrated in Figure 6-8 (B). Velocity magnitude error was more than 100% on average, with vector angular errors averaging close to 30° on the circle's boundaries. The Brox method continued to perform well, giving an average velocity magnitude error of less than 1% on the edges of the circle, with an average angular error of much less than 1°. As can be seen in Figure 6-10 and Figure 6-11, introducing larger object displacements by setting $U = V = 5.0\, pixels/frame$, the situation was considerably worsened for the Horn-Schunck method, while the Brox method remained robust, giving similarly small errors in velocity magnitude and vector direction to the case with smaller displacements. This further supports the assertion put forth by Brox et al. that their nonlinear formulation is not only much better at handling large displacements, as shown in the previous cases, but is also much better equipped to handle discontinuities in the image field.

6.3.4 Complex Geometry and Motion: The Heaviside Star Function

The last test image features a 7-pointed star shape (defined in [78]) which was given a numerical Heaviside function contour here to provide a smooth but steep edge gradient,

$$I(r) = s\mathcal{H}(r - R - A\sin 7\theta), \tag{6.20}$$

with the Heaviside function defined in the usual manner by $\mathcal{H}(x) = \frac{1}{2}\left(1 + \frac{2}{\pi}\arctan\frac{x}{\Delta h}\right)$, r denoting the radial distance from the shape's centroid, R representing a constant radial distance corresponding to the center of the Heaviside function's smooth contour, and s and A scaling the magnitude of the Heaviside function and sine wave amplitude, respectively. (These were set as $s = 10$ and $A = 0.65$ here.) This star shape was chosen because it features regions of high curvature near its edges and center, and its "arms" point in seven different directions, giving rise to multiple radial and azimuthal gradients that allow for both linear and angular motion, and combinations thereof, to be defined and imposed.

The first tests were run by prescribing a simple translational motion (with a large displacement) on the shape by setting $U = V = 5.0\, pixels/frame$. Visual results are given in Figure 6-12 and Figure 6-13 in the usual order, with error quantities listed in

Table 6-1 and Table 6-2. The method of Brox et al. continued to perform well as in previous cases, with an average magnitude error under 0.2% and average angular error under 0.05°. The Horn-Schunck method continued to produce much larger errors, even more so than in previous translational cases with the introduction of this more complex geometry, and with some of the largest errors occurring on the object's edges where having a valid solution is the most critical for our purposes.

Pure rotational motion (Figure 6-14 and Figure 6-15) appears to be a much greater challenge to these optical flow algorithms than translational motion, even when applying the nonlinear method. The error produced by the Horn-Schunck formulation was substantial nearly everywhere in the image domain, and greatest at the shape's edges, but even the method of Brox et al. produced magnitude errors averaging 20% and angular errors around 6° - although this error was predominantly located away from the edges of the shape (Figure 6-15 (C) and (D)). Even so, it would be difficult to justify using such results to set boundary conditions when the scale of the error so nearly approaches the scale of the motion being described.

As a final test of these methods' abilities, two types of motion were imposed on the Heaviside star shape simultaneously: translation with $U = V = 5.0\, pixels/frame$, and rotation with $\dot{\theta} = \frac{\pi}{24} rad/frame$. Here, the method of Horn and Schunck actually performed somewhat more favorably than it did when the star shape was undergoing pure rotation, but still gave unusable results in terms of setting boundary conditions (Figure 6-16). In this case, the Brox method began to give larger errors (Figure 6-17), though mostly limited to an area in the upper left part of the image. Decreasing the linear displacement by setting $U = V = 1.0\, pixel/frame$ improved the results obtained using the Brox method, although the reliability of these vectors in setting boundary conditions for such cases exhibiting the greatest degree of complexity in geometry and motion is still questionable.

Table 6-1. Average optical flow vector magnitude and angular errors computed over the entire image domain for selected test cases.

Test Case	**Motion Type**	**Optical Flow Method**	**RMS Image Magnitude Error %**	**RMS Image Angular Error °**
Circular Signed Distance Field	Translating-Small Displacement	Horn-Schunck	4.19	0.58
		Brox	0.57	0.10
	Translating-Large Displacement	Horn-Schunck	6.51	2.75
		Brox	2.71	0.32
Circular Heaviside Field	Translating-Small Displacement	Horn-Schunck	7.22	0.19
		Brox	4.83E-02	1.33E-02
	Translating-Large Displacement	Horn-Schunck	74.18	0.93
		Brox	4.82E-02	1.03E-02
Discrete Circle	Translating-Small Displacement	Horn-Schunck	104.05	23.03
		Brox	0.90	0.19
	Translating-Large Displacement	Horn-Schunck	124.69	24.41
		Brox	0.72	0.19
Heaviside Star	Translating-Large Displacement	Horn-Schunck	73.84	27.25
		Brox	8.98E-02	2.09E-02
	Rotating and Translating-Large Displacement	Horn-Schunck	325.78	28.24
		Brox	155.05	13.05

Table 6-2. Average optical flow vector magnitude and angular errors computed over pixels containing object boundaries for selected test cases.

Test Case	**Motion Type**	**Optical Flow Method**	**RMS Boundary Magnitude Error %**	**RMS Boundary Angular Error °**
Circular Signed Distance Field	Translating-Small Displacement	Horn-Schunck	1.32	0.59
		Brox	0.26	5.75E-02
	Translating-Large Displacement	Horn-Schunck	6.52	2.86
		Brox	1.52	0.27
Circular Heaviside Field	Translating-Small Displacement	Horn-Schunck	7.21	0.32
		Brox	2.37E-02	7.49E-03
	Translating-Large Displacement	Horn-Schunck	74.7	1.4
		Brox	2.28E-02	7.23E-03
Discrete Circle	Translating-Small Displacement	Horn-Schunck	101.24	25.83
		Brox	0.93	0.18
	Translating-Large Displacement	Horn-Schunck	125.33	26.44
		Brox	0.87	0.26
Heaviside Star	Translating-Large Displacement	Horn-Schunck	83.07	23.22
		Brox	0.15	4.37E-02
	Rotating and Translating-Large Displacement	Horn-Schunck	105.66	25.96
		Brox	30.21	7.62

Overall, however, the method of Brox et al. performed well on most of the test images given the large displacements and complex geometries found in some of them. The results trend toward reliability when small displacements and smooth contours are involved, so it is anticipated that optical flow should provide an accurate depiction of motion when applied to the images we are interested in modeling.

6.4 Optical Flow Applied to Real Video Data: The Swimming American Eel

With both optical flow methods developed and tested on a number of different geometries and kinematic conditions, they were then applied to the video sequence of the swimming American eel, which was described in detail in Chapter 3. First the image frames were denoised using the SRAD approach described in Chapter 4, with q_o set to an empirically favorable value of 0.3. After smoothing, each frame was segmented following the method of Gibou and Fedkiw, also described in detail in Chapter 4, with the inside weighting coefficient λ_1 set to 1.0 and the outer weighting coefficient λ_2 set to 3.0. (This had the effect of biasing the segmentation contour toward the outermost edge of the intensity contours describing the eel's body, which, due to poor contrast, are represented by a smooth gradient rather than a sharp boundary.) After segmentation, the zero-level contour position was calculated as described in Chapter 4 and a narrow-band level set field was constructed about the zero-contour using the fast marching method [24]. This narrow-band level set field is illustrated along with the zero-contour for one of the frames of the image sequence in Figure 6-19.

As mentioned in Chapter 3, one complete cycle of the eel's tail beat during swimming was captured in 36 video frames, thus giving 35 frame pairs (or 1 frame pair and 34 frame triplets with the Brox method) with which to calculate a set of optical flow field solutions. The optical flow was calculated for the entire tail beat cycle with both the

Horn-Schunck and Brox methods, setting the smoothness constraint coefficient to $\alpha^2 = 10.0$ for each, and the coefficient on the nonlinear terms to $\gamma = 1.0$ for the Brox method. An example optical flow field, calculated using frames 8 through 10 of the sequence using the method of Brox et al., is illustrated in Figure 6-20.

Because the correct vector solution for this sequence of images could not be known a priori, the performance of each of the optical flow methods was evaluated by comparing the displaced pixel values $I^m(x + u, y + v)$, resulting from iterating on the Euler-Lagrange equations until convergence, to the pixel values in their target frame (the next frame in the sequence, $I^{m+1}(x, y)$); a perfect optical flow solution should result in the two being identical everywhere in the image domain. Figure 6-21 and Figure 6-22 illustrate the zero-level contours for the displaced pixel field (black line) and the target pixel field (grey line) along with the optical flow vectors obtained using each method. It is immediately obvious that the Horn-Schunck approach was not able to produce the correct updated pixel intensity field using the optical flow vectors solved for, clearly demonstrating its limited usefulness in supplying boundary conditions to a modeled object. However, the method of Brox et al. resulted in a displaced image field that closely matched its target, with maximum spatial deviations turning out to be much less than one pixel. Thus, the nonlinear optical flow method of Brox et al. appears to be a justifiable means of supplying boundary conditions without the use of Lagrangian points for this set of images.

6.5 Optical Flow Applied to Particle Fields: Red Blood Cells

Another promising application of optical flow lies in the field of multiphase flows. The ability to track the motion of particles is important to the understanding of many phenomena of interest to engineering, such as the transport of pollutants, gastrointestinal mixing for nutrient absorption, and cardiovascular flows. For this work, optical flow was used to obtain preliminary results tracking the motion of red blood cells through a blocked channel, imaged during an experiment overseen by Dr. K.B. Chandran at the University Of Iowa Department Of Biomedical Engineering.

In this experiment, RBCs at a hematocrit of 0.5 were allowed to flow through a channel of 500 microns width and 100 microns depth, constricting step-wise to a width of 200 microns (Figure 6-23 (A)). The fluid flow behavior was completely laminar, with a Reynolds number of $\sim\mathcal{O}(100)$. In the image sequence, flow proceeds from left to right and, by virtue of its low Reynolds number, features a relatively thick boundary layer near the walls in which particle velocities appear to follow a nearly parabolic profile. Particle velocities are greatest near the top of the image, particularly within the constriction where the fluid clearly must accelerate to maintain its constant flow rate.

The optical flow methods of Horn and Schunck, and of Brox et al., were applied to several image frames in the video sequence, with one result from each method illustrated in Figure 6-23 (B) and (C). Immediately notable is the completely random nature of the Horn-Schunck result – virtually no organized information about the flow characteristics is present whatsoever, with the exception of perhaps a small region in the corner of the channel constriction near the lower right side of the image. A likely cause for this is the small size and relatively low density of the particles; object displacement is large between frames relative to the size of the objects, themselves, and, as was shown in the test image section leading this chapter, the linear Horn-Schunck equations are not able to handle large displacements very well, even when geometries are simple. However, in the near-wall boundary layer the particle motion is very small, so that each particle is displaced little from one frame to the next and the Horn-Schunck equations have less difficulty finding a plausible solution.

The results produced by the method of Brox et al. are qualitatively encouraging over the majority of the image, but unfortunately become nonsensical near the channel constriction, where the fluid motion becomes more vigorous. Close-up views of this region reveal that aliasing is likely a key reason for the lack of a realistic solution here—the amount of particle displacement between two image frames in this region is roughly the same as the particle spacing, itself. With all of the particles possessing such similar features in the image, it is impossible for the algorithm to correlate particle positions between frames in this part of the image domain. Experience in this work has taught through repetition that if human eyes and reasoning abilities cannot discern the motions of patterns in an image, then a computer algorithm has very little hope of doing the same; close-up views of the constricted channel region indeed provide no reliable visual cues about the motion occurring there.

Obviously, much more investigation remains to be done with respect to quantifying motion in this type of image field, but these preliminary results hold promise. The aliasing seen in the channel constriction will create difficulties regardless of the approach taken in attempting to accurately describe behavior there, and so must be eliminated as nearly as possible during the image acquisition process. A denser particle field might alleviate some difficulties, but that would require changing the physics of the experiment; the best solution is probably to simply increase the image acquisition rate, so that the particle displacement seen in image pairs is reduced. With the aliasing problem removed, optical flow may offer a route to quantifying the behaviors of particle flows with a level of simplicity not offered by other methods.

6.6 Optical Flow Applied to Particle Fields: PIV

The final application of optical flow considered in this work also deals with particles, now in the context of particle image velocimetry (PIV). PIV is a flow visualization technique in which small reflective particles being transported in an experimental fluid flow are illuminated with a laser sheet and photographed at two instants that are separated by some small time interval, the length of which is determined by the properties of the flow being imaged. A figure taken from [79] (Figure 6-24) illustrates how a typical PIV experiment is set up.

Standard PIV algorithms employ statistical correlation functions that are evaluated in multi-pixel windows in order to determine the most statistically likely motion of particles contained within each window. Particle displacement during the time interval between two image frames being captured leads to changes in the brightness pattern from one image (I) to the next (I'), as illustrated by Figure 6-25. Typically, these brightness patterns are cross-correlated over the pixel range of each interrogation window in the image domain, and the strongest correlation peak is used to determine a PIV vector. The peak’s direction of displacement from the center of the interrogation window gives the resultant vector's direction, and the distance it is displaced gives the vector's magnitude (Figure 6-26 and Figure 6-27). For more details, the reader is referred to the comprehensive overview on PIV methods given in [79].

One of the drawbacks to the standard correlation-based PIV approach is that it produces a vector field that is usually significantly less dense than the pixel count of the image pair being evaluated. Interrogation windows must be large enough so that several particles remain within their bounds from one image to the next; otherwise, there will be insufficient correlation data and the information required for constructing a vector will not exist. This has led several authors to propose instead using optical flow methods as a means to track particle motion in PIV image pairs, thereby giving a velocity vector for every pixel in the image domain. See, for example [80], [81], [82], and [83].

One of the reasons the optical flow methods of Horn and Schunck and of Brox et al. were compared to such a large extent earlier in this chapter is that linear formulations are still quite prevalent in the literature. In [84], for example, the authors employ a Horn-Schunck based optical flow algorithm, but it requires a multi-resolution Gaussian filter scheme with many tunable parameters just to produce an acceptable vector field. As was shown in previous sections, their heavy reliance on image smoothing was probably necessary because of the discontinuous nature of the intensity field at particle boundaries and because of the potentially large displacements undergone by particles relative to their size. By smoothing with multiple passes of a Gaussian filter, the image is in effect being defocused or "smeared," so that groups of particles behave more as a deforming cloud with smooth gradients.

Despite the continued prevalence of linear optical flow methods, some authors have made improvements by following a nonlinear formulation similar to that of Brox et al., and by using physically meaningful regularization terms in the brightness functionals in hopes of forcing the motion of brightness patterns to adhere to trajectories that could be realistically expected in a fluid flow. Corpetti et al. ([80], [81]) noted that the Euler-Lagrange equations given by

$$\int_{\Omega} |\nabla u|^2 + |\nabla v|^2 \tag{6.21}$$

are the same as those associated with a first-order div-curl regularizer

$$\int_{\Omega} \mathrm{div}^2 \boldsymbol{u} + \mathrm{curl}^2 \boldsymbol{u}, \tag{6.22}$$

leading to a set of constraint equations that estimate the diverge and vorticity of the flow fields being solved for. Ruhnau et al. ([82]) employed the full incompressible Navier-Stokes equations in their regularization term, giving the cost functional

$$E = \frac{1}{2}\int_{\Omega} \{(\boldsymbol{u} \cdot \nabla I + I_t)^2 + \lambda(\boldsymbol{\omega} - \boldsymbol{\omega}_T)^2 + \kappa |\nabla \boldsymbol{\omega}|^2\} d\boldsymbol{x}, \tag{6.23}$$

such that $\nabla \cdot \boldsymbol{u} = 0$ and $\nabla \times \boldsymbol{u} = \boldsymbol{\omega}$, with $\boldsymbol{\omega}_T$ denoting the vorticity solution from the previous image pair (set to zero on the first pair).

This work focuses on applying the method of Brox et al. to PIV images, with changes to the regularization terms in the constraint equations left as future work. It should be noted that initial efforts with the Horn-Schunck method on PIV images produced poor results similar to those shown for the RBCs in Figure 6-23, and thus the method was not considered further. To evaluate the performance of nonlinear optical flow on PIV images, several particle image sets were obtained from the PIV Challenge web site (http://www.pivchallenge.org/), the internet home of a contest held open to members of industry and academia in 2003, 2005, and 2008 for the purpose of stimulating creative efforts toward developing improved PIV algorithms [85], [86], [87].

6.6.1 A Synthetic Rankine Vortex Image Pair

The first pair of PIV images evaluated using nonlinear optical flow was a synthetic random Gaussian particle field prescribed with the motion of a Rankine vortex, which is defined as:

$$u_r = \frac{1}{2}\alpha r \tag{6.24}$$

$$u_\theta = \frac{\Gamma}{r}\left(1 - \exp\left(-\frac{\alpha}{4\nu}r^2\right)\right) \tag{6.25}$$

$$u_z = \alpha z \tag{6.26}$$

with $\Gamma = 160$, $\alpha = 6$, and $\nu = 400\ \text{pixel}^2$. The plane of view was inclined 20° to the vortex axis to give it a slightly elliptic shape. Figure 6-28 illustrates the optical flow solution for the synthetic vortex (D) compared to the solution given by the commercial PIV software package DaVis 7.2, a product of LaVision, Inc., using 16×16 pixel interrogation windows overlapped 50%. It can be seen that the results are qualitatively comparable, with the Brox result giving a denser velocity field (though the vectors are plotted every 8 pixels for visual clarity and better comparison with the DaVis result; contours of u-velocity are plotted along with the vectors). The Brox result also features fewer gaps in the vector field, and better resolves information near the vortex core, which is approximately outlined with a circle of equal radius and centroid on each image. (Note that the Davis y-axis is reversed, as the software is designed to read CCD digital camera images directly, which are exported in this y-reversed format.) Thus, optical flow results produced by the established algorithm are encouraging on this image pair.

6.6.2 Real vortex

The next PIV image pair evaluated was again of a vortex, but this time it was acquired in a real experimental setup rather than generated synthetically. Figure 6-29 (A) and (B) show the original image pair, with (C) and (D) showing the results given by DaVis and nonlinear optical flow, respectively (the optical flow figure has been zoomed in 2 × on the region surrounding the vortex for visual clarity). Unlike with the synthetic image pair, the DaVis results are clearly of much higher quality than the optical flow results in this case. This is thought to be caused by two fundamental differences exist between this image pair and the synthetic pair. First, the particle field is not uniformly populated. The vortex core has far fewer particles than the rest of the image, giving a strong intensity gradient radially toward the vortex center. Second, the overall intensity varies considerably between the first image and the second, probably due to differences in the two laser sheets used during image acquisition. Both the image brightness and the image brightness gradient are conserved quantities in the Brox formulation, which could be a leading contributor to this problem. It should be noted that the brightness and smoothness weighting parameters were left at the values used for other cases in this chapter, and could well have been maladapted to the specific requirements of this image pair. Clearly much work remains before the optical flow methods developed herein can be used to evaluate PIV images of this nature.

6.6.3 Synthetic shear flow

Figure 6-30 shows the results of calculating the vector field on a synthetic shear flow image with a velocity gradient given by $u = A \tanh \beta y$. This case was considered to be of interest because the algorithms that were entered into the PIV Challenge that year, including one produced by LaVision, had such variation in their level of success with it; many of the algorithms produced large errors, with solution vectors nearly opposing their intended directions. In this work, however, DaVis was found to produce what appeared to be a reasonable solution, with the solution of the nonlinear optical flow method matching it closely, albeit more densely. Due to the high particle density and small image size, the interrogation regions were set to be 8×8 with 50% overlap in the DaVis software for this case.

Of particular note here is the presence of an overall spatial intensity gradient in this image pair. Yet, the Brox method was able to perform well qualitatively in this case, where it could not in the case of the real vortex images. This leads to the conclusion that the overall image intensity change has a large negative impact on the abilities of optical flow to accurately handle such images, or perhaps that radial intensity gradients are problematic for elliptic equations such as those of the optical flow algorithm.

6.7 Conclusions

Both the linear (Horn and Schunck) and nonlinear (Brox et al.) formulations of the optical flow method were tested extensively on a set of synthetically generated images possessing a variety of characteristics with regard to geometry, intensity gradients, and kinematics. In each case, the nonlinear formulation of Brox et al. proved to be the superior method, consistently yielding small errors most of the time even when errors in the linear formulation became quite large. Applied to the video sequence of the swimming American eel, the nonlinear method proved its superiority further by successfully shifting a set of pixels from one frame to the next, with little discernable difference between the resulting configuration of the featured object and that which was expected.

Overall, the nonlinear optical flow method of Brox et al. appears to be a reliable means of providing boundary conditions to imaged objects without the need to populate them with a set of Lagrangian points along their interfacial curves. It also holds promise as a method for tracking particles in video files such as that of the experimental channel populated with red blood cells, as well as in image pairs acquired for PIV analysis, particularly if it can be made more robust with physically meaningful regularization terms in the vein of Corpetti et al. or Ruhnau et al.

Because nonlinear optical flow possesses the ability to solve for the motion of brightness patterns occurring in such disparate scales—i.e. large objects such as the eel, as well as small objects such as red blood cells and PIV particles—there exists the possibility for quantifying motion at several different scales of interest simultaneously through imagery, e.g. particles in a flow interacting with moving boundaries. With further refinement, it is hoped that optical flow can be brought to a state where obtaining the trajectories of particles, such as blood cells, along with the surface motion of large adjacent objects, such as vascular or cardiac tissues, can become a reality.

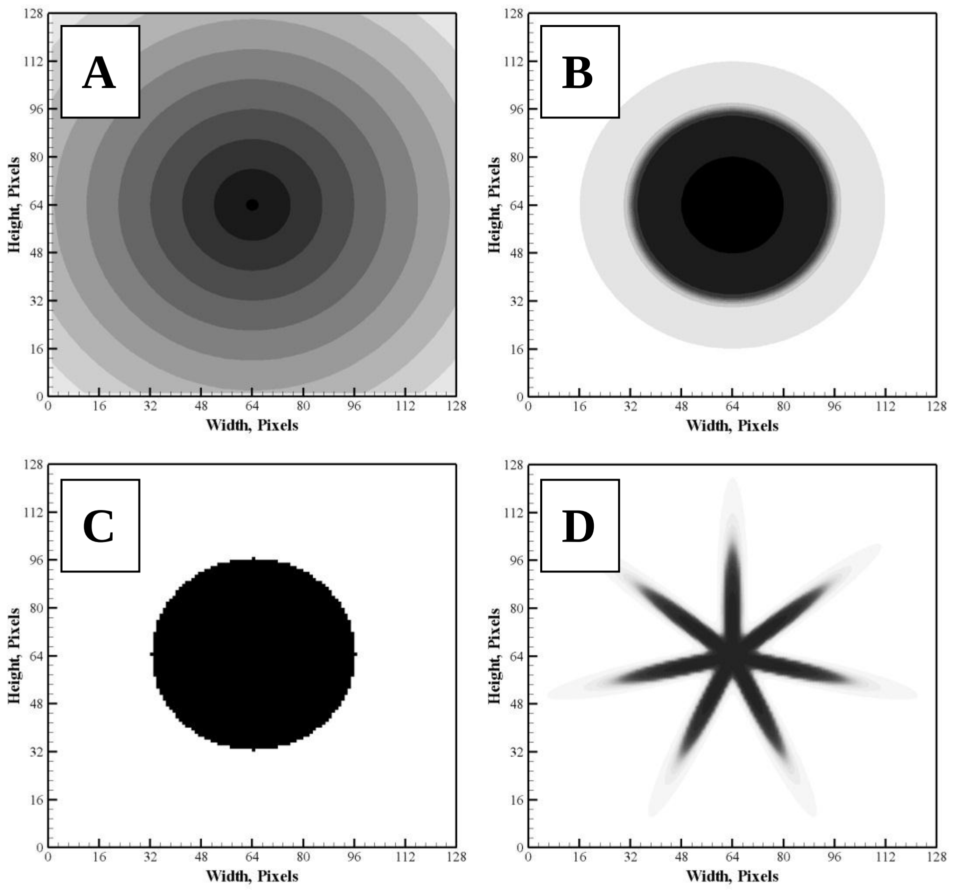


Figure 6-1. 128 × 128 images generated for optical flow testing: (A) circular level set field; (B) circular Heaviside field; (C) binary circle; (D) star-shaped Heaviside field.

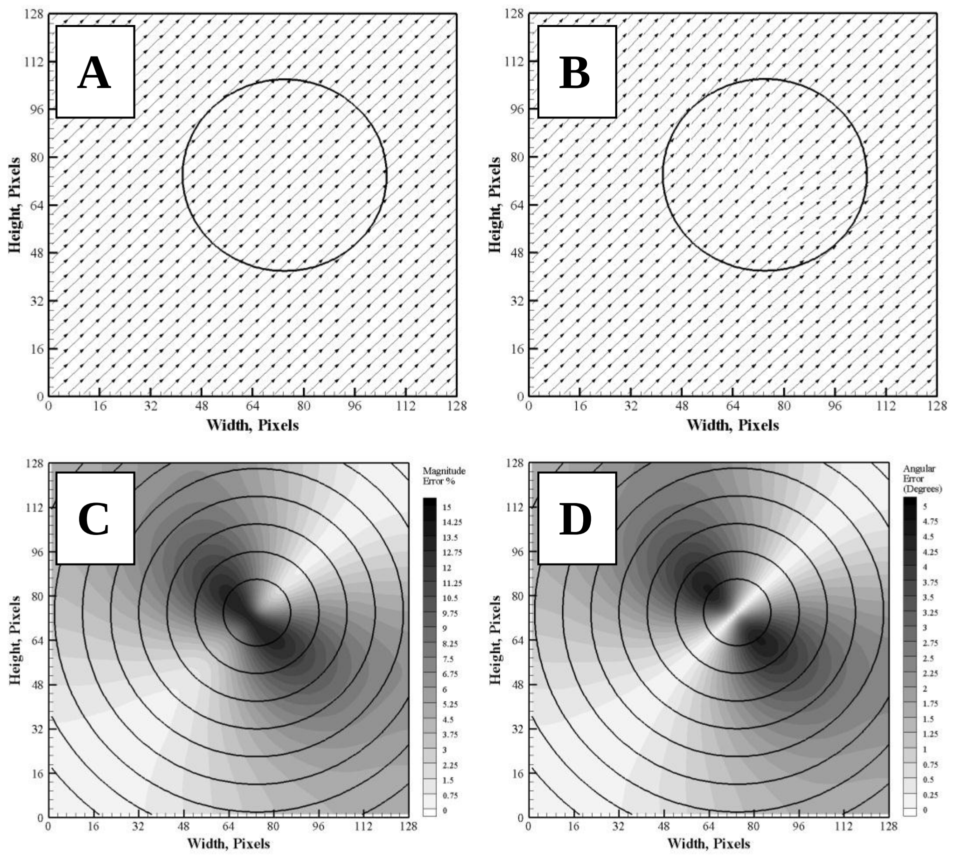


Figure 6-2. Horn and Schunck solution of the translating circular level set field, $U = V = 5.0\ pixels/frame$: (A) exact vector field, with the zero level set shown; (B) Horn and Schunck vector field, with the zero level set shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

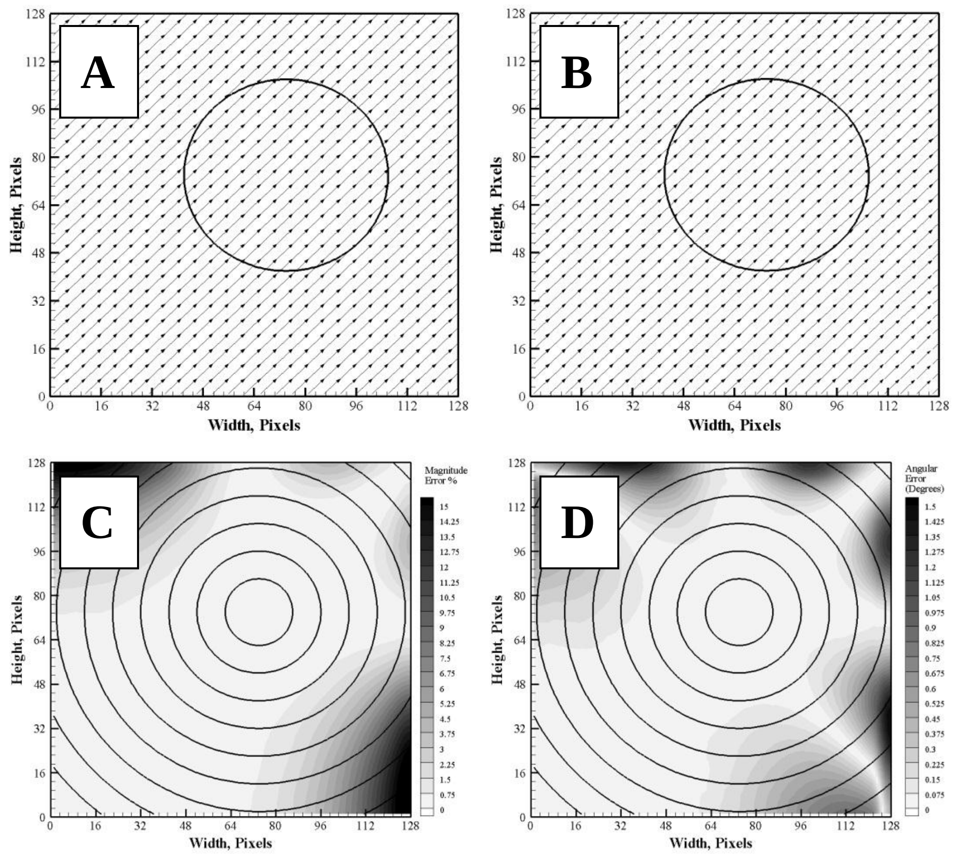


Figure 6-3. Brox et al. solution of the translating circular level set field, $U = V = 5.0$ $pixels/frame$: (A) exact vector field, with the zero level set shown; (B) Brox et al. vector field, with the zero level set shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

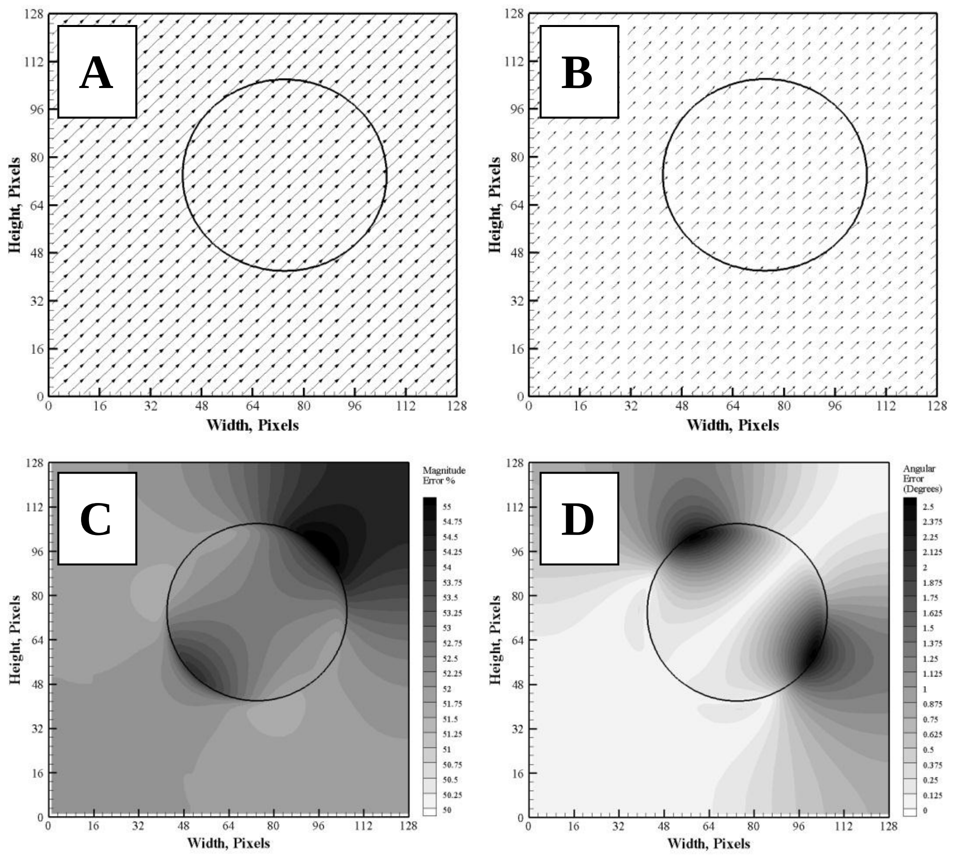


Figure 6-4. Horn and Schunck solution of the translating circular Heaviside field, $U = V = 5.0\ pixels/frame$: (A) exact vector field, with the zero level set shown; (B) Horn and Schunck vector field, with the zero level set shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

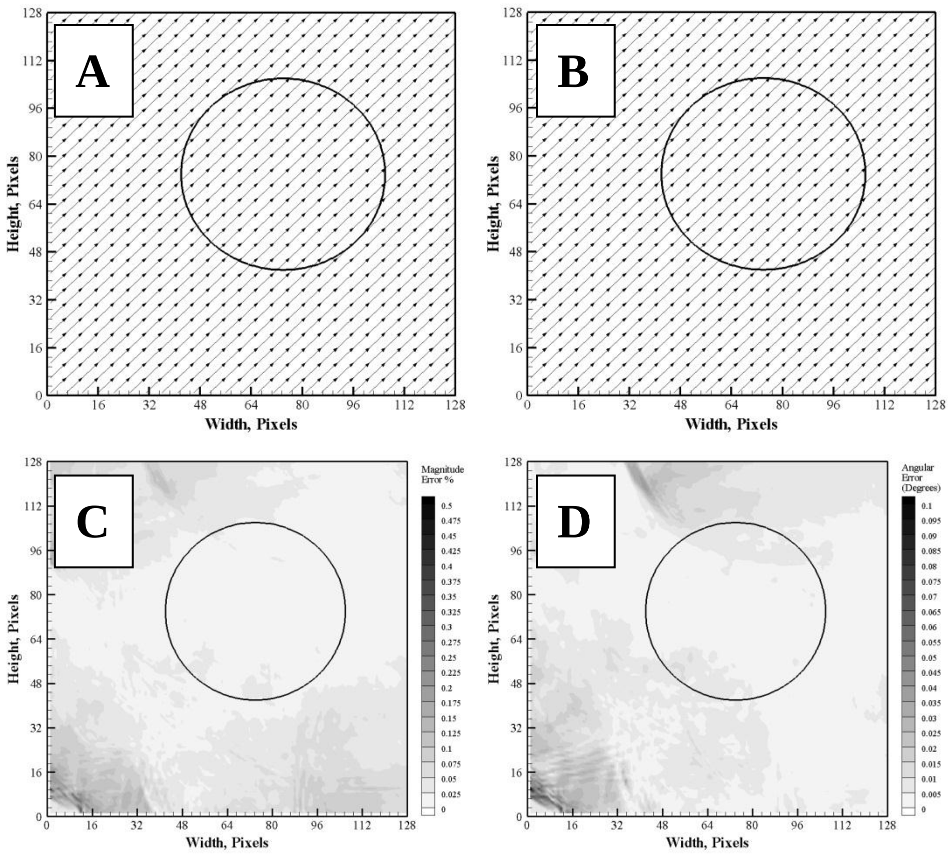


Figure 6-5. Brox et al. solution of the translating circular Heaviside field, $U = V = 5.0$ $pixels/frame$: (A) exact vector field, with the zero level set shown; (B) Brox et al. vector field, with the zero level set shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

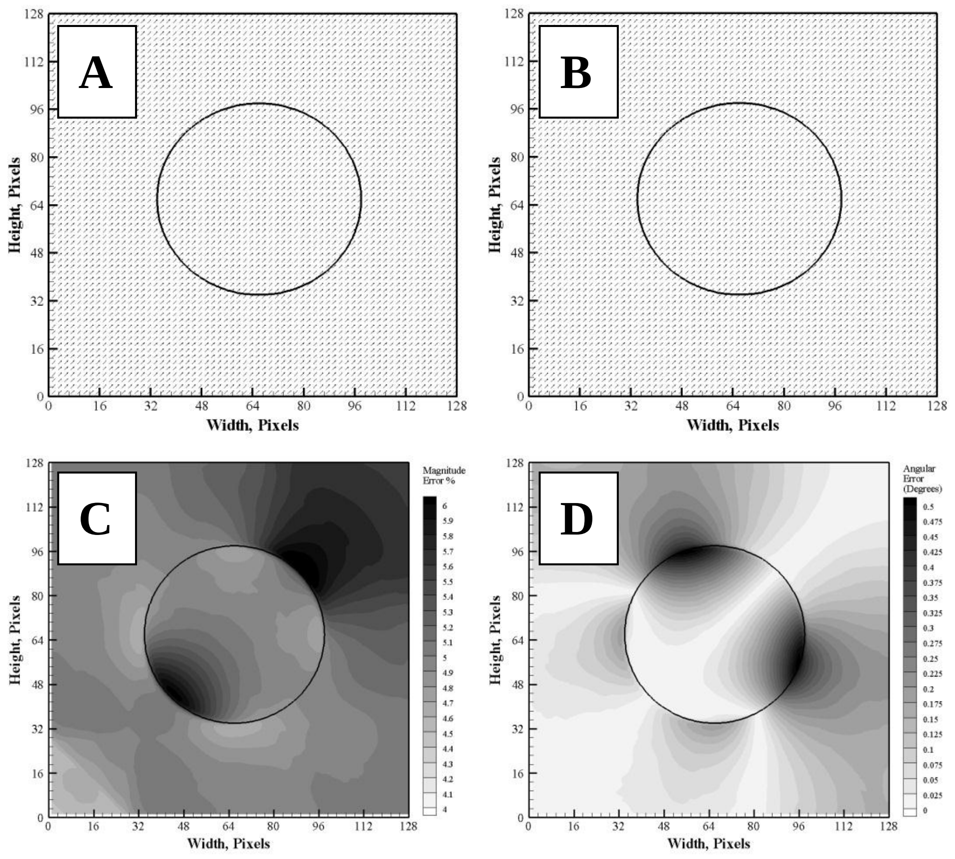


Figure 6-6. Horn and Schunck solution of the translating circular Heaviside field, $U = V = 1.0\ pixel/frame$: (A) exact vector field, with the zero level set shown; (B) Brox et al. vector field, with the zero level set shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

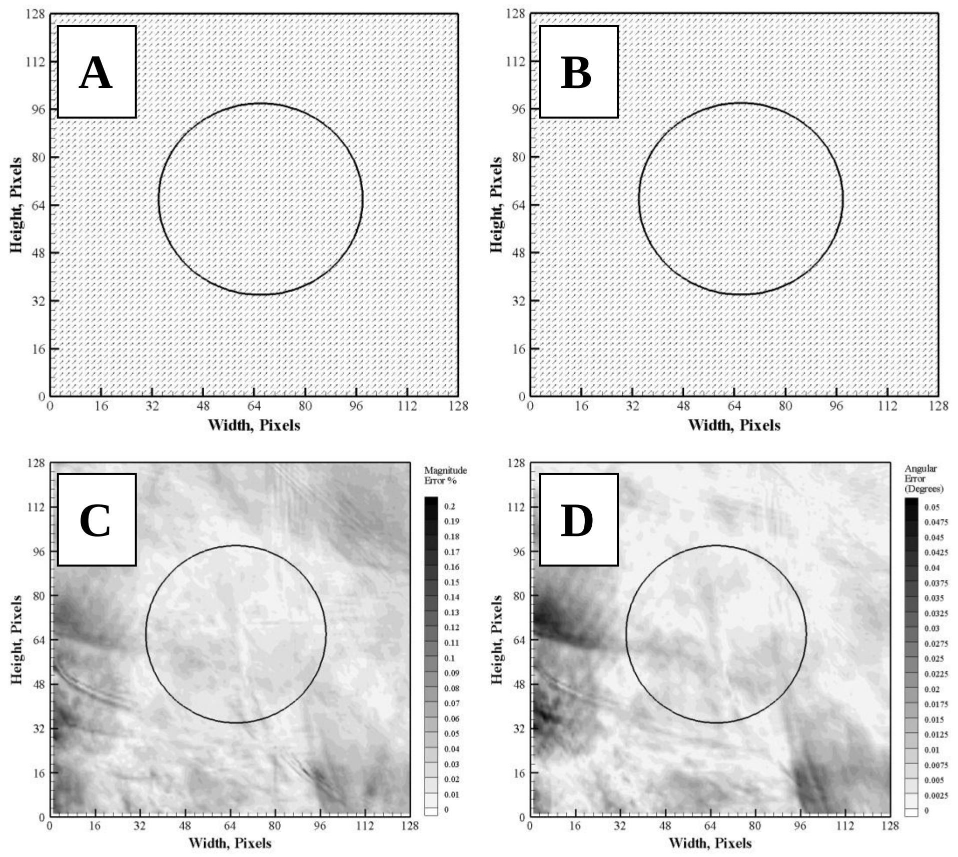


Figure 6-7. Brox et al. solution of the translating circular Heaviside field, $U = V = 1.0$ $pixel/frame$: (A) exact vector field, with the zero level set shown; (B) Brox et al. vector field, with the zero level set shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

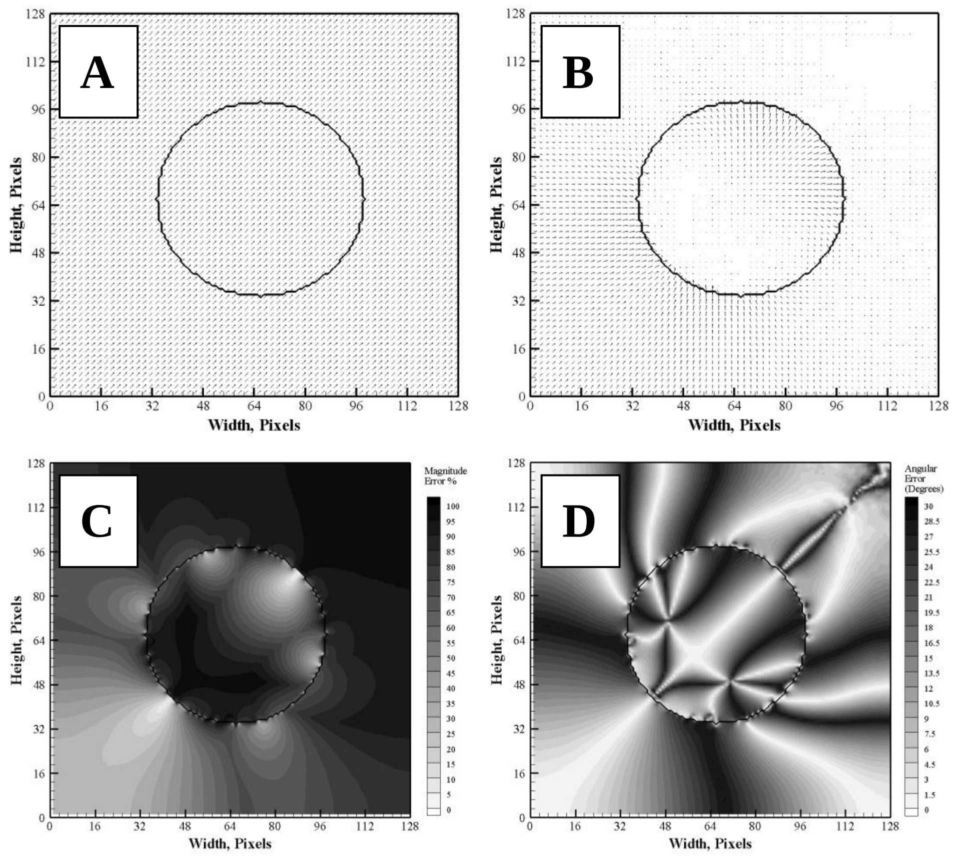


Figure 6-8. Horn and Schunck solution of the translating binary circle, $= V = 1.0$ $pixel/frame$: (A) exact vector field, with the circle boundary; (B) Horn and Schunck vector field, with the circle boundary shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

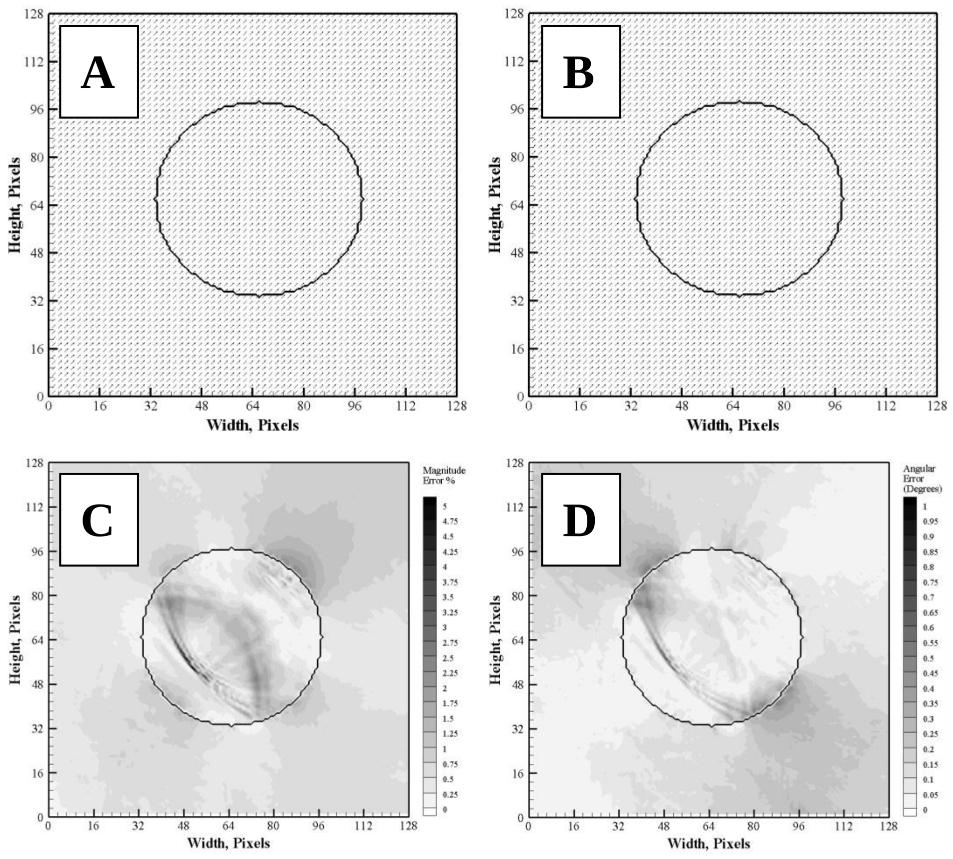


Figure 6-9. Brox et al. solution of the translating binary circle, $U = V = 1.0\ pixel/frame$: (A) exact vector field, with the circle boundary; (B) Brox et al. vector field, with the circle boundary shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

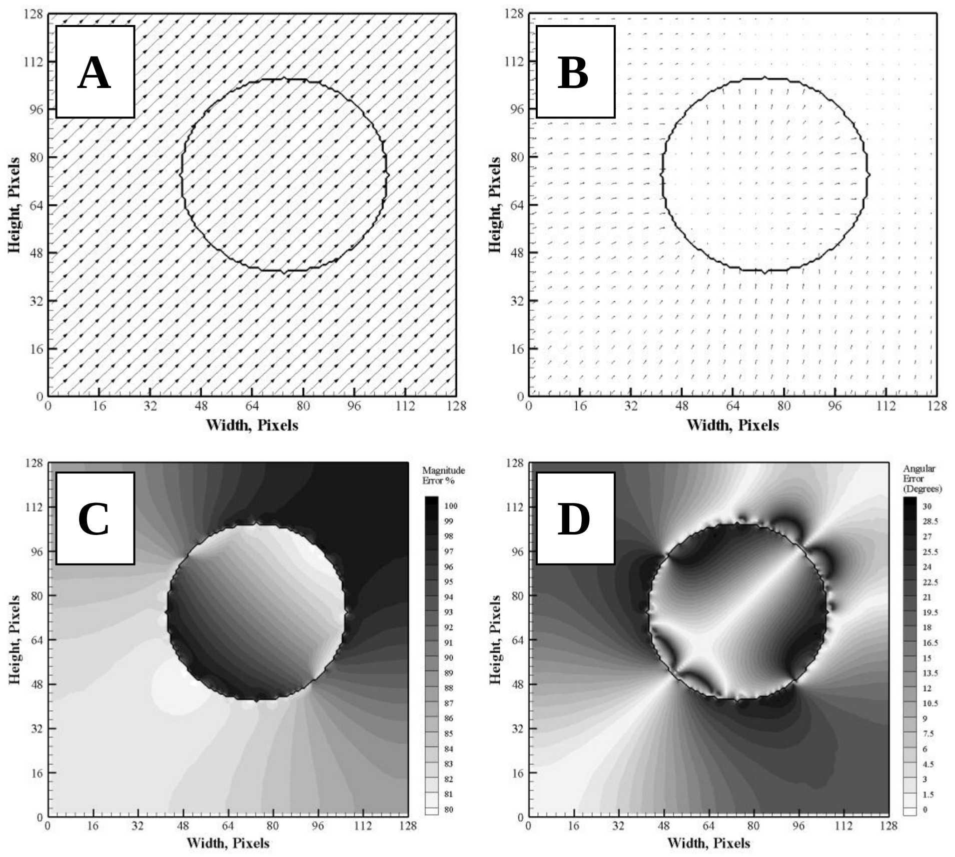


Figure 6-10. Horn and Schunck solution of the translating binary circle, $= V = 5.0$ $pixels/frame$: (A) exact vector field, with the circle boundary; (B) Horn and Schunck vector field, with the circle boundary shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

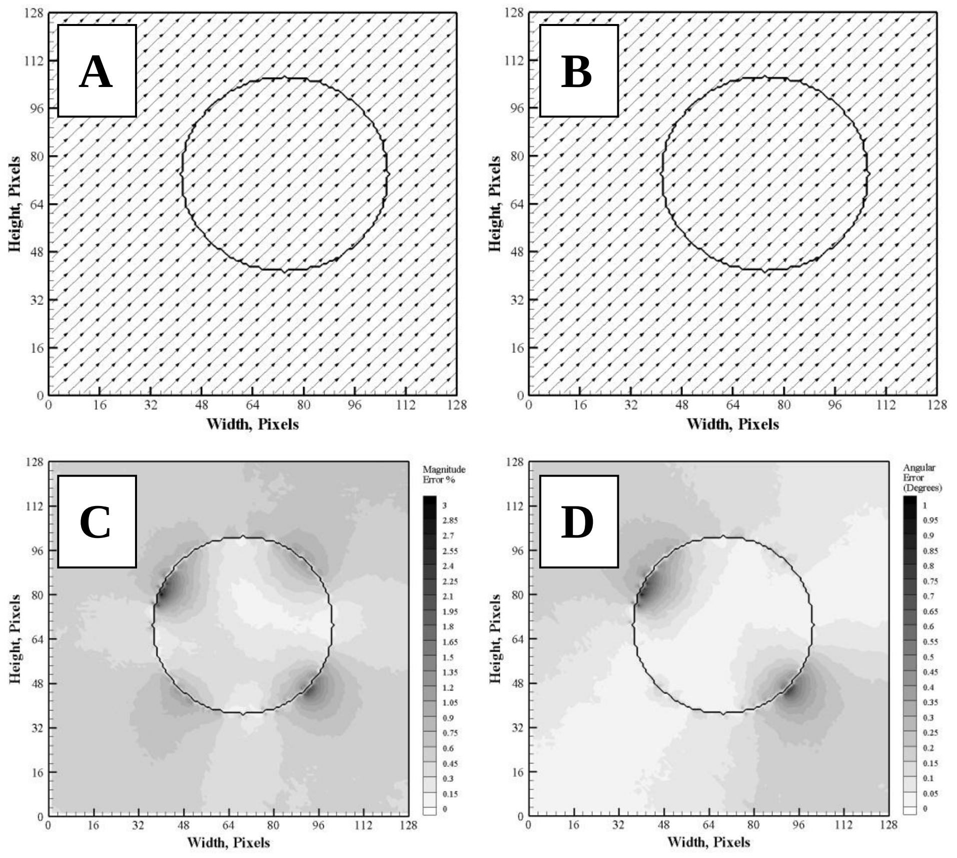


Figure 6-11. Brox et al. solution of the translating binary circle, $U = V = 5.0\ pixels/frame$: (A) exact vector field, with the circle boundary; (B) Brox et al. vector field, with the circle boundary shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

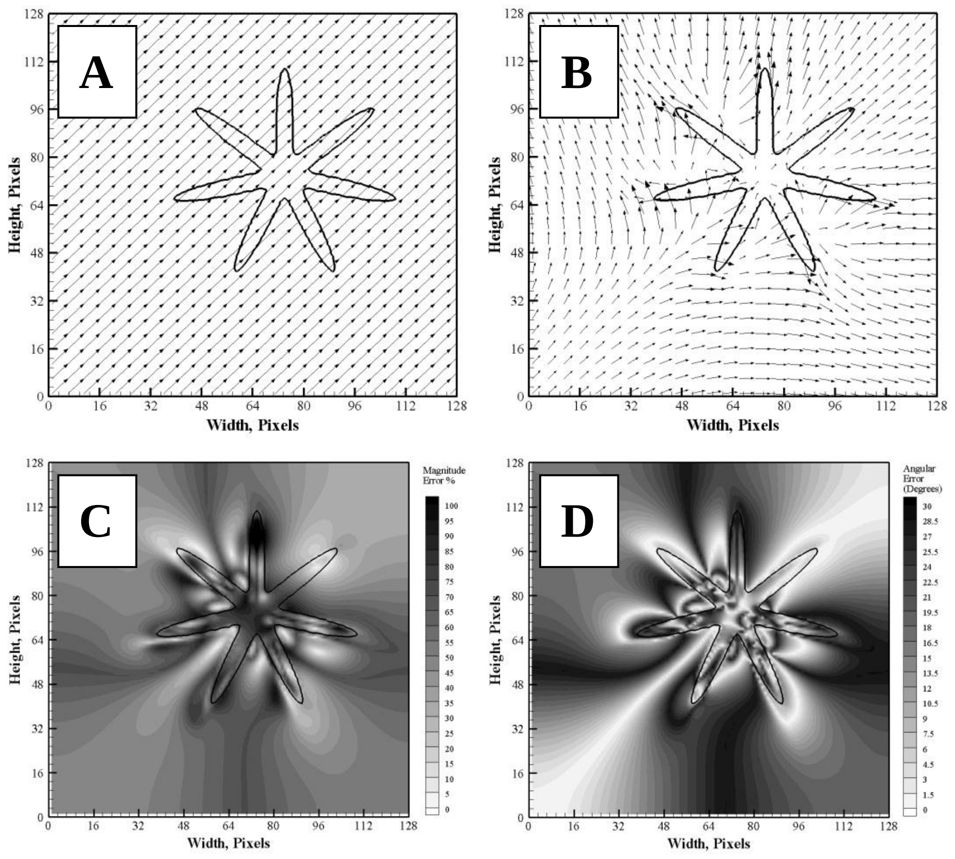


Figure 6-12. Horn and Schunck solution of the translating Heaviside star, $U = V = 5.0$ $pixels/frame$: (A) exact vector field, with the mean Heaviside contour shown; (B) Horn and Schunck vector field, with the mean Heaviside contour shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

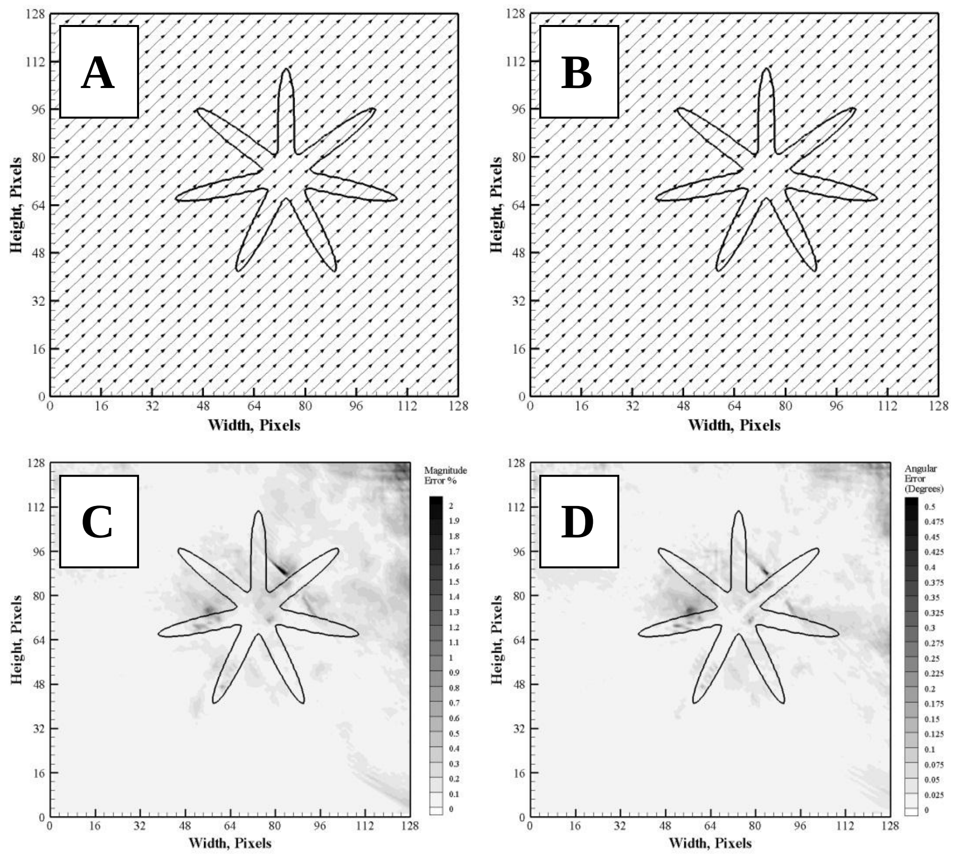


Figure 6-13. Brox et al. solution of the translating Heaviside star, $U = V = 5.0\ pixels/frame$: (A) exact vector field, with the mean Heaviside contour shown; (B) Brox et al. vector field, with the mean Heaviside contour shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

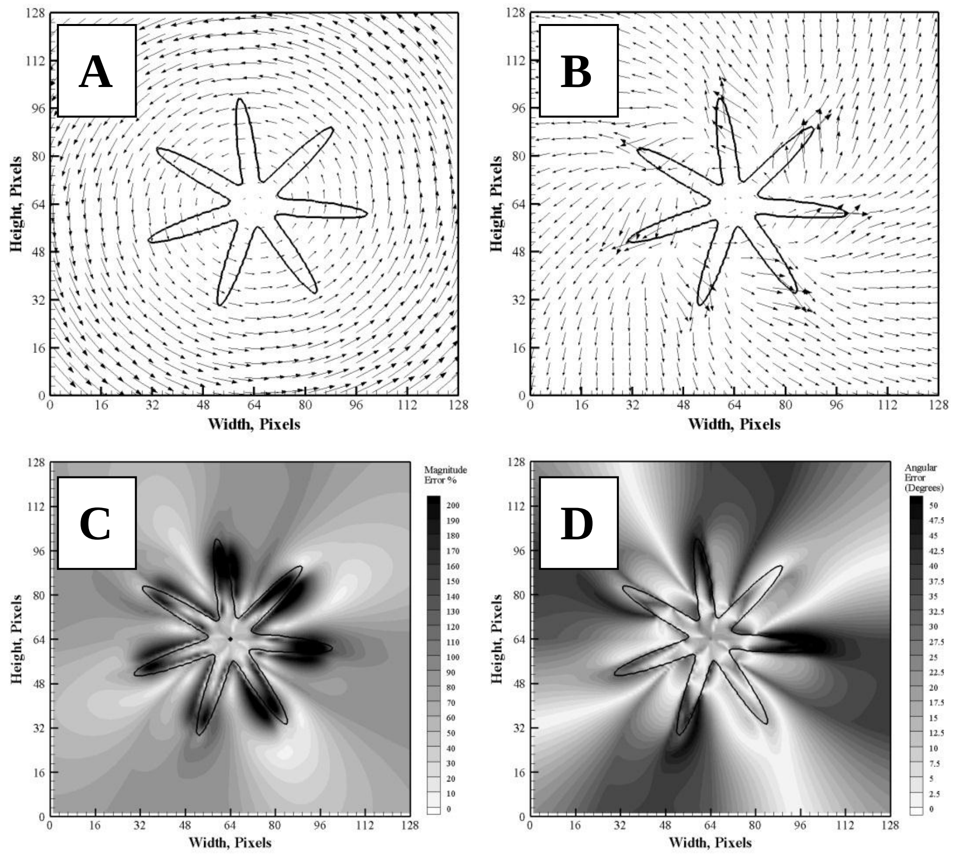


Figure 6-14. Horn and Schunck solution of the rotating Heaviside star, $\dot{\theta} = \pi/24\ rad/frame$: (A) exact vector field, with the mean Heaviside contour shown; (B) Horn and Schunck vector field, with the mean Heaviside contour shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

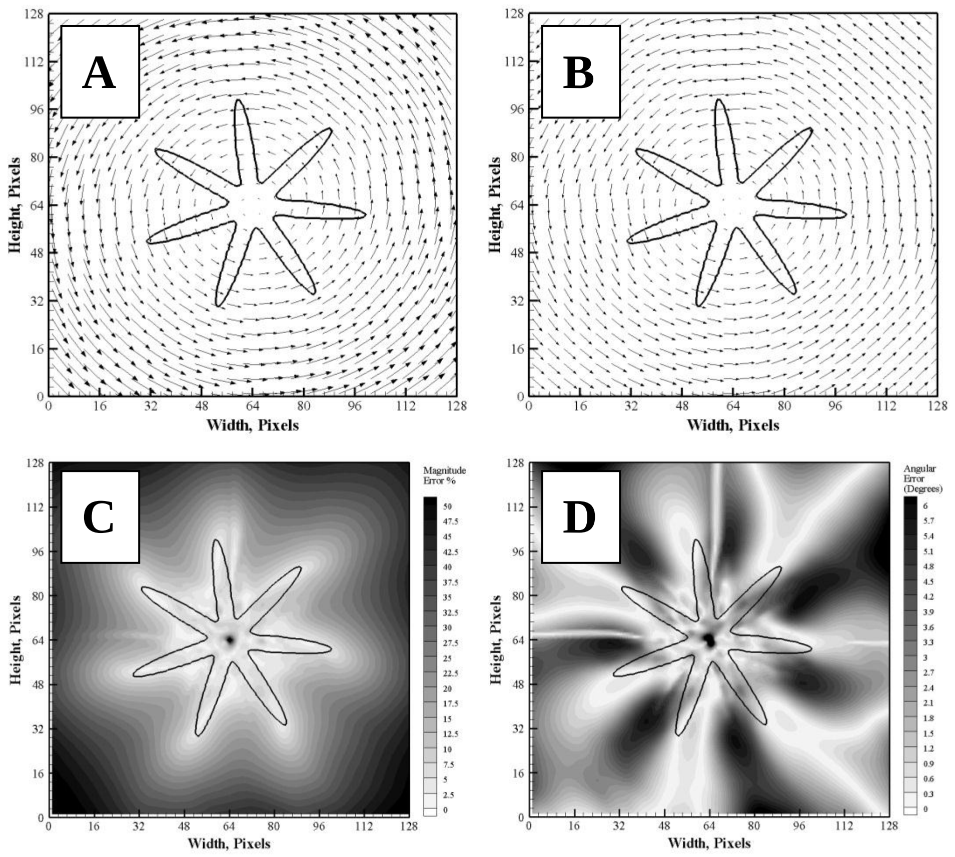


Figure 6-15. Brox et al. solution of the rotating Heaviside star, $\dot{\theta} = \pi/24\ rad/frame$: (A) exact vector field, with the mean Heaviside contour shown; (B) Brox et al. vector field, with the mean Heaviside contour shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

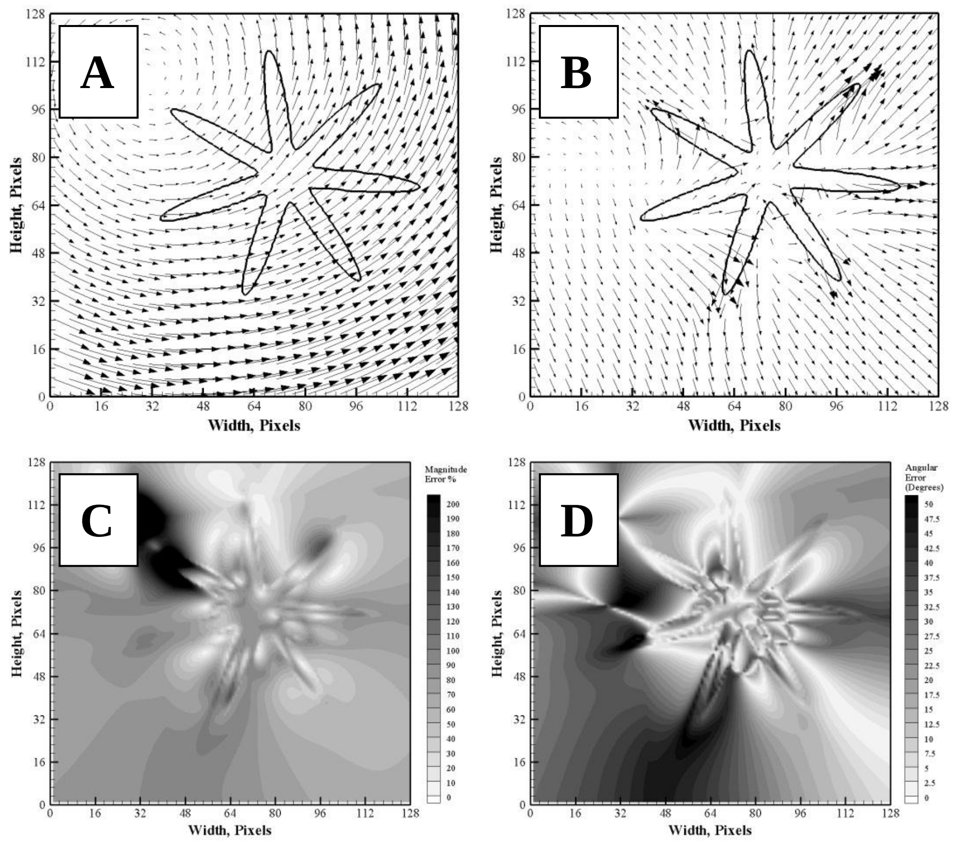


Figure 6-16. Horn and Schunck solution of the translating and rotating Heaviside star, $U = V = 5.0\ pixels/frame,\ \dot{\theta} = \pi/24\ rad/frame$: (A) exact vector field, with the mean Heaviside contour shown; (B) Horn and Schunck vector field, with the mean Heaviside contour shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

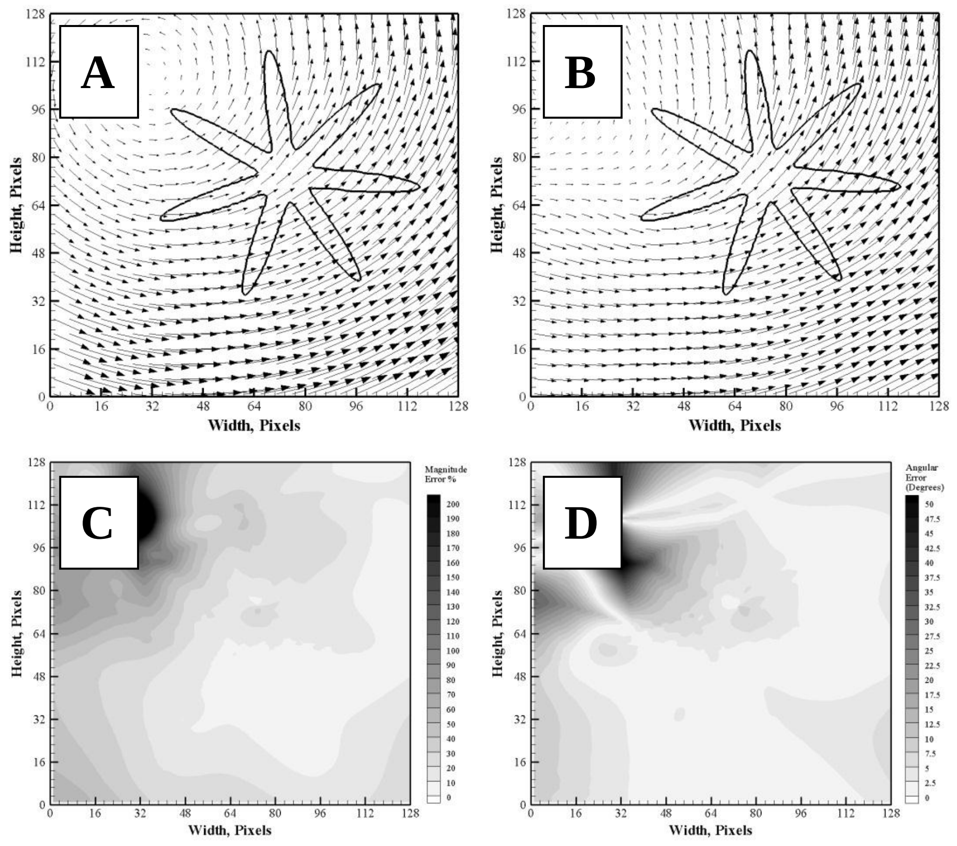


Figure 6-17. Brox et al. solution of the translating and rotating Heaviside star, $U = V = 5.0\ pixels/frame$, $\dot{\theta} = \pi/24\ rad/frame$: (A) exact vector field, with the mean Heaviside contour shown; (B) Brox et al. vector field, with the mean Heaviside contour shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

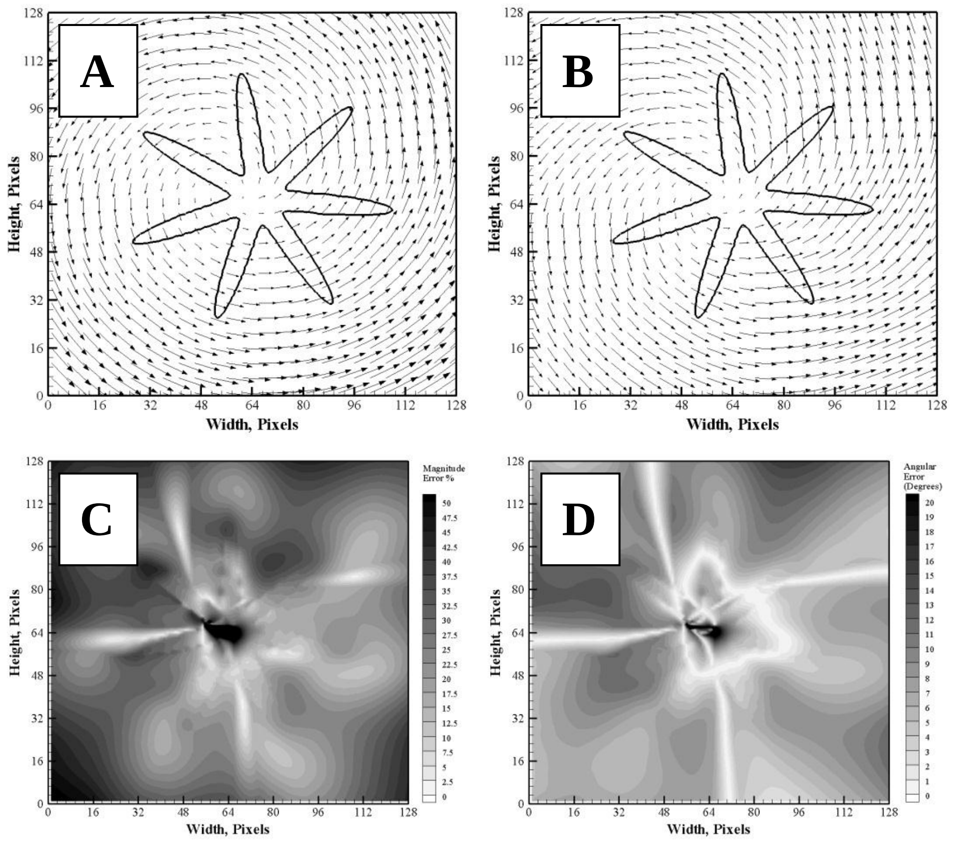


Figure 6-18. Brox et al. solution of the translating and rotating Heaviside star, $U = V = 1.0\ pixel/frame$, $\dot{\theta} = \pi/24\ rad/frame$: (A) exact vector field, with the mean Heaviside contour shown; (B) Brox et al. vector field, with the mean Heaviside contour shown; (C) map of velocity magnitude error %; (D) map of velocity angular error in degrees.

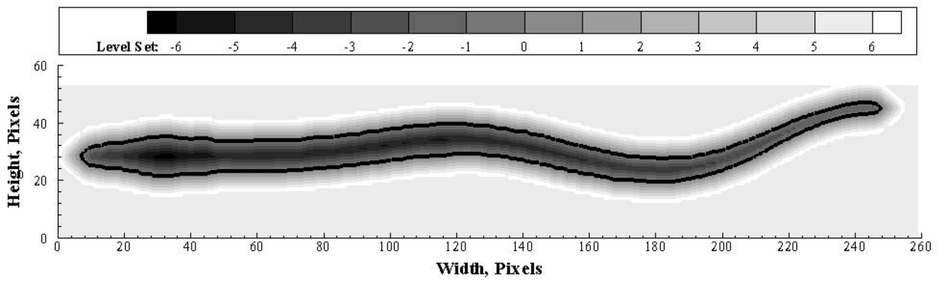


Figure 6-19. Eel narrow band level set field, with the zero level contour (the surface) shown as a black line.

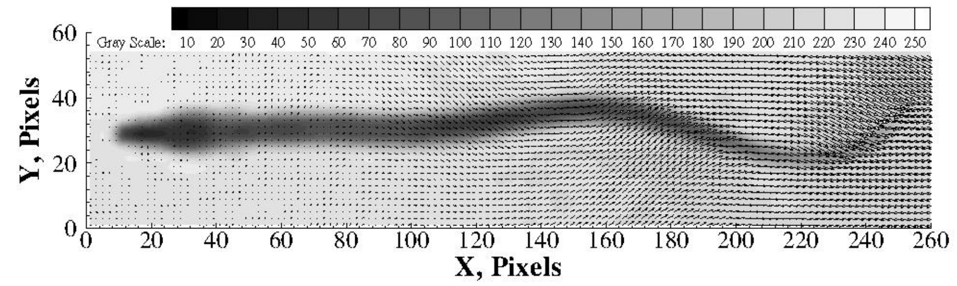


Figure 6-20. Eel optical flow vector field for one image frame pair, solved using the nonlinear method of Brox et al.

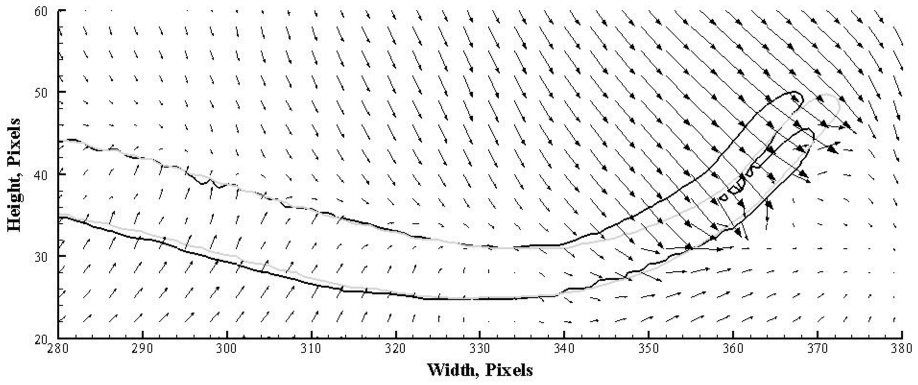


Figure 6-21. Eel tail vectors for one image pair using the optical flow method of Horn and Schunck. The advected zero-level is superimposed onto the target zero-level, showing poor matching between the two.

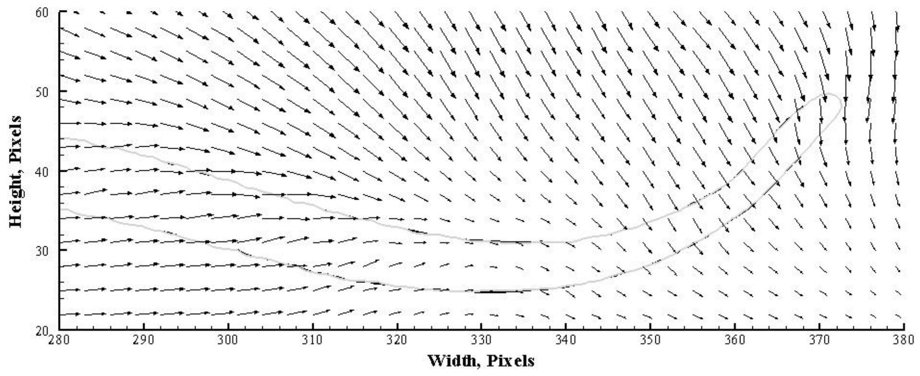


Figure 6-22. Eel tail vectors for one image pair using the optical flow method of Brox et al. The advected zero-level is superimposed onto the target zero-level, showing little difference between the two.

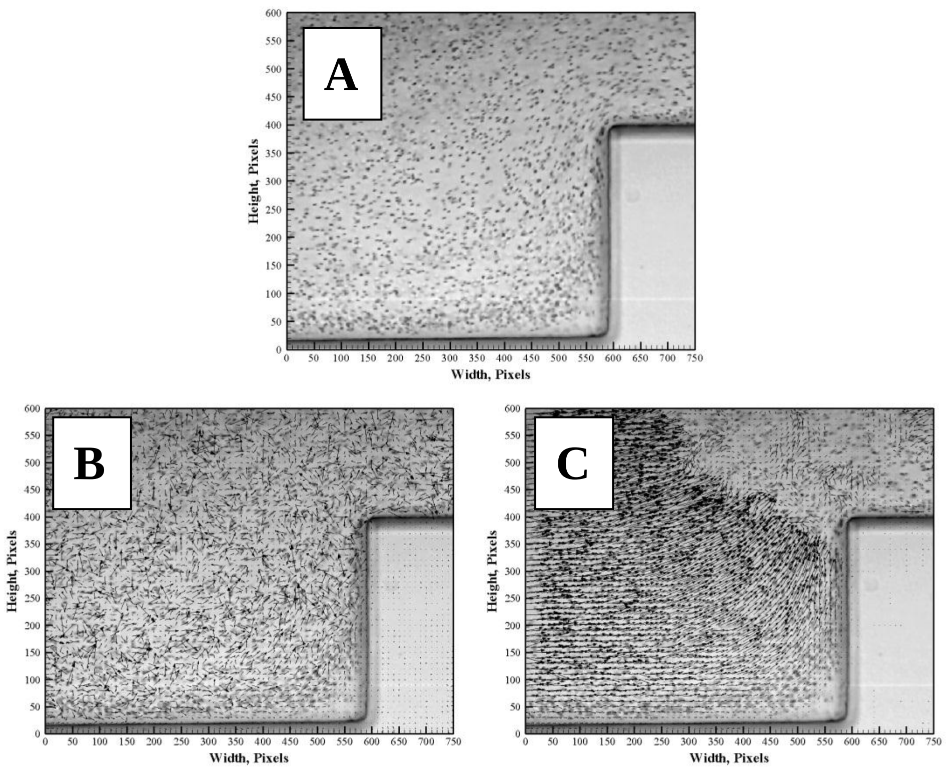


Figure 6-23. Application to particle tracking. Video image of channel with red blood cells flowing through it (A). Horn and Schunck result (B) versus Brox result (C).

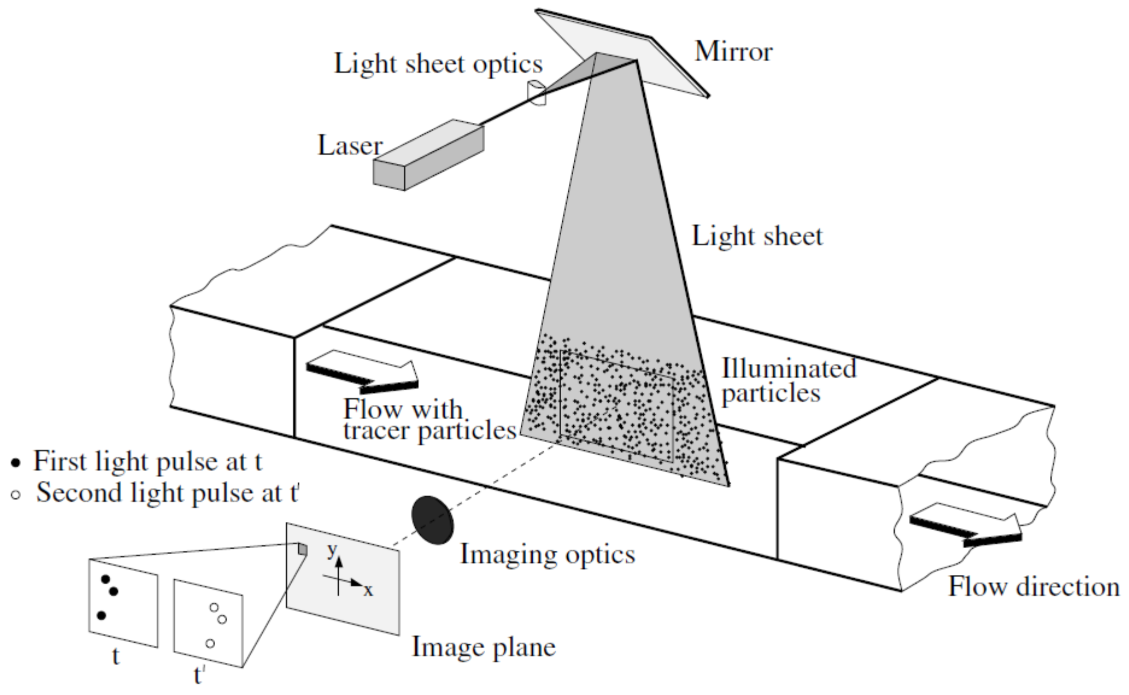


Figure 6-24. Image acquisition in a PIV experimental setup (figure taken directly from [79]).

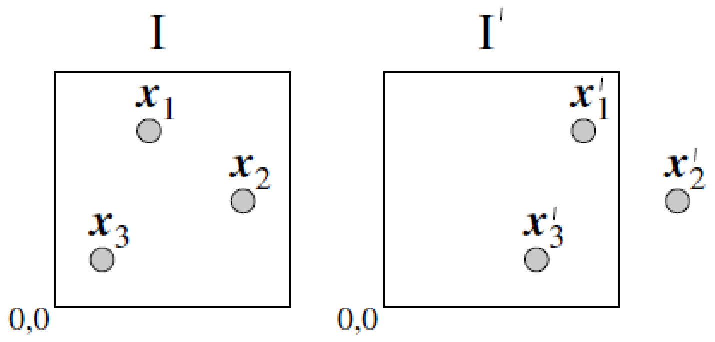


Figure 6-25. Changes in the brightness patterns between two interrogation windows (one on the first image frame *I* and one on the second image frame *I′*) allow for particle motion to be evaluated (figure taken directly from [79]).

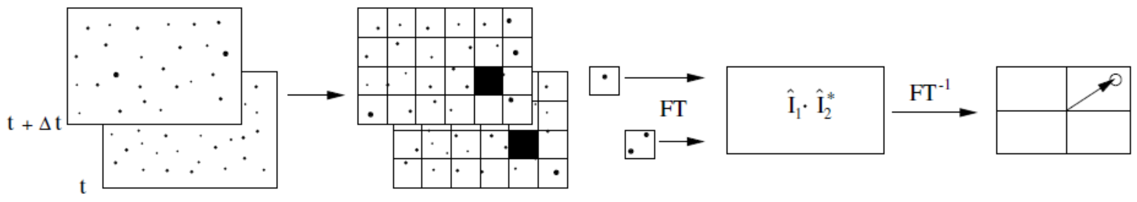


Figure 6-26. The evaluation of PIV algorithms is typically performed by cross-correlation of brightness patterns within interrogation windows. The size of the interrogation windows is dictated by particle density, which in turn dictates the density of the resultant vector field (figure taken directly from [79]).

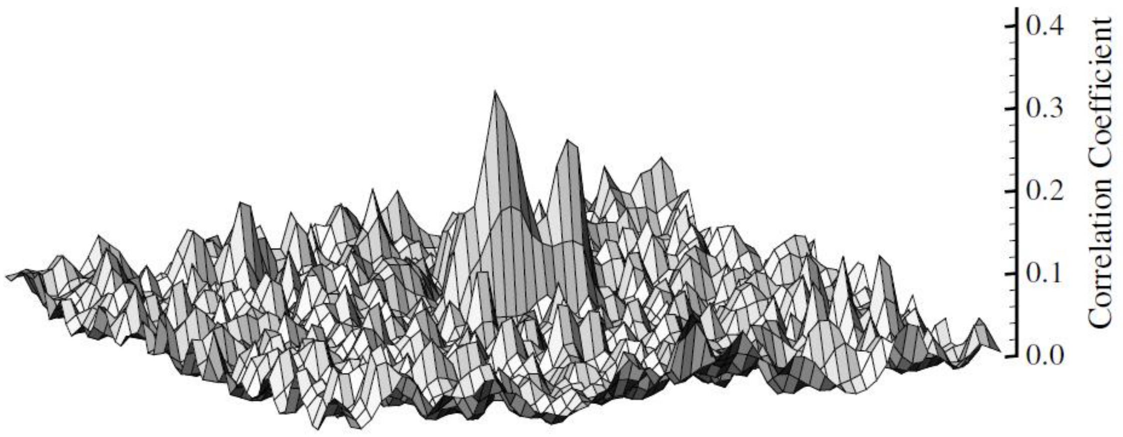


Figure 6-27. The strongest correlation peak is generally used to determine a PIV vector; the peak’s direction of displacement from the center of the interrogation window gives the resultant vector’s direction, and the distance it is displaced gives vector magnitude (figure taken directly from [79]).

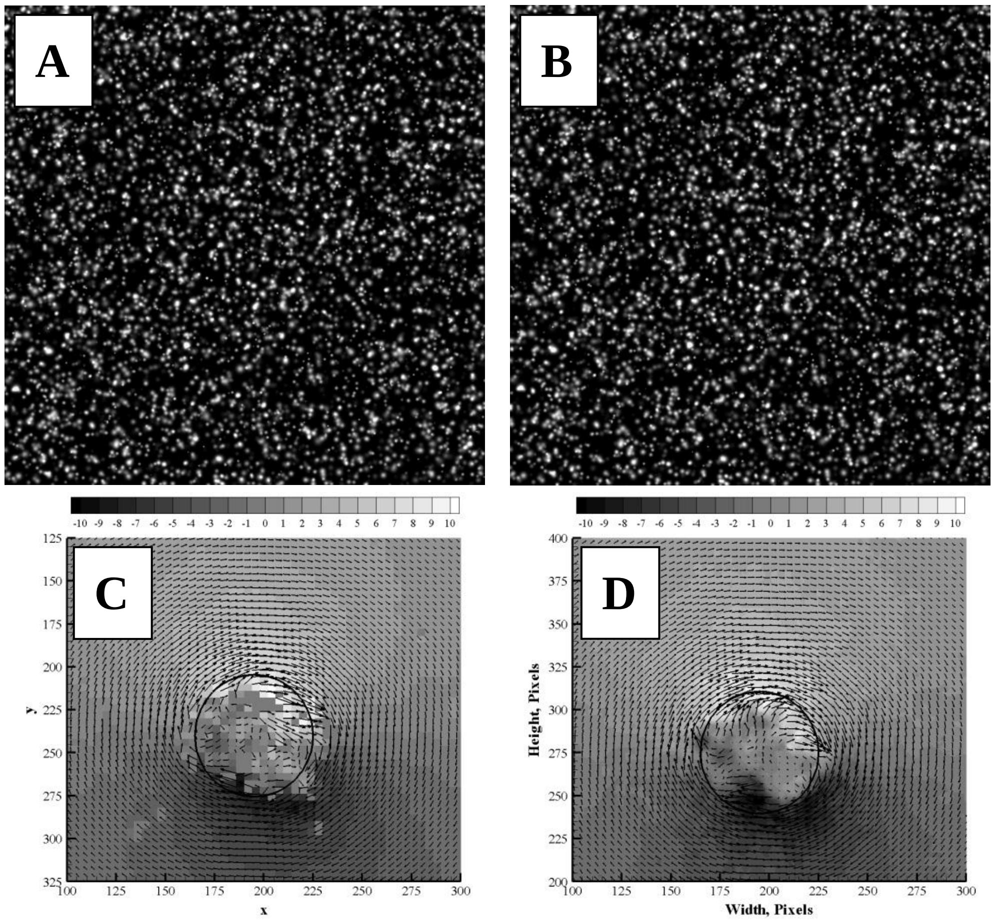


Figure 6-28. Comparison of the Brox et al. optical flow method with the DaVis commercial PIV software on a synthetic Rankine vortex field.

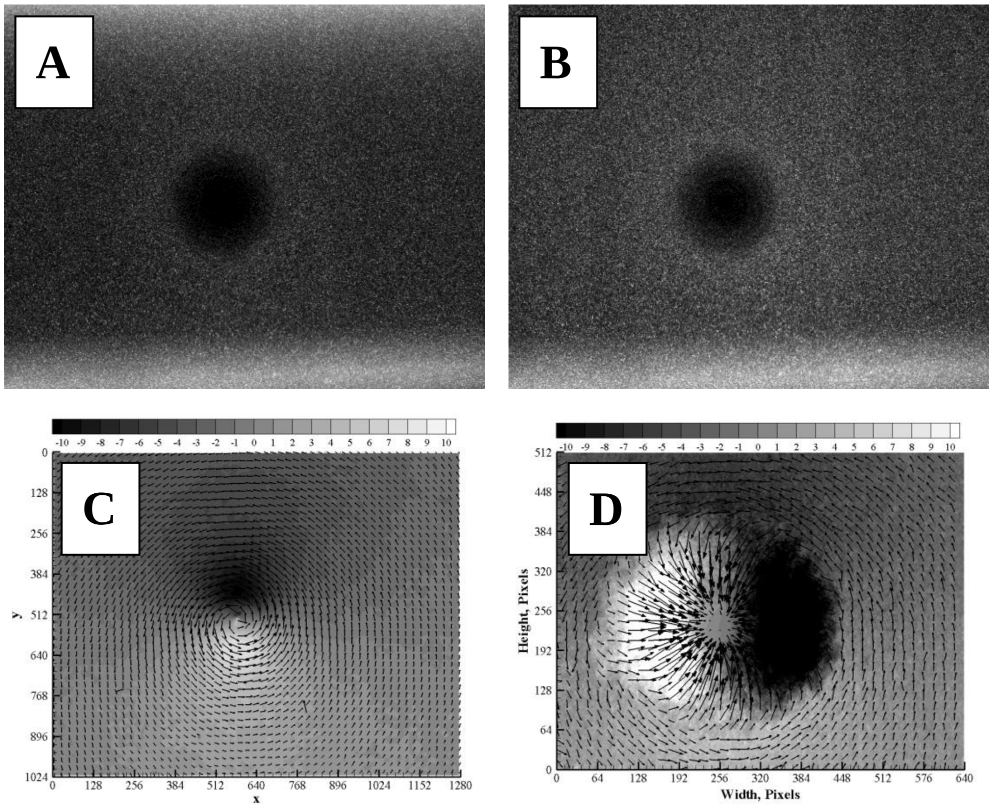


Figure 6-29. Comparison of the Brox et al. optical flow method with the DaVis commercial PIV software on a real image pair of a strong vortical flow.

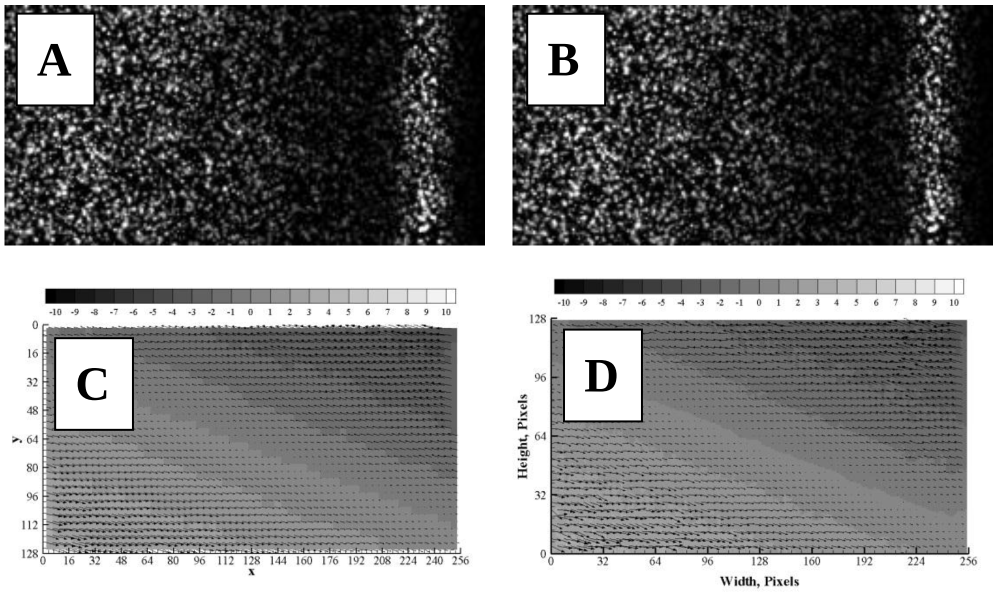


Figure 6-30. Comparison of the Brox et al. optical flow method with the DaVis commercial PIV software on a synthetic image pair of a shear flow.

CHAPTER 7

A TOTALLY EULERIAN APPROACH TO IMAGE-BASED COMPUTATION VIA OPTICAL FLOW AND THE TECHNIQUE OF MORPHING

7.1 Introduction

So far, this body of work has outlined the individual steps taken toward modeling complex objects and their interactions with fluids in a manner that not only preserves fidelity, but adheres to the simple framework of Cartesian grid formulations using the concept of level sets. This chapter presents the culmination of everything accomplished up to now, and represents the crux of this thesis: a demonstration of the complete process of modeling complex moving boundaries in CFD simulations from a completely Eulerian viewpoint without the use of surface meshes or functional approximations.

In the previous chapter, optical flow was discussed as a means for setting boundary conditions on moving imaged objects without the need to construct surfaces by stringing together pairs of Lagrangian marker points. It was also originally hoped that optical flow vectors would provide the necessary information to reconstruct boundary configurations and effect motion between image frames, which are in general much too temporally coarse for CFD modeling (as discussed in Section 7.2). However, this approach didn't end up performing as anticipated. Section 7.3, examines the results of attempting such a strategy and seeks to address its strengths and shortcomings. Section 7.4 introduces the field of image morphing (which became necessary, to overcome the shortcomings of optical flow with respect to its ability to effect boundary motion on its own), and provides a background overview of its fundamental concepts. Section 7.5 outlines the morphing algorithm adopted for use in this work, and subsequent sections revisit the CFD simulation of the swimming American eel discussed in Chapter 3, now modeled entirely within the desired Eulerian framework without surface meshing.

7.2 Translating Information from the Image Domain to the Computational Domain

One of the key issues in developing a purely Eulerian level set-based approach to embedding imaged objects in flow computations is the need to translate information contained in the image (i.e. pixelized) domain onto the computational (adaptively refined flow mesh) domain. This issue assumes two forms:

1. Temporal interpolation: Within the context of this work, there is a need for temporal interpolation in the image sequence in order to obtain intermediate configurations of an object between image frames, so that its variations in position may be rendered with the temporal density required to couple it with a simulated fluid flow environment. For instance, if a fluid flow of maximum velocity $U = 1$ is simulated on a uniform mesh with a grid spacing of $\Delta x = 0.001$, the $CFL\ condition$,

$$CFL = \frac{U\Delta t}{\Delta x} \leq 1, \qquad \textbf{(7.1)}$$

mandates that time stepping be smaller in magnitude than the grid spacing in order to reach a stable solution to the Navier-Stokes equations. Thus, using the same example, two image frames separated by a time step of $\Delta T_{Image} = 1$ would need to be morphed incrementally over 1,000 smaller steps in order for the object motion to match smoothly with the fluid motion.

In practice, the maximum flow velocity is not constant throughout a CFD simulation (nor is the minimum grid spacing, if local mesh refinement is employed), and so time step sizes are continuously being updated by checking whether the CFL condition is being satisfied everywhere on the flow mesh. For this reason, it is necessary to keep track of the *running time* in the CFD solution process, and correlate it to the frame rate in the image sequence being used to construct the moving boundaries embedded in the fluid mesh.

To elucidate this point, let ΔT_{Image} represent a local image time scale, which has an initial time of $T_{Image,o} = 0$ for the source frame and a final time of $T_{Image,f} = 1$ for the target frame of an ordered pair in a sequence. Let ΔT_{flow} denote a corresponding flow time scale, having a start time t_{f1} during which an object's location is given by the source frame, and an end time t_{f2} during which an object's location matches that of the target frame. Any intermediate time t_f^* between t_{f1} and t_{f2} can then be correlated to a fraction of ΔT_{Image} (Figure 7-8), and this fraction can be used to determine the morphing required to correctly update the object's position.

2. Spatial interpolation: The object to be modeled must be spatially interpolated from the coarse image domain consisting of pixels to a fine flow domain consisting of grid cells that may be locally refined to capture flow phenomena of interest. Thus, a key process in generating purely Eulerian CFD models involves mapping the level set fields obtained by image analysis onto the flow domain.

This mapping is performed in several steps. First, the image pixel locations are converted to real (x, y) pairs (mapping the image domain $\Omega \longmapsto \mathbb{R}^2$), so that imaged objects can be easily scaled and placed wherever desired on the corresponding flow mesh. For example, flow domains in this work are typically constructed with dimensions of $\mathcal{O}(1)$ or $\mathcal{O}(10)$, having grid spacing $\Delta x \ll 1$ and containing objects that are of $\mathcal{O}(1)$ in size, in order to non-dimensionalize the fluid flow problems and to facilitate the variation of important parameters like the Reynolds number. Thus, an imaged object with dimensions of, for example, 1000 pixels in length and 200 pixels in height is converted to a modeled object of 1.0 unit length and 0.2 unit height by multiplying each (x, y) pair in the image domain by a scaling parameter s. In addition, x- and y-shifting constants d_x and d_y may be added to the (x, y) image addresses if necessary, to move the zero level set interface some desired distance away from the flow domain boundaries.

7.3 Image Advection using Optical Flow

It was demonstrated in Chapter 6 that the nonlinear optical flow method of Brox et al. offers a possible route to describing the motions of objects through image frames with a relatively high degree of accuracy. In particular, linear motion was captured quite well, even when the geometry was complex and the displacements between image frames were large. Angular motion errors, on the other hand, were found to be considerably higher, but it was hoped that an accurate description of motion could still be achieved in real image sequences, provided that they have good enough spatio-temporal resolutions to justifiably approximate their motions as linear within local pixel neighborhoods.

Chapter 6 also illustrated the effectiveness possessed by the optical flow method of Brox et al. at iteratively displacing brightness patterns to their correct locations; there is little discernable difference between the displaced source frame intensity field, $I_{source}(\boldsymbol{x}+\boldsymbol{w})$, and the target intensity field, $I_{target}(\boldsymbol{x})$, being iterated toward once the solution is found in most cases. Thus, it was hoped that supplying an image or a segmented object boundary with its corresponding optical flow vector field would be sufficient to move it to its correct target location in the next frame of the sequence. However, in practice this did not turn out to be the case.

The iterative process followed in the method of Brox et al. is nonlinear, so pixels undergoing displacement do not, in general, follow the straight path that is represented by the final result in reaching their target destination. Thus, when the image brightness is advected in the usual sense by projecting the optical flow vectors onto the intensity gradient at each pixel location with the assumption of brightness constancy, that is, by solving the advection problem

$$\frac{\partial I}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} I = 0, \qquad \textbf{(7.2)}$$

or

$$\frac{\partial I}{\partial t} = -u\frac{\partial I}{\partial x} - v\frac{\partial I}{\partial y}, \qquad \textbf{(7.3)}$$

the same end result is not achieved. It can be seen that this way of moving brightness patterns becomes particularly inaccurate in regions where optical flow vectors are oriented normal to the brightness gradient; $\boldsymbol{u} \cdot \boldsymbol{\nabla} I \to 0$ in this situation, resulting in little or no motion of the imaged objects there regardless of the magnitude of the optical flow vector $|\boldsymbol{u}|$.

Attempts to move the level set representation of imaged objects under the influence of optical flow vectors failed for precisely the same reason. According to the level set equation, the motion of a level set occurs in a direction normal to itself according to some speed function F:

$$\frac{\partial \varphi}{\partial t} + F|\nabla\varphi| = 0. \quad \textbf{(7.4)}$$

In this case, the speed function F is composed of the optical flow vectors projected onto the normal vectors of the level set field surrounding an object's boundary, or $F = \boldsymbol{u} \cdot \boldsymbol{n}$, giving

$$\frac{\partial \varphi}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{n}|\nabla\varphi| = 0. \quad \textbf{(7.5)}$$

Rewriting the normal vector in terms of the gradient of the level set field,

$$\boldsymbol{n} = \frac{\nabla\varphi}{|\nabla\varphi|}, \quad \textbf{(7.6)}$$

the level set equation becomes

$$\frac{\partial \varphi}{\partial t} + \boldsymbol{u} \cdot \frac{\nabla\varphi}{|\nabla\varphi|}|\nabla\varphi| = 0, \quad \textbf{(7.7)}$$

or

$$\frac{\partial \varphi}{\partial t} + \boldsymbol{u} \cdot \nabla\varphi = 0. \quad \textbf{(7.8)}$$

This leaves us in the same predicament we had with the brightness advection equation (Equation 7.2), because the gradient of a level set field about an object boundary is oriented the same way as the pixel intensity gradients marking the "edge" of that object, and so attempts to advect the level set with the optical flow vectors give the same results as similar attempts to advect the intensity field.

Figure 7-1 and Figure 7-2 help to elucidate the problems encountered during optical flow-based advection. Figure 7-1 shows five intermediate zero level set positions describing the contour of the swimming American eel, taken at regular intervals for each of two optical flow field solutions that are temporally adjacent. It can be seen that along the part of the eel's body away from the tail near the left side of the figure, the level set contours are spaced fairly regularly, indicating near constant motion there, which is the desired result. In this part of the image, the optical flow vectors are predominantly normal to the interface, pointing in the direction of the level set gradient and thus in the direction of the image brightness gradient. However, at the tip of the eel's tail, motion is almost completely tangential to the level set normal vectors, resulting in a large under-prediction of motion there (Figure 7-2). The result is that the tail makes large jumps in position between the final advection step of one image frame pair and the first advection step of the subsequent image pair.

So, despite the fact that the optical flow vectors produced by the method of Brox et al. possess a high degree of accuracy, another method of moving interfaces from one location to another in intermediate steps between image frames was needed. This is the subject of the next chapter section.

7.3 Introduction to Image Morphing

Image morphing is defined as the process of constructing a smooth, natural-looking sequence of images between an ordered image pair [88], consisting of two steps. The first is warping, which typically involves a nonlinear coordinate transformation that aligns user-defined landmarks while imposing regularity or smoothness constraints on the coordinates that lie between landmarks [88]. The second step is blending, which fills in the differences in detail and color not captured during the warping process, and is typically based on pixel color/intensity interpolation. Applying the two steps incrementally results in a smooth transition from one image to the other that includes both the shape (warping) and the intensity/color (blending) [88].

Unfortunately, the warping/blending approach assumes that all relevant image properties have been properly matched during the warping step. Let $u_{i,j}$ denote a picture in the first (source) image and $v_{i,j}$ a corresponding pixel in the second (target) image, where $i = 1, \dots, N$ and $j = 1, \dots, M$ on a pair of images that are each of size $N \times M$. The method of *cross dissolving* one image to the next is accomplished by setting a transition parameter $\alpha \in [0,1]$, so that the blending process gives a family of images $w_{i,j}(\alpha)$ in which $w_{i,j}(0) = u_{i,j}$ and $w_{i,j}(1) = v_{i,j}$. The cross dissolving method of image blending is then simply a linear interpolation, and can be represented as

$$w_{i,j}(\alpha) = (1 - \alpha)u_{i,j} + \alpha v_{i,j} \tag{7.9}$$

When the source and target images are not properly matched, cross dissolving creates a "ghosting" effect (Figure 7-4) in which the source image appears to fade out of the image domain while the target image fades in—an effect that is quite noticeable in regions of high contrast, such as object boundaries, thereby giving limited usefulness to this method for the purposes of image-based modeling.

In [88], the author (Whitaker) argues that better approaches to image blending place fewer demands on the warping process, thereby reducing the need for user input in the morphing process overall, and so improvements in image blending methods are important for applications that must generate sequences of images with little to no user input. In Whitaker's proposed approach, transition images between given image frames are not constructed by simple interpolation as in Equation 7.9, but rather are generated based on the shapes of level sets in the input images. A distance metric is constructed which penalizes images based on difference in the shapes of their level sets, and thus acts to deform the shapes of image objects as they are evolved between frames. This results in a set of transition images that represent intermediate shapes naturally through the blending process.

Treating an image as a continuous function $F\colon \Omega \mapsto I$, where $\Omega \subset \mathbb{R}^2$ is the image domain and $I \subset \mathbb{R}$ is the set of intensity values contained in the image, we let $F(x, y)$ and $G(x, y)$, with $(x, y) \in \Omega$, represent a source and a target image, respectively. The image blending strategy, then, involves constructing a family of images, indexed by α and starting with the source image $F(x, y)$ at $\alpha = 0$, which progressively appear more like the target image $G(x, y)$ as α increases [88].

Regarding the functions F and G as representations of points in some higher-dimensional function space, a linear interpolation is simply a straight line path through the function space between F and G, parameterized by α (Figure 7-3 a). The path from F to G may be constructed by gradient descent over parameter t on a distance function that approaches zero as $F \rightarrow G$. Alternatively, a path may be constructed on which F and G are both allowed to progress toward each other simultaneously with respect to t (Figure 7-3 b). The paths of these functions through the function space can be defined as $f(x, y, t)$ and $g(x, y, t)$, where $f(x, y, 0) = F(x, y)$ and $g(x, y, 0) = G(x, y)$, and the point where they meet in function space is the transition image between F and G. In this way, a blend, $b(x, y, \alpha)$, is obtained by reparameterizing the solutions of f forward in time and the solutions of g backward in time to produce a sequence from F to G with respect to the parameter α.

Clearly, different metrics will produce different paths through the function space occupied by F to G, and will thus lead to different blends between image frames (Figure 7-3 c). A metric typically employed in image analysis is the L_2 norm:

$$\mathcal{E}(t) = \frac{1}{2}\int_{\Omega}\big(f(x,y,t) - g(x,y,t)\big)^2 dx\, dy \quad \textbf{(7.10)}$$

Setting the temporal derivatives of f and g to the negative of the first variation of Equation 7.10 gives a system of gradient descent equations over time t,

$$\frac{\partial f(x,y,t)}{\partial t} = g(x,y,t) - f(x,y,t) \quad \textbf{(7.11)}$$

$$\frac{\partial g(x,y,t)}{\partial t} = f(x,y,t) - g(x,y,t), \quad \textbf{(7.12)}$$

where $f(x,y,0) = F(x,y)$ and $g(x,y,0) = G(x,y)$. Since Equations 7.11 and 7.12 only contain derivatives in t, x and y may be regarded as parameters, and thus they may be treated as a coupled pair of ordinary differential equations at each point in the domain, with the solution

$$f(x,y,t) = \frac{1}{2}\big(G(x,y) + F(x,y)\big) + \frac{1}{2}\big(F(x,y) - G(x,y)\big)e^{-2t} \quad \textbf{(7.13)}$$

$$g(x,y,t) = \frac{1}{2}\big(G(x,y) + F(x,y)\big) + \frac{1}{2}\big(G(x,y) - F(x,y)\big)e^{-2t}. \quad \textbf{(7.14)}$$

The steady-state solution $f(x,y,t) = g(x,y,t) = \frac{1}{2}\big(G(x,y) + F(x,y)\big)$ is the transition image between F and G.

Because Equations 7.13 and 7.14 represent an exponential decay toward the transition image, rather than a linear interpolation as given in Equation 7.9, the final blend is obtained by

$$b(x,y,\alpha) = \begin{cases} f(x,y,-\ln(1-2\alpha)) \text{ for } 0 \le \alpha \le \frac{1}{2} \\ g(x,y,-\ln(2\alpha-1)) \text{ for } \frac{1}{2} \le \alpha \le 1 \end{cases}. \quad \textbf{(7.15)}$$

Here it is noted that such a distance metric formulation on the image domain is limited in that it only considers single-pixel comparisons between images, and that it is sensitive to nonlinear transformations acting on the intensity functions f and g. Therefore, it is proposed to treat the image domain as a set of isocontours (level sets) rather than a collection of individual pixels. This allows for inter-pixel relationships to be considered in the difference metric when blending an image from one frame to the next.

Defining the k^{th} level set of path f through function space from F to G as

$$\mathcal{L}_k = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \middle| \, f(x,y) = k \right\}, \tag{7.16}$$

a point's location can be determined to be either "inside" or "outside" of the level set k by applying the Heaviside function to the values of f: $\mathcal{H}(f(x,y) - k)$. (The definition of what is "inside" versus "outside" of a level set is arbitrary, as long as the definition is applied consistently.)

Using this new level set definition of the image domain, a distance metric may be constructed as before, but now based upon the differences between regions contained within level sets on the two images (f and g), so that the metric's potential is proportional to the area of level set shapes in one image that are not in the other. Defining functions of f and g that are negative inside the k^{th} level set and positive outside,

$$\mathcal{D}_k[f] = \tfrac{1}{2}\big(\mathcal{H}(k - f) - \mathcal{H}(f - k)\big) \tag{7.17}$$

$$\mathcal{D}_k[g] = \tfrac{1}{2}\big(\mathcal{H}(k - g) - \mathcal{H}(g - k)\big), \tag{7.18}$$

the overall distance metric for level set k is given by

$$\mathcal{E}(t) = \tfrac{1}{2}\int_\Omega (\mathcal{D}_k[f(t)] - \mathcal{D}_k[g(t)])^2 dx\, dy = \tfrac{1}{4}\int_\Omega \Big(\big[\mathcal{H}\big(k - f(t)\big) - \mathcal{H}(f(t) - k)\big] - \big[\mathcal{H}\big(k - g(t)\big) - \mathcal{H}(g(t) - k)\big]\Big)^2 dx\, dy. \tag{7.19}$$

Minimizing the metric with respect to a single level set k leads to a pair of Euler-Lagrange equations

$$-d\mathcal{E}_{k,f} = \frac{1}{2}[\mathcal{H}(k-g)\delta(k-f) + \mathcal{H}(k-g)\delta(f-k) - \mathcal{H}(g-k)\delta(k-f) - \mathcal{H}(g-k)\delta(f-k)] \quad \textbf{(7.20)}$$

$$-d\mathcal{E}_{k,g} = \frac{1}{2}[\mathcal{H}(k-f)\delta(k-g) + \mathcal{H}(k-f)\delta(g-k) - \mathcal{H}(f-k)\delta(k-g) - \mathcal{H}(f-k)\delta(g-k)], \quad \textbf{(7.21)}$$

where δ is the Dirac delta function (the derivative of $\mathcal{H}$), and $\big(\mathcal{H}(x) - \mathcal{H}(-x)\big)\delta(x) = 0$. Because $\delta(-x) = \delta(x)$, Equations 7.20 and 7.21 reduce to

$$-d\mathcal{E}_{k,f} = \frac{g-k}{|g-k|}\delta(f-k) \quad \textbf{(7.22)}$$

and

$$-d\mathcal{E}_{k,g} = \frac{f-k}{|f-k|}\delta(g-k). \quad \textbf{(7.23)}$$

Since the δ functional performs a wave front propagation of the k^{th} level set of f, while the gradient magnitude of f gives the same result on all level sets of f simultaneously, the Dirac delta term $\delta(f-k)$ in Equation 7.22 can be remapped to $|\nabla f|$ whenever $f = k$. (The same holds true for g in Equation 7.23.) Thus, performing gradient descent on f and g, and parameterizing the sequence of images by t gives a pair of differential equations which are the level set equivalent of Equations 7.11 and 7.12

$$\frac{\partial f}{\partial t} = \frac{g-f}{(\epsilon^2+(g-f)^2)^{1/2}}|\nabla f| \quad \textbf{(7.24)}$$

$$\frac{\partial g}{\partial t} = \frac{f-g}{(\epsilon^2+(f-g)^2)^{1/2}}|\nabla g|, \quad \textbf{(7.25)}$$

where ϵ is a small constant (set to 10^{-4} in [88]) provided to safeguard against division by zero in the limiting case where $f = g$.

In the pixel-based linear interpolation given by Equation 7.9, the resulting distance metric equation (Equation 7.10) was solved using gradient descent to give a pair of coupled ordinary differential equations (7.11 and 7.12), and then resampled from the time parameter t to the transition parameter α. This resampling equation (7.15) was constructed metric to ensure that the sum of absolute values of image differences would decrease at a constant rate with respect to α, giving a blend that appears to change consistently as α is varied from 0 to 1. This type of behavior may be replicated in the level set formulation of image blending by adjusting the sampling rate to give a constant change in some image metric $\mathcal{D}(t)$, thereby giving the intervals in t at which to take "snapshots" in order to produce the desired sequence of blended images. Due to its straightforward implementation and the qualitatively good results it is able to consistently give, Whitaker proposes a root-mean-squared metric,

$$\mathcal{D}(t) = \left[\int_\Omega (f-g)^2 dx\, dy\right]^{1/2}, \qquad \textbf{(7.26)}$$

with the discrete form

$$\mathcal{D}(t) = \sum_{i=1}^{n} \sum_{j=1}^{m} \left(u_{i,j} - v_{i,j}\right) \qquad \textbf{(7.27)}$$

to be used during implementation.

The algorithm given to produce n transition images between frame F and frame G was thus given as follows:

1) Compute the difference metric on the source and target images to give

$$\mathcal{D}(0) = \left[\int_\Omega (F-G)^2 dx\, dy\right]^{1/2}, \qquad \textbf{(7.28)}$$

2) For each of the k transition images out of n total transition images between image frames, solve the level set blending equation with forward differencing until a time t is reached which satisfies

$$\mathcal{D}(t) = \frac{n-k}{n}\mathcal{D}(0). \tag{7.29}$$

In this way, image morphing proceeds consistently through the intermediate steps between image frames.

Another method proposed by Mukherjee and Ray [89] takes an alternative approach to that of Whitaker, applying morphing to a level set that is defined by a single segmentation contour. This was considered worth investigating, as it is directly applicable to the idea of morphing a modeled object in order to ascertain intermediate configurations within the framework already developed. Recall that the level set equation describes the motion of a level set in a direction normal to itself according to some speed function F. Mukherjee and Ray proposed a speed function that incorporates three different effects: an elastic force applied to the zero-level curve, a curvature-dependent force, and a speed contribution that comes from the optical flow vectors surrounding the moving object segment.

The elastic (expansion-contraction) force is supplied to a segmentation curve by defining a distance metric similar to that given in Whitaker's level set formulation, i.e.

$$\mathcal{D}(x,y) = \mathcal{H}\big(\varphi_S(x,y)\big) - \mathcal{H}\big(\varphi_T(x,y)\big), \tag{7.30}$$

where φ_S and φ_T represent level set representations of segments in the source and target images, respectively, and $\mathcal{H}$ represents the Heaviside function as usual. Level set contours are again defined as separating an "inside" region from an "outside" region, so that the distance metric takes a nonzero value when source and target level sets are separated by some distance, and approaches zero as the morphed level set $\varphi(x,y) \rightarrow \varphi_T(x,y)$.

Thus, the evolution of a level set curve under the influence of the elastic force defined in Equation 7.30 is given by

$$\frac{\partial\varphi}{\partial t} = -\mathcal{D}(x,y)|\nabla\varphi|. \tag{7.31}$$

In addition to the distance metric Equation 7.31, Mukherjee and Ray introduce a curvature-dependent force, β_c, to the speed function in order to force level set curvature toward its target value throughout the morphing process.

$$\beta_c = \kappa\big(c(s,t)\big) - \kappa\big(c_F(s)\big) \tag{7.32}$$

In Equation 7.32, κ represents the curvature of any level set curve c, and is given by

$$\kappa\big(c(s,t)\big) = \boldsymbol{\nabla}\cdot\left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}\right) = \boldsymbol{\nabla}\cdot\boldsymbol{n}. \tag{7.33}$$

Level set motion under the influence of curvature differences, then, is simply

$$\frac{\partial\varphi}{\partial t} = -\beta_c|\boldsymbol{\nabla}\varphi|. \tag{7.34}$$

Finally, Mukherjee and Ray incorporated optical flow-based shape deformation into the morphing process, arguing that motion vectors on each level set contour direct to a new position of the initial curve, and therefore should yield a curve which has evolved with respect to its initial position due to optical flow. Assuming level set evolution in the normal direction, the optical flow vectors are projected onto the unit normal vectors on the level set field to give their speed contribution:

$$\frac{\partial\varphi}{\partial t} = \left\langle -(u,v)\cdot\left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}\right)\right\rangle|\boldsymbol{\nabla}\varphi|. \tag{7.35}$$

Combining the effects of the elastic force, curvature-dependent force, and the optical flow field into a single speed function, the final level set evolution equation is

$$\frac{\partial\varphi}{\partial t} = -\left(\mathcal{D}(x,y) + \beta_c\left\langle(u,v)\cdot\left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}\right)\right\rangle\right)|\boldsymbol{\nabla}\varphi|, \tag{7.36}$$

which reduces to

$$\frac{\partial\varphi}{\partial t} = -\mathcal{D}(x,y) - \beta_c \left\langle (u,v) \cdot \left(\frac{\nabla\varphi}{|\nabla\varphi|}\right)\right\rangle \qquad \textbf{(7.37)}$$

when the level set field is maintained as a signed distance function.

7.4 The Image Morphing Algorithm

In practice, the method proposed by Mukherjee and Ray did not work for our purposes. While the elastic force turned out to morph the level set field from one frame to the next quite well on its own, adding the other terms only served to detract from its ability to reach the target solution in the regular fashion being sought. Adding the optical flow advection term to the speed function was especially problematic; as was described in Section 7.3, the optical flow information did little to alter the morphing path of regions where the level set gradient was orthogonal to local optical flow vectors, such as the end of the eel's tail. In regions where optical flow vectors were oriented in the same direction as the level set gradient (normal to the body's surface), the additive effect of the vectors on the morphing process caused an over-prediction of motion. This resulted in the level set curve "overshooting" its target position, which then had to be corrected for by the elastic force acting in a direction back toward the source again. Thus, adding these extra terms proved to be counterproductive, so it was decided to morph the level set field using only the elastic force, employing the root-mean-squared metric proposed in [88] to evolve the level set curve in a regular fashion between image frames.

Of course, the elastic force chosen to produce morphing is not linear, but rather decays exponentially as $\mathcal{D}(x,y) \to 0$ (Figure 7-7), so morphing must take place on a third time scale ΔT_{morph}, ranging between $t_{morph} = 0$, when $\alpha = 0$ and $w_{i,j}(0) = u_{i,j}$, and $t_{morph} = \infty$, when $\alpha = 1$ and $w_{i,j}(1) = v_{i,j}$.

In order to deal with the disparity between the morphing time scale and the image time scale, the root-mean-squared metric proposed by Whitaker was used to obtain a sampling rate in which morphed image objects would match up with their correct locations throughout progression through the fluid flow time scale (Figure 7-8). The algorithm is summarized as follows:

1) Calculate the initial image metric value $\mathcal{D}(0) = \left[\int_{\Omega}(F - G)^2 dx\, dy\right]^{1/2}$.

2) For any t_f^* falling between flow time intervals t_{f1} and t_{f2}, find the corresponding T_I^* between image frame times $T_{Image,o} = 0$ and $T_{Image,f} = 1$ using an appropriate scaling relationship between t_f and T_I, i.e.

$$T_I^* = \frac{t_f^* - t_{f1}}{\Delta T_{flow}}. \quad \textbf{(7.39)}$$

3) Evolve the morphing process until the appropriate morphing time t_{morph}^* by checking the metric value

$$\mathcal{D}\left(t_{morph}^*\right) = \mathcal{D}(0) + \frac{\mathcal{D}(0) - \mathcal{D}(\infty)}{T_{Image,o} - T_{Image,f}}\left(T_I^* - T_{Image,o}\right). \quad \textbf{(7.40)}$$

Because $\mathcal{D}(\infty) = 0$ when morphing is complete, and $T_{Image,o} = 0$ and $T_{Image,f} = 1$, Equation 7.40 reduces to

$$\mathcal{D}\left(t_{morph}^*\right) = \mathcal{D}(0)(1 - T_I^*). \quad \textbf{(7.41)}$$

The morphing process is evolved until Equation 7.41 is satisfied, thus giving morphed images at regular time intervals corresponding to the intervals at which flow information is plotted.

Like Figure 7-1, Figure 7-5 shows five intermediate zero level set positions describing the swimming American eel, taken at regular intervals for each of two optical flow field solutions that are temporally adjacent. Comparing to Figure 7-1, it can be seen that the jumps in image position between the final morph of one frame and the initial state of the next have been eliminated; a consistent motion of the boundary has been achieved through use of the elastic force model and an appropriate sampling rate. Figure 7-6 gives a close-up view of the tail region again, just as Figure 7-2 did, illustrating the morphing process through 3 image frame pairs.

Like any method, the elastic force model chosen for image morphing is not perfect. The primary difficulty found with respect to our purposes involves the morphing path rather than the final result. Careful examination of Figure 7-5 and Figure 7-6 reveals that the eel's tail does not take a linear trajectory when morphing from one frame to the next, but rather takes one in which the eel's overall body length appears to slightly shorten and then lengthen again. This happens because the distance metric has a zero value where the initial and final curves cross (Figure 7-9); the curves are not separated by any distance there, and so there is no driving potential for motion. The effect is that curve motion can occur around this zero-potential point, but not through it, so it acts as a sort of node during the iterative morphing process. This problem will always exist when initial and target fields overlap, but can be minimized in its effect with greater temporal resolution, so that curves are never greatly displaced between frames and the relative effect of zero-potential curve crossings becomes smaller.

7.5 Interpolation onto the Flow Mesh

The morphing algorithm just outlined provides a temporal interpolation from a coarsely sampled image time scale to a finely sampled fluid flow time scale. Similarly, modeled objects must be spatially interpolated from a coarse image domain consisting of pixels to a fine flow domain consisting of grid cells that may be locally refined to capture flow phenomena of interest. Thus, the final process left in generating purely Eulerian CFD models involved mapping the level set fields obtained by image analysis onto the flow domain.

This mapping was performed in several steps. First, the image pixel locations were converted to real (x, y) pairs (mapping the image domain $\Omega \longmapsto \mathbb{R}^2$), so that imaged objects could be easily scaled and placed wherever desired on the corresponding flow mesh. For example, flow domains in this work are typically constructed with dimensions of $\mathcal{O}(1)$ or $\mathcal{O}(10)$, having grid spacing $\Delta x \ll 1$ and containing objects that are of $\mathcal{O}(1)$ in size, in order to make problems more dimensionally intuitive and to facilitate the variation of important parameters like the Reynolds number. Thus, an imaged object with dimensions of, for example, 1000 pixels in length and 200 pixels in height could be converted to a modeled object of 1.0 unit length and 0.2 unit height by multiplying each (x, y) by a scaling parameter s. In addition, x- and y-shifting constants d_x and d_y could be added to the (x, y) image addresses if necessary, to move the zero level set interface some desired distance away from the flow domain boundaries. The result was a scaled, shifted version of each pixel location $\hat{x}_{i,j} = s\, x_{i,j} + d_{i,j}$.

With each of the image pixels defined as a point with a scaled and shifted spatial address, the *flow mesh* points were swept over to examine whether they were within the bounds of $\hat{x}_{i,j}$, i.e. between $\hat{x}_{min}$ and $\hat{x}_{max}$ in the horizontal direction, and between $\hat{y}_{min}$ and $\hat{y}_{max}$ in the vertical direction. If they were in fact found there, then their 4 nearest neighbors on the corresponding scaled and shifted image domain were used to interpolate the image level set value onto the flow mesh via bilinear interpolation. Because a narrow band level set field was used in constructing interfaces, anything in the fluid domain outside of the narrow band was set to the outermost level set value, e.g. $6\Delta x$, so that the result was a fluid flow mesh consisting of a level set value of $6\Delta x$ everywhere except for within the narrow band field describing an interface (or beyond the inner edge of the narrow band, where the level set value was set to $-6\Delta x$).

It is important to note here that when an image domain is rescaled and shifted for mapping to the fluid mesh, Δx_{image} is still not the same as Δx_{flow} in general. Thus, when mapping level set values from an image domain onto a fluid domain, the level set values must be scaled appropriately. Before scaling, the pixel Δx_{image} was assumed to be 1.0 for simplicity, giving a field of narrow band level set values ranging between ± 6. In this work, those values were maintained even after image rescaling, so that the level set field could simply be multiplied by Δx_{flow} after mapping to get the correct values.

Although the elastic force model was chosen for image morphing due to the limitations of optical flow-based advection on its own, optical flow vectors are still used to set boundary conditions on the zero level set contour in moving boundary flow simulation problems; indeed, optical flow may be the only way to do this in a Eulerian setting without reverting back to the use of surface points. So, the optical flow vector field was interpolated onto the flow mesh from the image domain just as the level set field values were. Since optical flow vectors are computed on image sequences under the assumption that pixels are unit size and frames are separated by some arbitrary unit time, the resultant vector field is in terms of $pixels/frame$. The displacement aspect of the optical flow velocity field was scaled by multiply by Δx_{flow} just as the level set field was, but the temporal component was set separately based on the physics of the problem, by simply determining what the frame rate should be in order to produce a desired velocity within the flow domain and setting ΔT_{image} accordingly.

Since the optical flow field is piecewise constant between image frames, the 3-frame strategy described in Chapter 3 was also used here to create a piecewise linear optical flow field, thereby averting the discontinuous jumps in velocity otherwise produced by a piecewise constant profile:

$$\boldsymbol{u} = \boldsymbol{u}_{1\rightarrow 2} + \boldsymbol{u}_{2\rightarrow 3}\Delta T^{*}_{image}, \tag{7.42}$$

where $\boldsymbol{u}_{1\rightarrow 2}$ and $\boldsymbol{u}_{2\rightarrow 3}$ represent optical flow velocity vectors solved between on ordered frame pair and the next, respectively.

It is noted that an image must be morphed and re-mapped to the corresponding flow domain after each fluid time step during CFD simulations. While this accounts for little of the total run time compared with solving for the optical flow field, or certainly compared to solving for the fluid flow field, it would still be computationally wasteful to sweep over the entire flow domain every single time step, find corresponding image domain locations, and then update the values everywhere (even if they're "updated" to the same value). To eliminate this wastefulness, the flow solver code has a built-in memory structure containing only points that lie within the narrow-band level set, storing their spatial locations within the flow domain along with their level set values. Mapping the image level set and interface velocity from the scaled and shifted image domain to the flow domain need only take place within the narrow band where such information is necessary for setting boundary conditions, and by sweeping over these level set "tube" points rather than the entire flow domain, the vast majority of flow grid points can often be eliminated from the search and update process.

7.6 CFD: Revisiting the American Eel Simulation

To demonstrate the entire framework developed in this thesis, i.e. directly linking the image information to computations, the swimming American eel simulation that was discussed in detail in Chapter 3 was simulated once again. The domain was set up precisely as before, with dimensions of 5.0 units length in the x-direction and 2.0 units height in the y-direction. The eel's body was scaled to be unit length, and it was placed in the domain centered vertically and placed one unit length from the inlet. A base mesh size $\Delta x_1^{flow} = 0.0125$ was assigned, with four levels of refinement giving a minimum mesh size of $\Delta x_4^{flow} = 0.0025$. The domain was assigned an inlet boundary condition with velocity $U = 1.0$ on the left side, a passive outlet condition on the right side, and Dirichlet boundary conditions with $U = 1.0$ on the upper and lower sides as before, in an attempt to approximate open flow conditions. As in the Lagrangian model before, the Reynolds number was set to $\mathrm{Re} = 5000$, and three different Strouhal numbers – 0.3, 0.5, and 0.7 – were evaluated by setting $\Delta T_{image} = 0.01389$, $\Delta T_{image} = 0.00833$, and $\Delta T_{image} = 0.005952$, respectively (yielding tail beats that took $0.5\ s$, $0.3\ s$, and $0.21429\ s$ to complete).

Despite the fact that both the Eulerian and Lagrangian cases were set up in the same manner, the two methods produced distinctly different results. Plots of vorticity are supplied in Figure 7-10 through Figure 7-13, y-velocity in Figure 7-14, and x-velocity in Figure 7-15; Side-by side comparisons of the Lagrangian and Eulerian methods are provided in Figure 7-16. Though the methods share some common traits – both transition to a wake with a thrust signature at a Strouhal number of $\mathrm{St} = 0.7$, for example – it is immediately evident that the wake structures have significant differences in all Strouhal number regimes between the two methods. In the original Lagrangian method, each traverse of the eel's tail generated a shear layer with an unstable aspect ratio, causing it to separate into two vortices of the same sense. Vorticity was diffused quickly in the wake, but the beginnings of a 2p-type wake structure appeared to be developing. The new Eulerian method, on the other hand, produced one distinct vortex with each reversal of the eel's tail, resulting in a more prototypical vortex street. An unstable shear layer was developed as before, but it did not separate into 2 vortices like the shear layer in the original calculation did.

Examining the surface of the eel's body a little bit closer reveals some possible reasons for this disparity. The first reason has to do with boundary conditions. Recall from Chapter 3 that the Lagrangian surface meshing algorithm employed in the original method required a one-to-one point correspondence between sets of surface points in each of the three frames being used, in order to calculate interface position and velocity. Thus, each of the surface meshes required the same number of points. However, because curve length was not conserved between image frames, the surface points had to be re-spaced each time the image frame was updated, giving surface meshes with points arranged in equal intervals. Such translation of the points along the surface introduced spurious tangential velocities in the boundary conditions on the surface of the eel; velocities that were often in the wrong direction altogether.

Figure 7-17 and Figure 7-18 illustrate velocity vectors near the tail region for each of the two methods used. It is plain that the surface vectors produced by the Lagrangian method are completely incorrect; there is a strong tangential component pointing from the eel’s tail toward its front, a result completely at odds with what is expected. The optical flow surface vectors, on the other hand, behave according to expectations, with strong normal components everywhere except where there is a reversal in the eel’s side-to-side motion (at which point the surface vectors become very small). In any case, the vectors produced by optical flow are never seen to point in the opposite direction to where they should.

Examining the strength of vorticity along the eel’s body, it was found to be stronger overall in the original method ($\sim 40\ s^{-1}$ versus $\sim 30\ s^{-1}$), likely due to the incorrect tangential velocity component on the surface, which pointed opposite to the direction of the flow and thereby effectively increased skin friction and the flow’s angular velocity there. This translated into a stronger shear layer being shed from the tail during its traverse from one side to the other, leading to the formation of a stronger vortex pair once the shear layer separated (Figure 7-19). It is interesting to note that these incorrect boundary conditions actually led to a wake structure that was closer to what was expected based on experimental data, suggesting that a higher Reynolds number may need to be simulated before the expected behavior is seen using the new optical flow method.

Another possible reason for differences in vortical strength between the two methods involves geometry. When the surface was meshed with Lagrangian points, a good deal of smoothing had to be performed on the set of points in order to recover a curve sufficient for flow modeling. In the process of smoothing the points, some of the geometric information describing the eel was lost. This can be easily seen in Figure 7-20, which shows the zero level set curve for each method at one instant in time, as well as in

Figure 7-21, which shows the nose of the eel up close. The tail region suffered the greatest deformation in the original case, with smoothing acting to flatten the high curvature there. Since this is where all of the vortex shedding takes place, it is reasonable to expect that the tail's geometry plays a significant role in what happens beyond it in the wake. The nose of the eel was also significantly deformed in the original method, yielding a streamlined shape which could have affected the development of the boundary layer downstream, and subsequently the vorticity on the body. The new optical flow-based method eliminates these surface smoothing operations, with error coming only from image denoising, and from the lack of image resolution, itself – both of which affected the original method as well.

Like in the original Lagrangian models, each of the Eulerian cases was examined with a control volume analysis that was set up in the same manner described in Chapter 3, for the purpose of measuring drag (or thrust) production by the eel during swimming. Figure 7-22 shows the instantaneous drag coefficients measured in the wake through one complete tail beat for each Strouhal number, and Figure 7-23 plots the average drag coefficient for each tail beat modeled using both the Lagrangian and Eulerian methods. The character of the instantaneous drag coefficients turned out to be quite different between the two methods, with the Eulerian model lacking the strong periodic thrust production that came with the start of each new tail beat as seen previously. This difference also shows up in the average coefficient values, as the Eulerian model did not show nearly the overall thrust production of the Lagrangian model. It is possible that this is due to the fact that the spurious point motion seen in the Lagrangian models favored more shear along the eel's surface, and hence the shedding of stronger vortices, or it could be due to differences in the overall geometric description between the two methods, or it could simply be a product of invalid control volume assumptions. Future analyses should include a more thorough control volume study, whereby the control volume can be more closely fitted to the body’s surface, and all flow variables can be accounted for along each of the control volume’s edges in order to eliminate the need for the assumptions made here.

7.7 Conclusions

In this chapter, it was shown that it is not only possible to generate computational models without the use of any surface meshing whatsoever, but that such computational models can potentially produce better results that more tedious schemes that build surface meshes on to images prior to computation. The original image-based modeling algorithm employed in this thesis was advantageous in that it enabled modeling complex geometries and motions directly from video files rather than having to immediately move to mathematical approximations. However, the requirement of surface meshing introduced its own set of assumptions, many of which altered the geometry sufficiently to nearly obviate the benefits of the method. In addition, boundary conditions produced by the assumption of point connectivity on Lagrangian surfaces translated to incorrect fluid flows near surfaces during CFD simulations.

The new Eulerian method completed for this thesis work dispenses with many of the limitations that are inherent in the Lagrangian approach, giving more realistic geometries and boundary conditions. The weaknesses of this approach lie mostly with the assumptions made regarding motion; in the optical flow calculations, objects are assumed to move along their intensity gradients. The constraint of smoothness, combined with the nonlinear iterative path taken by the method of Brox et al. in reaching a solution, give optical flow vector components that are not constrained to follow image gradients everywhere, but there are still problems with assessing purely tangential boundary motion correctly, particularly in regions that have little or no curvature.

The other potential limitation of the Eulerian method lies with the morphing process. The elastic force model used to move a level set curve between frames does not conserve curve length, and produces errors where curves cross each other at steep angles. In the case of the American eel, this created the effect of shortening the tail between image frames, leading to a level set motion that does match its boundary conditions in some isolated areas.

Even with these problems, however, the Eulerian method developed here is distinguished by its elegance; implementation is straightforward and entirely (i.e. all the way from the image domain to flow computations) follows the level set framework employed in the CFD code nicely. There is no tedious point generation algorithm, no surface smoothing, no assumptions regarding point correspondence and no need to convert from an Eulerian image field to a Lagrangian surface mesh and then back to an Eulerian field again. And absent the point-based assumptions made in the previous method, the new method produces results that appear to be more plausible, particularly with respect to boundary conditions.

Although the idea of Eulerian image based modeling has been demonstrated here using a low-resolution and relatively simple video of an eel swimming, the real power of the method lies in its promise with regard to extending to 3-D moving images obtained through modalities like ultrasound. Modeling an organ system such as the intestine functionally in 3-D would be extremely difficult, perhaps impossible, and meshing its surface with a set of points based on imagery would be a tedious, difficult process. However, if we can combine the denoising and segmentation algorithms outlined in Chapter 4 with the nonlinear optical flow method of Brox et al. used to set boundary conditions and morphing used to generate intermediate geometries, a great number of possibilities suddenly present themselves. Thus, at the culmination of this work, we have arrived at a paradigm that presents many avenues of extension to capture a wide range of moving boundary phenomena that occur in biofluid mechanics and other applications.

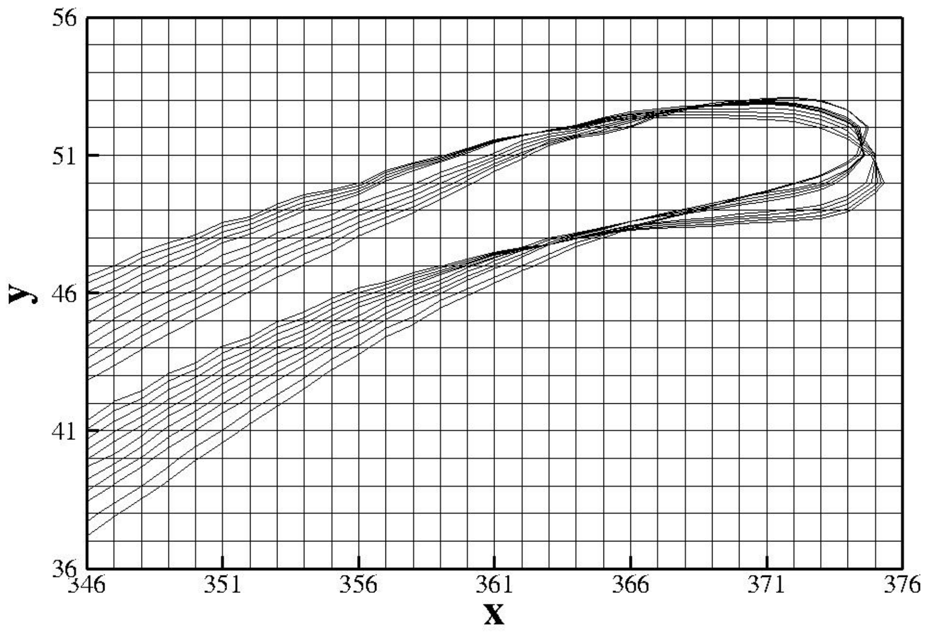


Figure 7-1. Morphing level set contours by optical flow alone proved inadequate for the purposes of image-based CFD modeling. In regions where optical flow vectors are oriented normal to level set boundaries (pixel columns 346-351), the motion is consistent. However, large jumps in position from one image frame to the next occur where the optical flow vectors are nearly orthogonal to the normal vectors (appearing most dramatically near the tip of the eel’s tail).

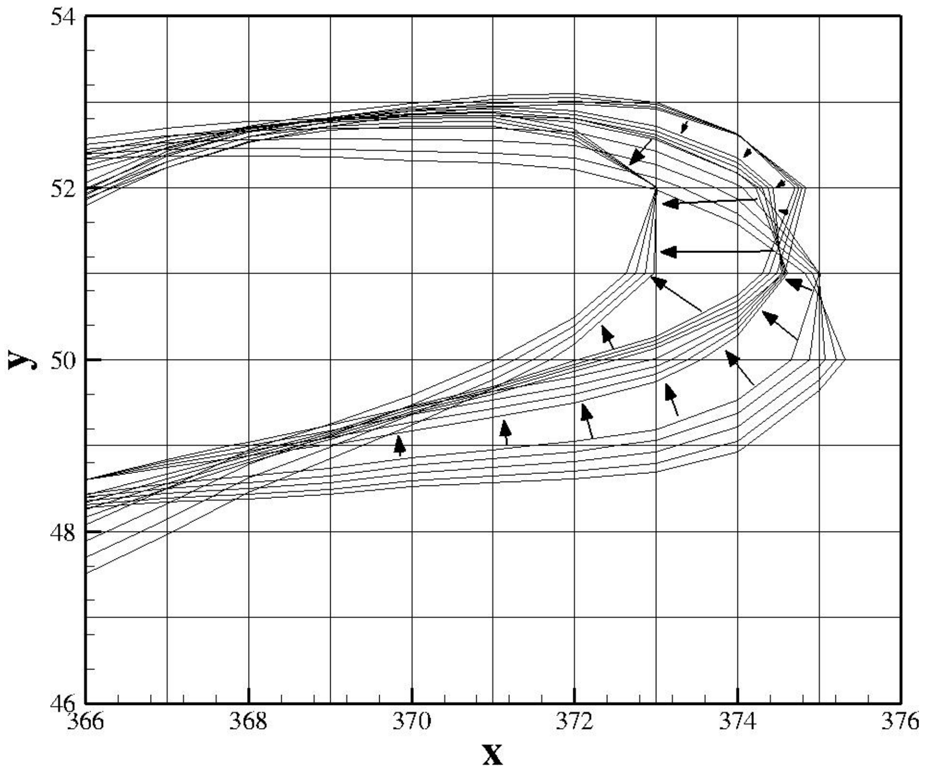


Figure 7-2. A close-up view of the tail tip region being moved with the optical flow vector field through three image frames.

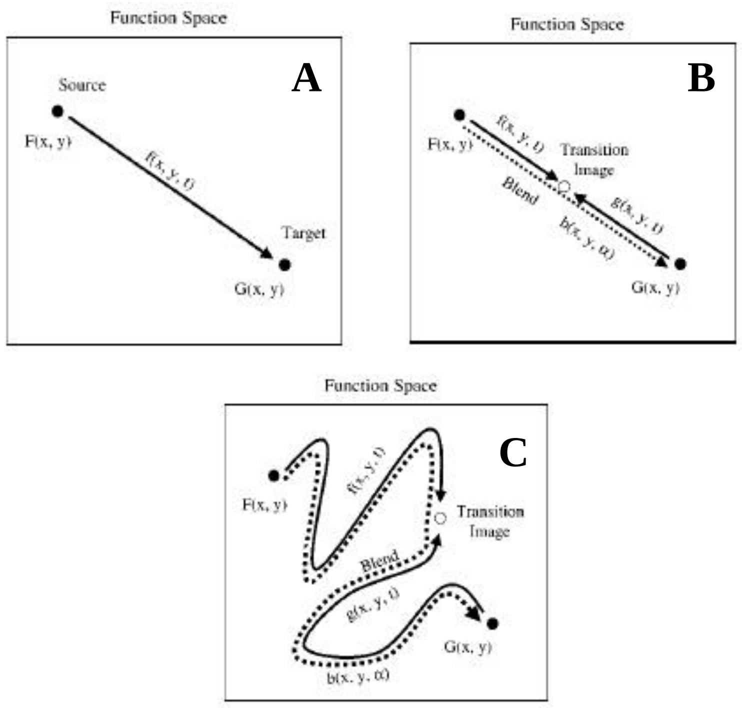


Figure 7-3. Image paths taken through a higher dimensional function space: (A) linear interpolation from a source image to a target image; (B) Linear interpolation between a source image and a target image to get an intermediate transition image; (C) A transition image resulting from a nonlinear metric comparing a source and target image. (Figures taken from [88].)

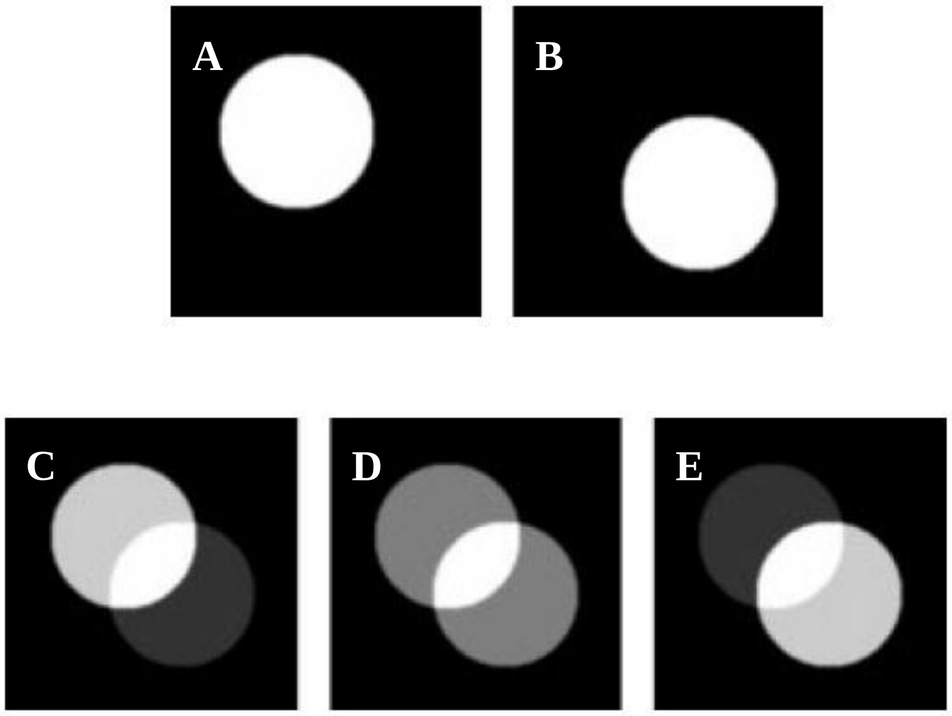


Figure 7-4. Blending by interpolation between a source image (A) and a target image (B) results in “ghosting,” a phenomenon in which the source image appears to fade out as the target image fades in (C-E). Figures taken from [88].

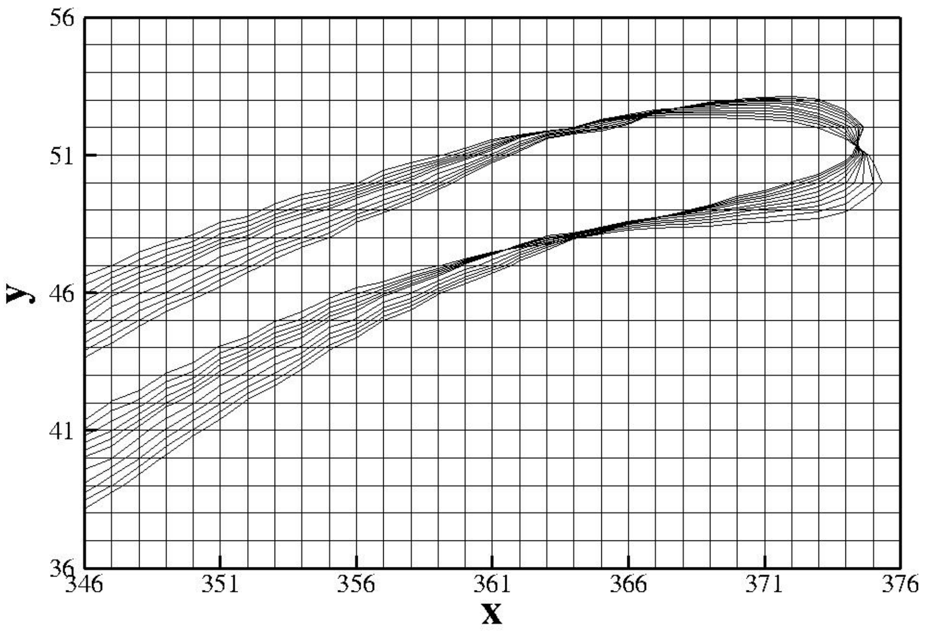


Figure 7-5. Morphing the level set using the elastic force model.

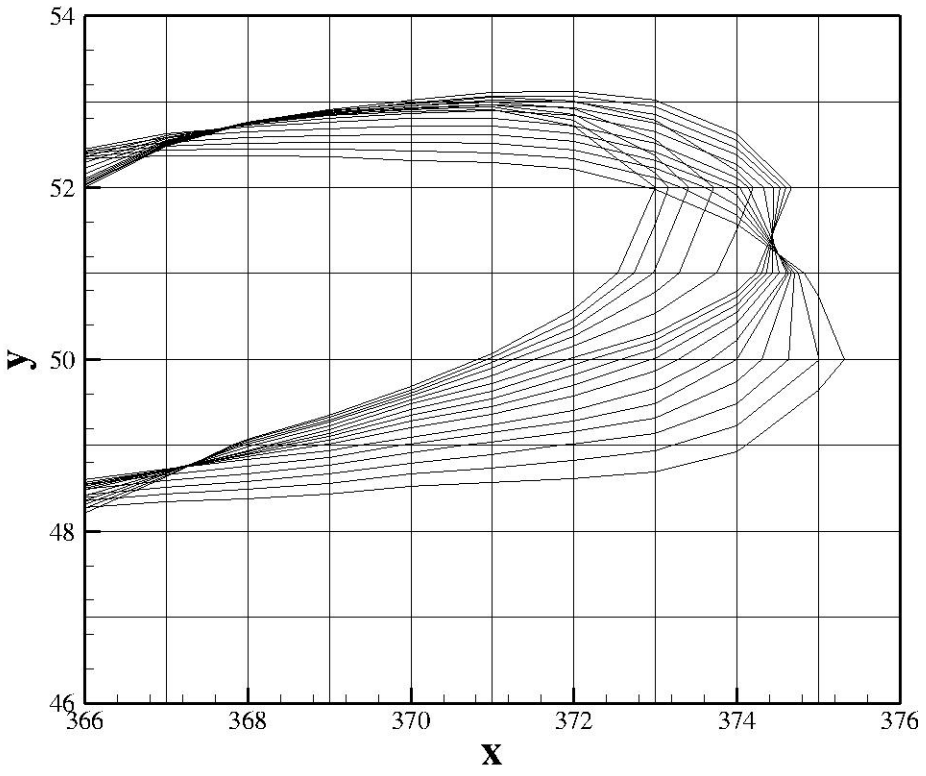


Figure 7-6. A close-up view of the tail tip region being moved with the elastic force model through three image frames.

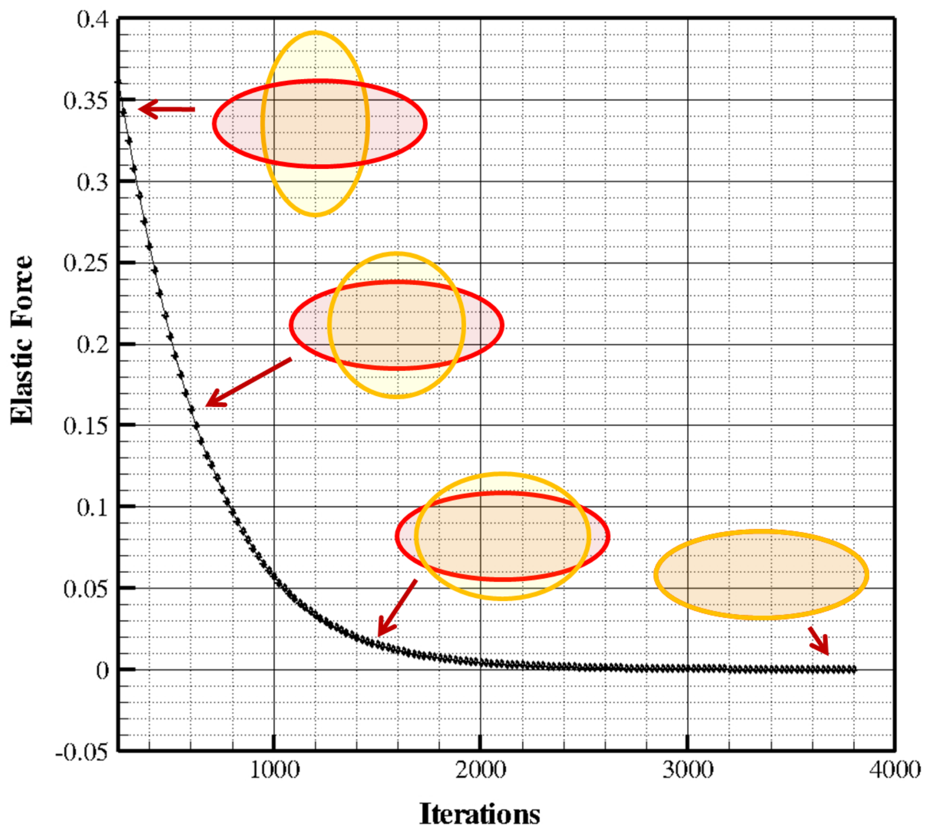


Figure 7-7. Exponential decay of the elastic force through the morphing process required sampling at exponentially increasing time intervals in order to produce a constant morphing motion.

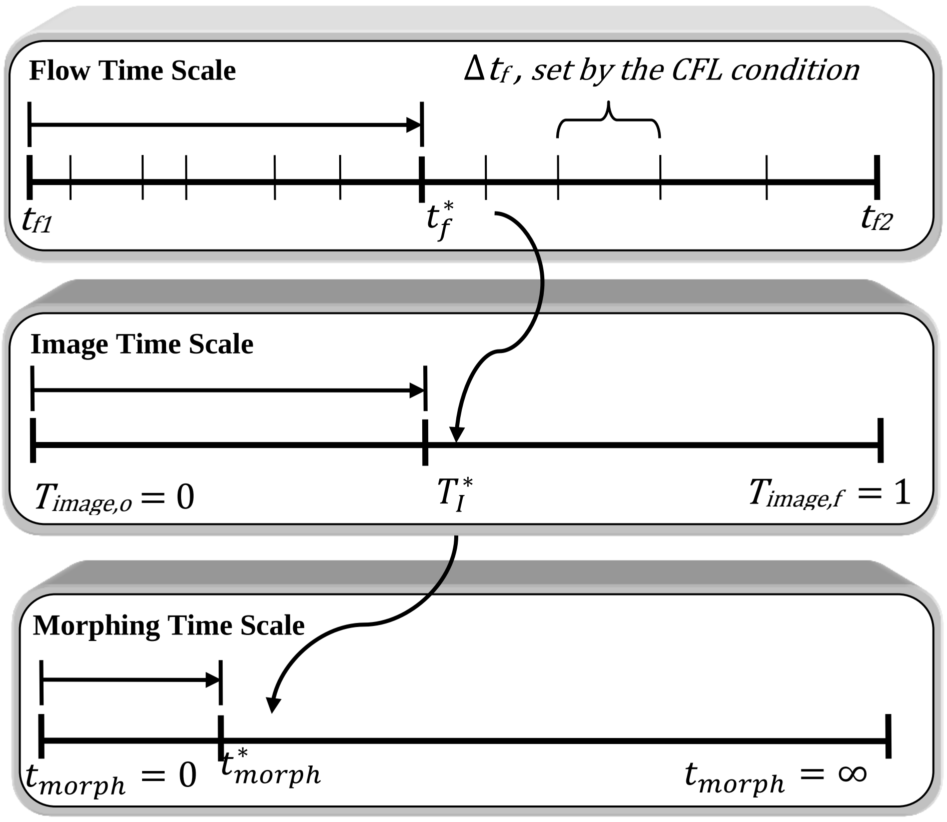


Figure 7-8. Position in the flow time scale is cast as a fraction between two ordered frames, which in turn gives the number of morphing steps required to update the image to a new position according to the root-mean-squared distance metric.

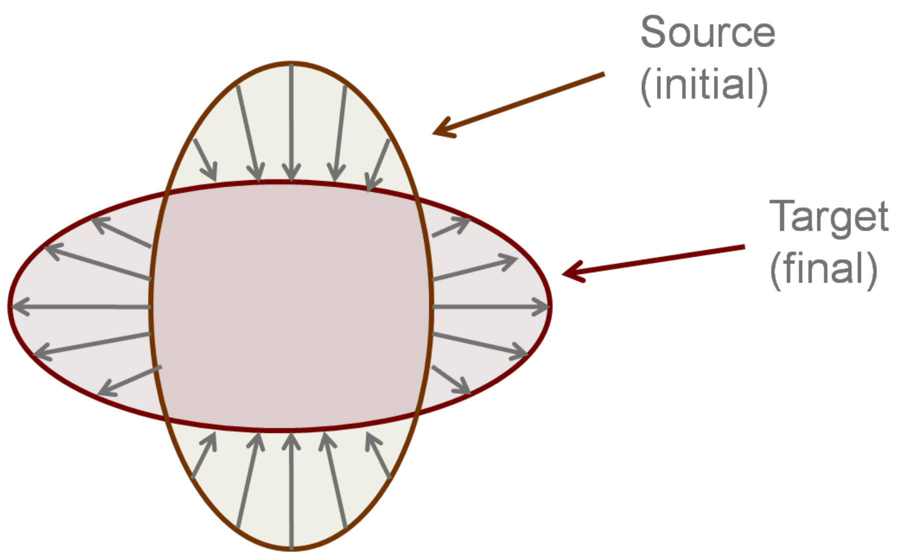


Figure 7-9. The force generated by the elastic model is directly proportional to the distance separating the initial and final contour configurations, resulting in a non-uniform blend from one state to the next (for instance, no motion occurs where the contours cross).

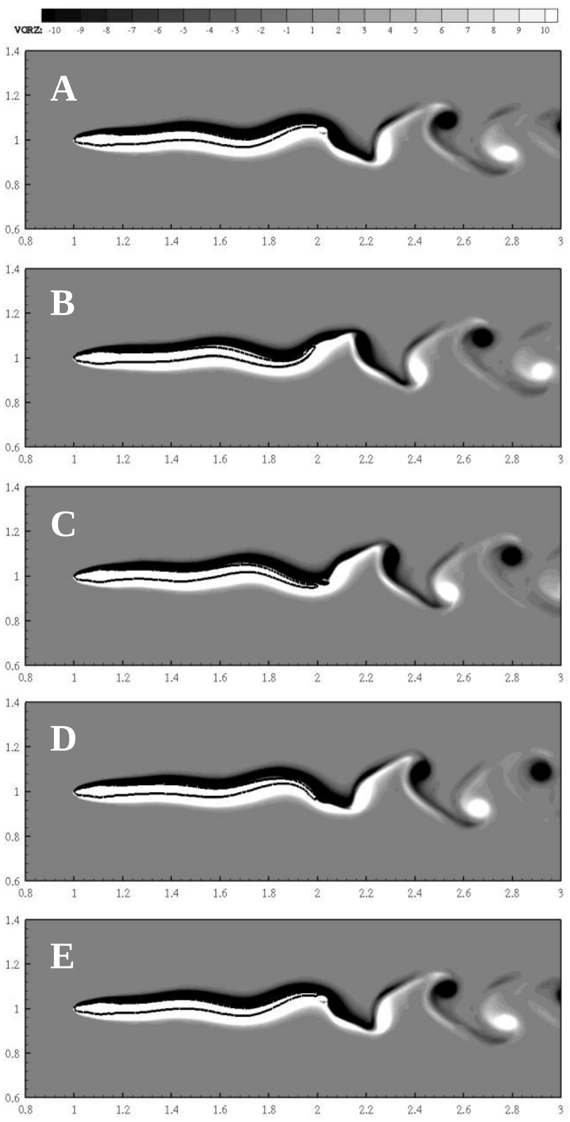


Figure 7-10. Contours of vorticity plotted through one complete tail beat; St = 0.3 (Eulerian method).

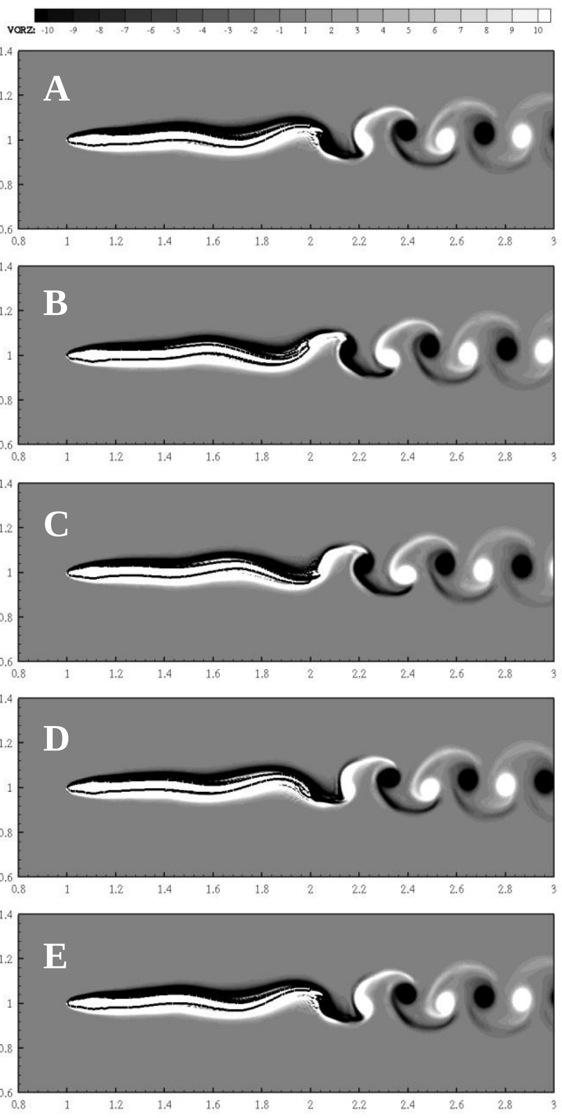


Figure 7-11. Contours of vorticity plotted through one complete tail beat; St = 0.5 (Eulerian method).

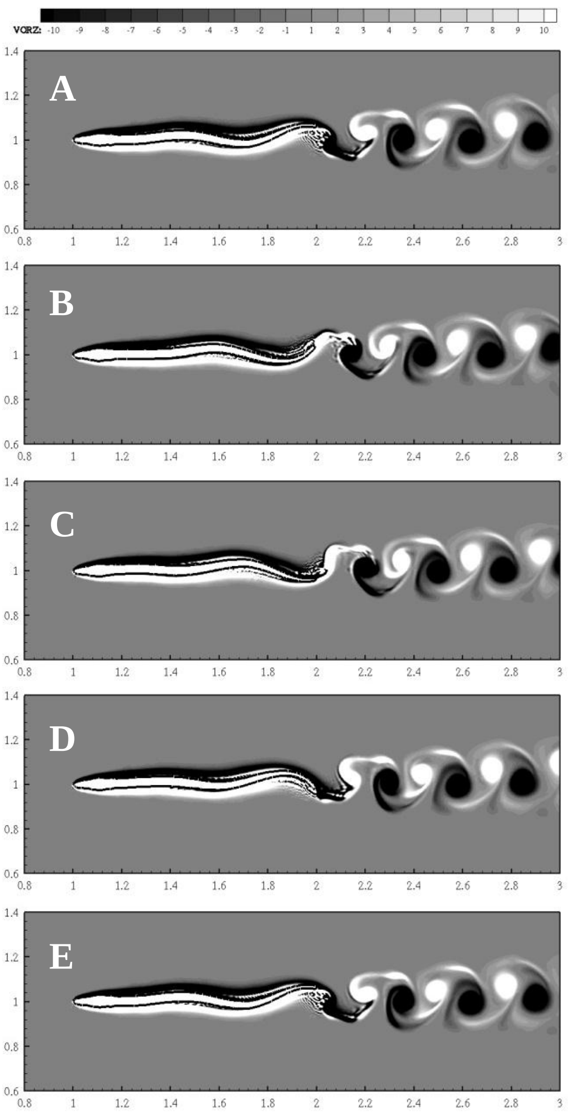


Figure 7-12. Contours of vorticity plotted through one complete tail beat; St = 0.7 (Eulerian method).

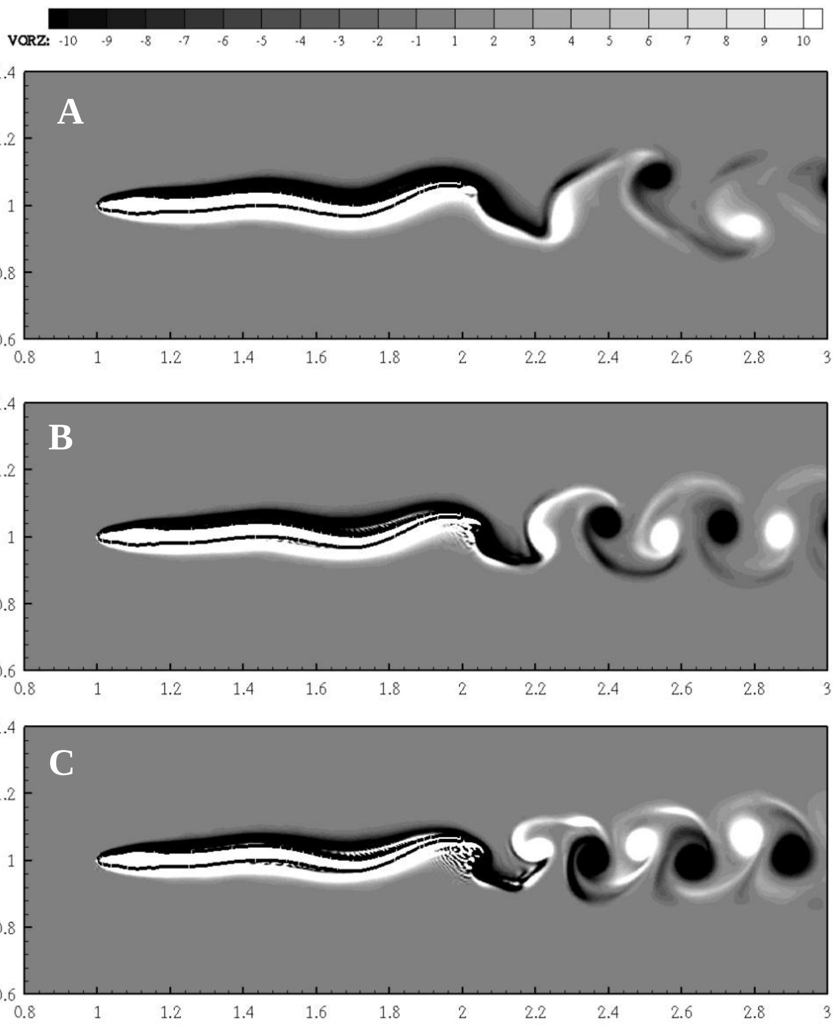


Figure 7-13. Contours of vorticity: (A) St = 0.3; (B) St = 0.5; (C) St = 0.7 (Eulerian method).

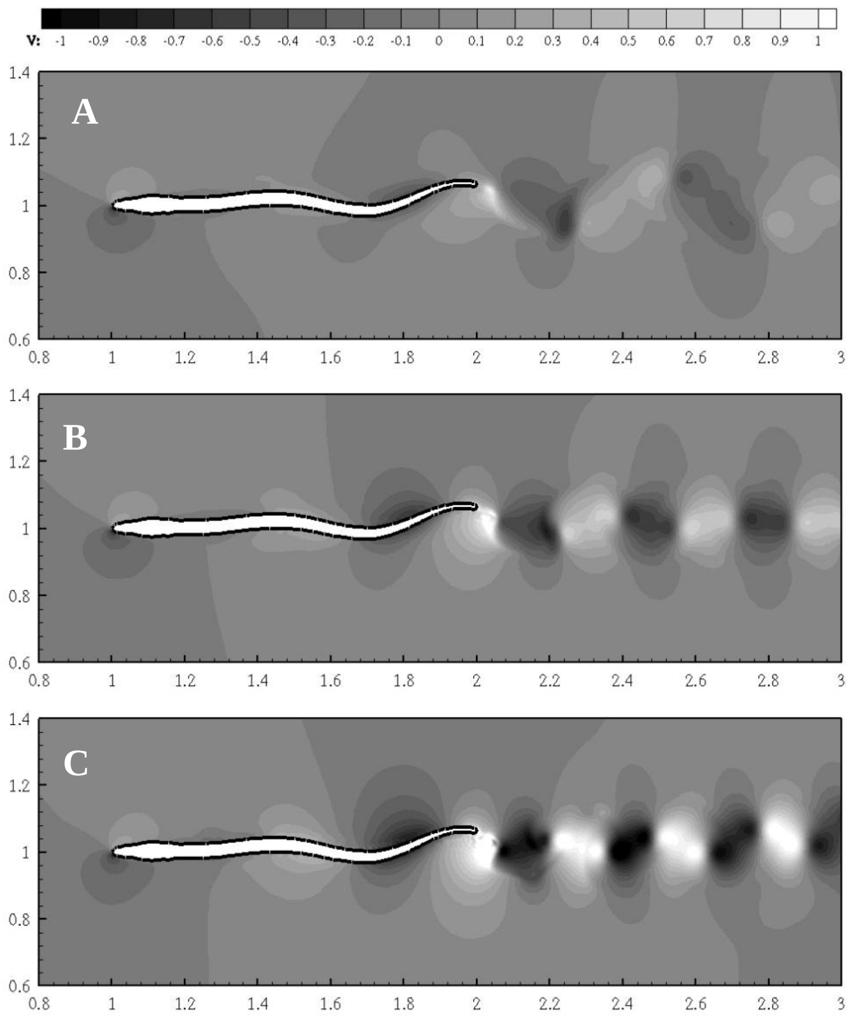


Figure 7-14. Contours of y-velocity: (A) St = 0.3; (B) St = 0.5; (C) St = 0.7 (Eulerian method).

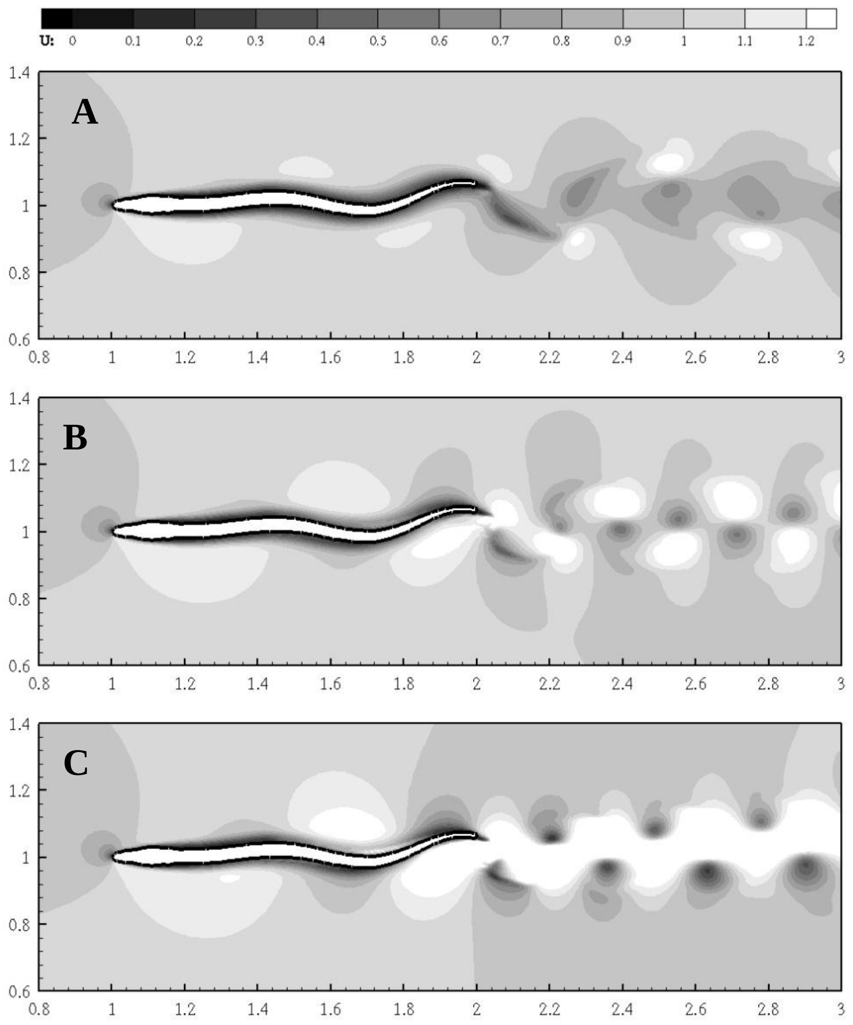


Figure 7-15. Contours of x-velocity: (A) St = 0.3; (B) St = 0.5; (C) St = 0.7 (Eulerian method).

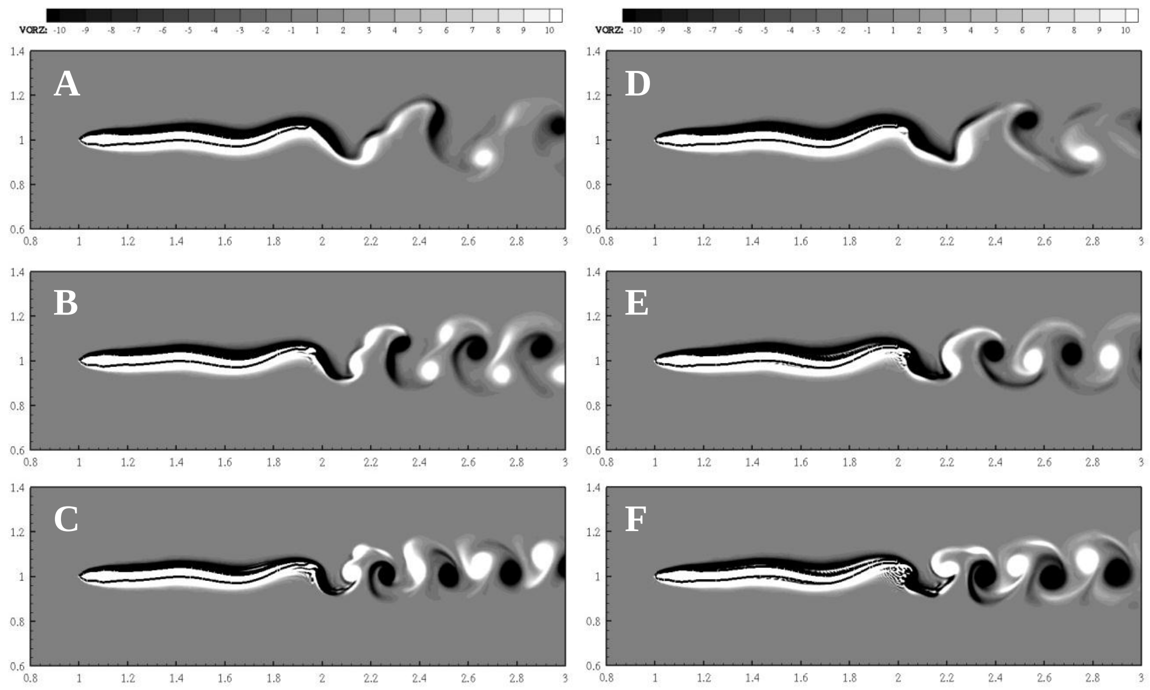
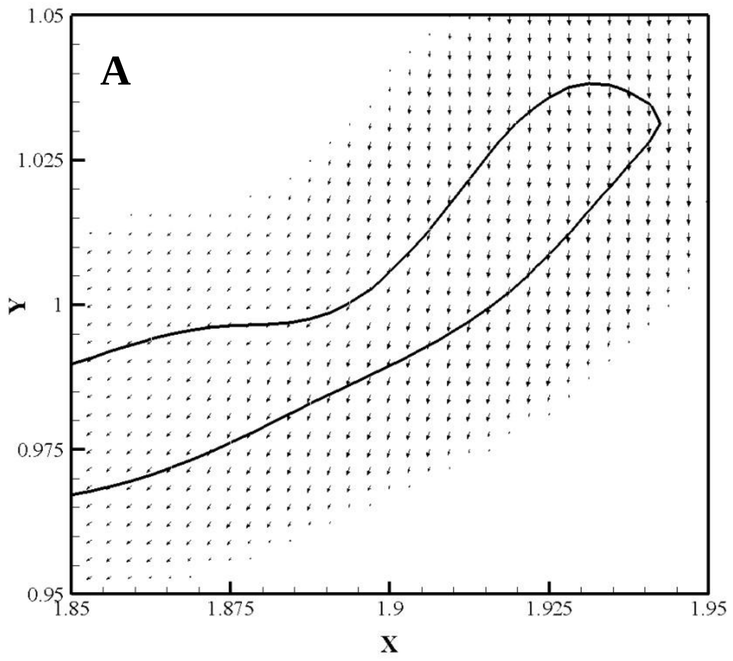


Figure 7-16. Wake structures produced at each Strouhal number by the original Lagrangian method (A-C) and the new Eulerian method (D-F) differed considerably, due to differences in boundary conditions.



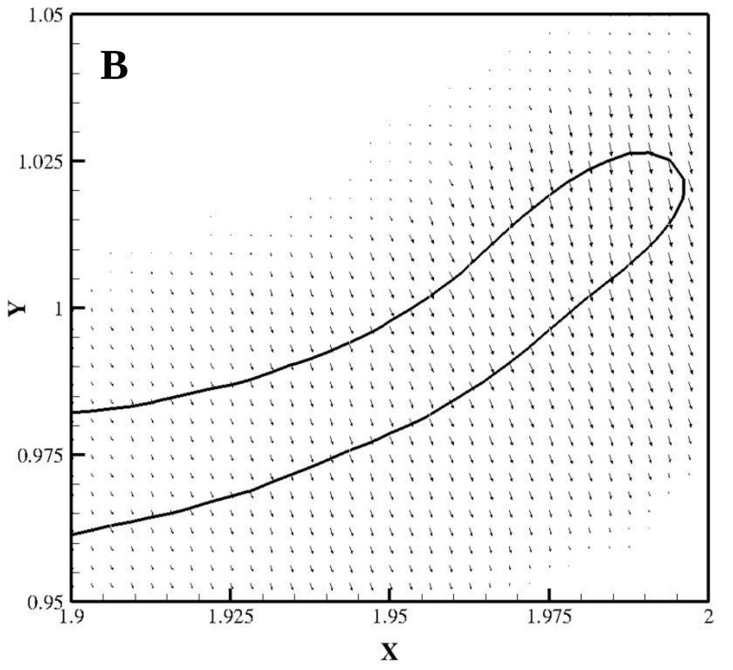
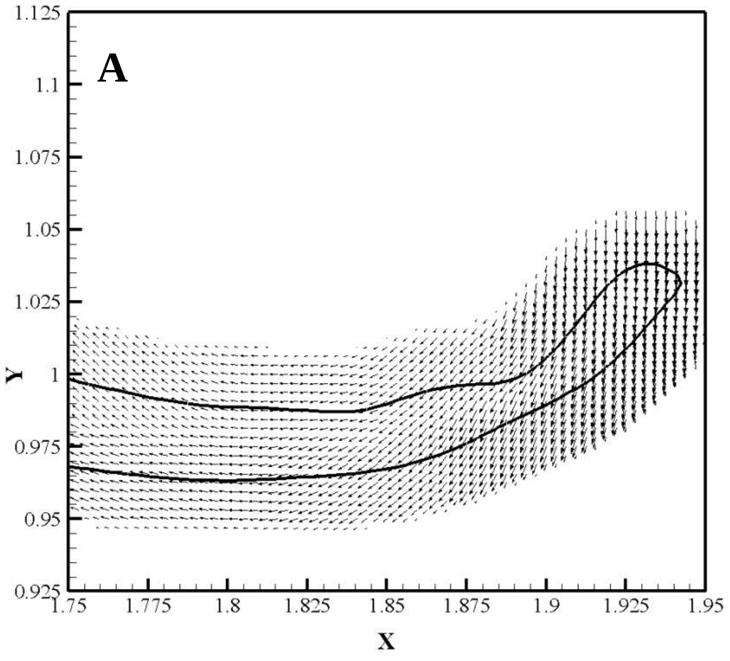


Figure 7-17. The surface geometry of the eel’s tail was modeled much more accurately by the new Eulerian method (B) than by the Lagrangian method (B). Interface velocity vectors are quite different between the two cases, appearing to be correct in (B) and unphysical in (A).



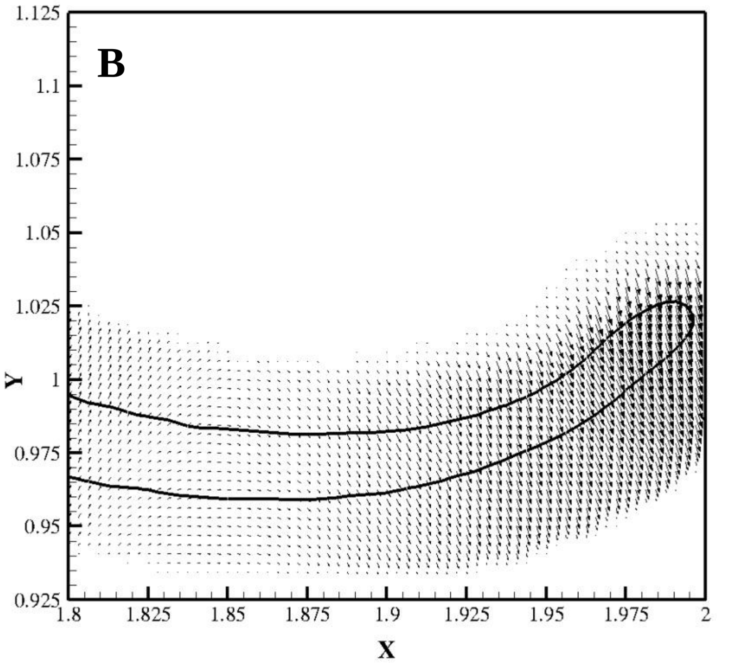


Figure 7-18. Surface velocity vectors along the eel's tail region produced by the Lagrangian method (A) and the Eulerian method (B); the vectors in (B) are in close agreement with observed motion, while the vectors in (A) make no physical sense.

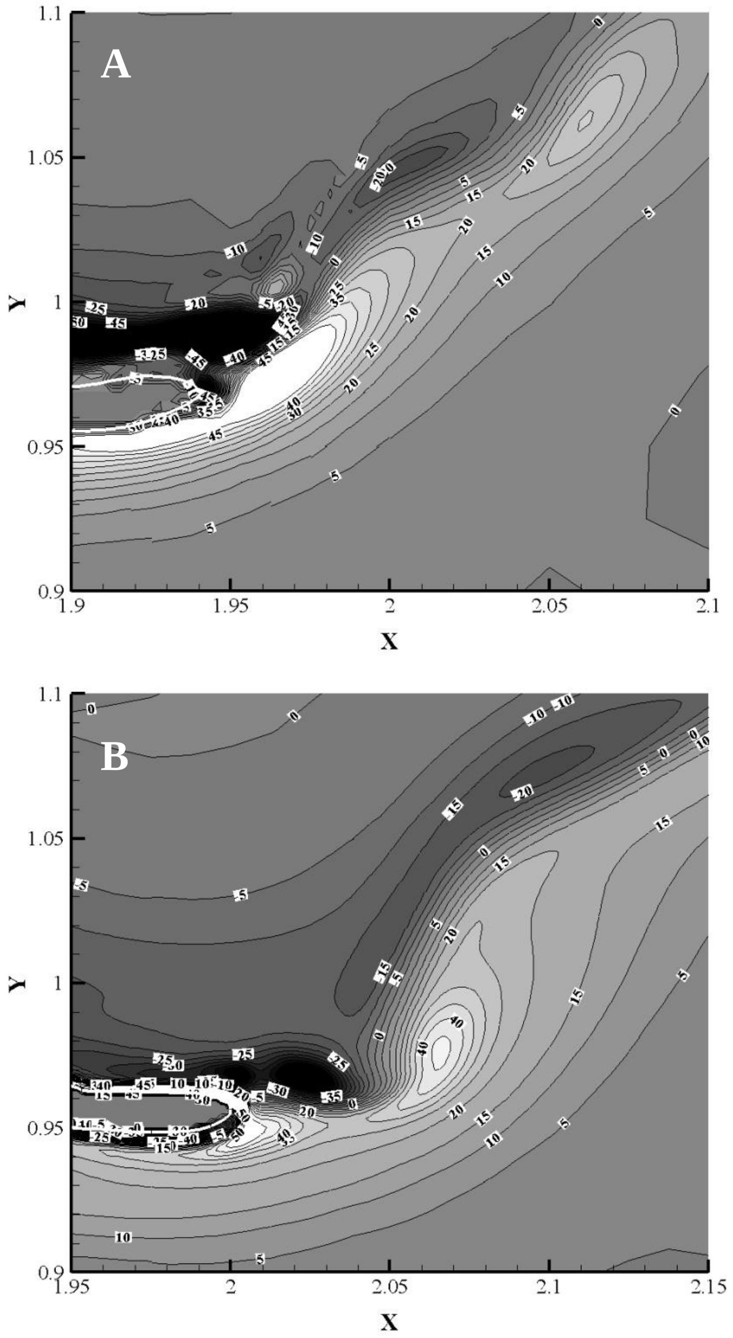


Figure 7-19. Plots of vorticity shed from the eel’s tail, produced by the Lagrangian method (A) and the Eulerian method (B); the vorticity magnitude is higher at the tail in (A), due to the presence of unphysical surface vectors increasing shear in that case, and thus the two cases produce wake structures that are quite different.

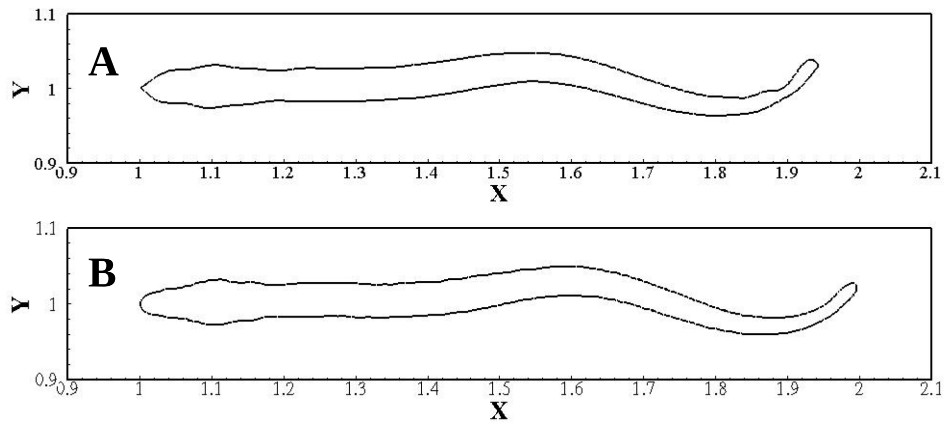
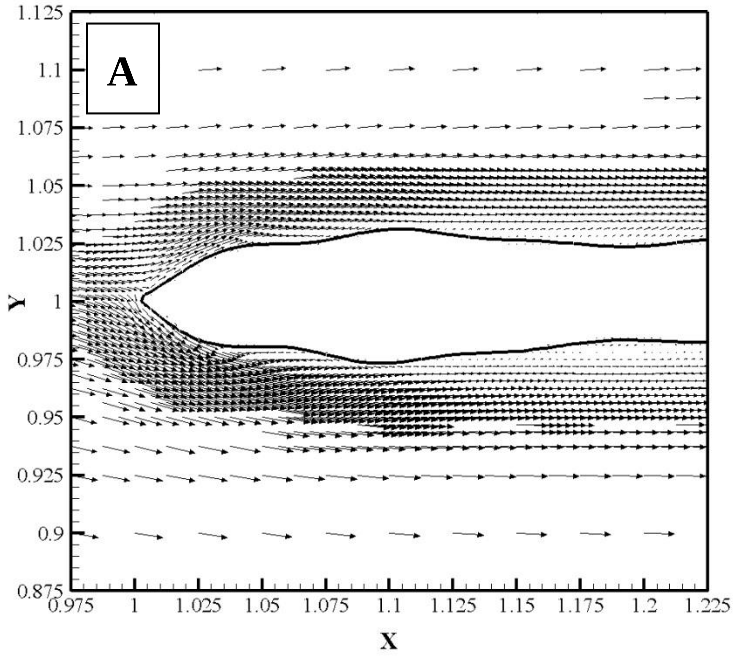


Figure 7-20. A comparison of the eel surface geometries produced by the Lagrangian method (A) and the Eulerian method (B) clearly shows the greater fidelity with which the new method is able to represent the imaged surface.



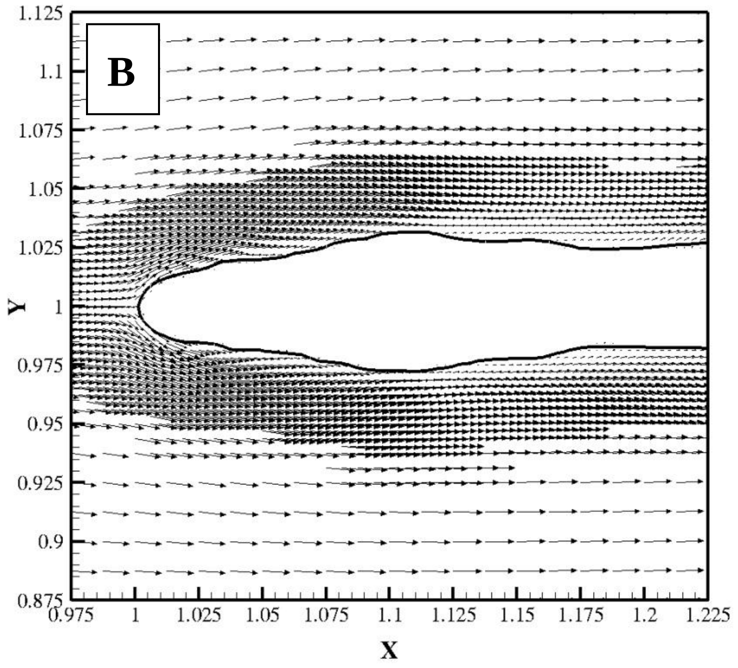


Figure 7-21. The eel's nose modeled with the Lagrangian point-based method (A) and the new Eulerian method (B), again illustrating the superior fidelity maintained by (B) with respect to the eel's imaged geometry. These geometrical differences, along with the incorrect boundary conditions calculated by method (A) produced different surface flow conditions, which show up clearly in the wake structures.

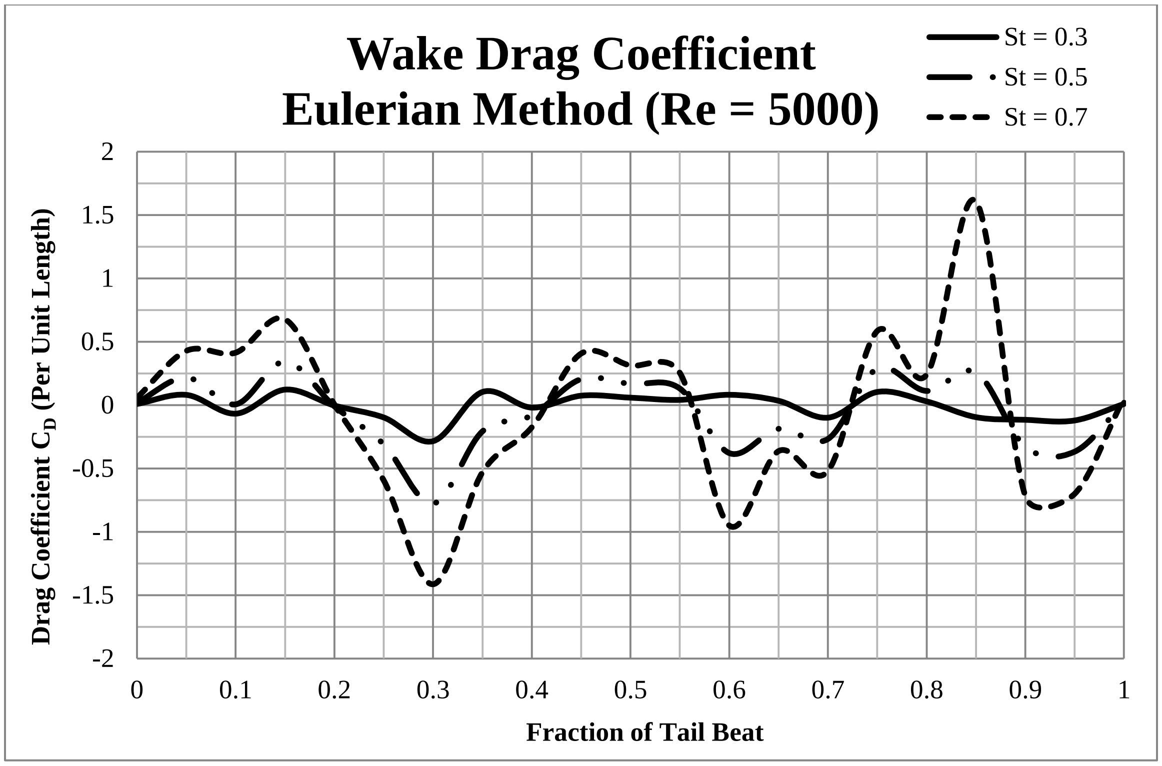


Figure 7-22. Instantaneous drag coefficients measured by control volume analysis during one complete tail beat (Eulerian method).

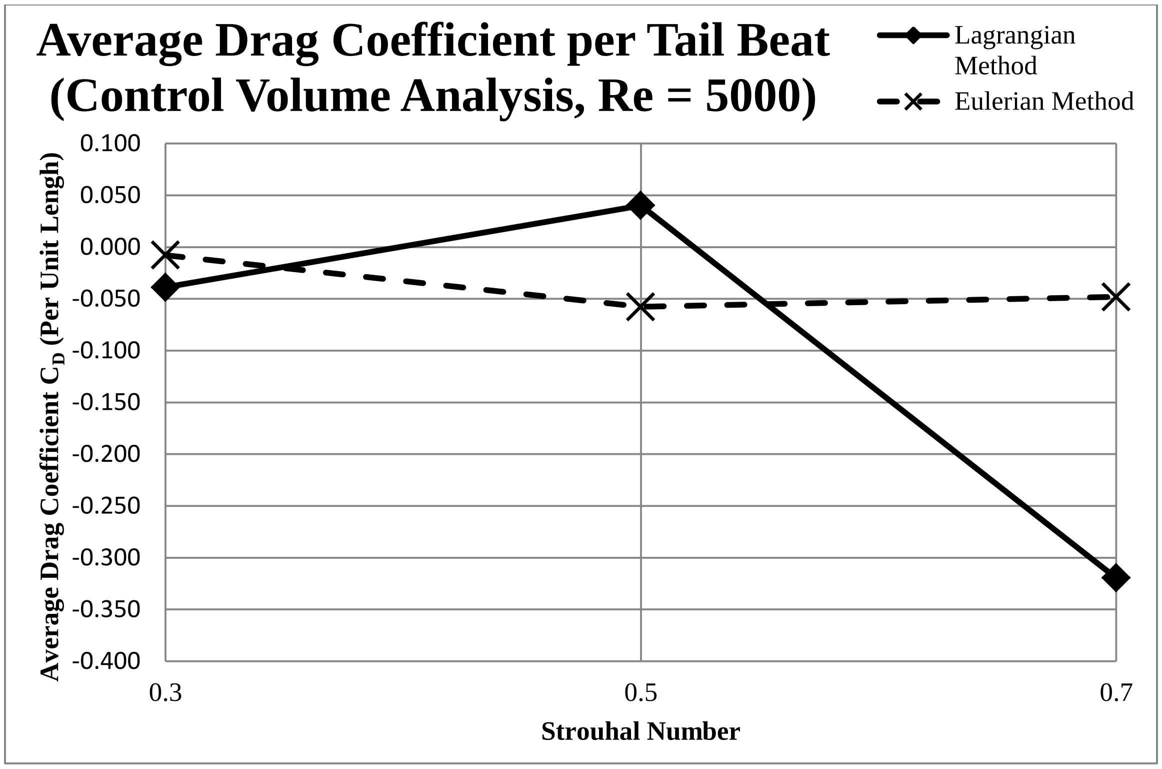


Figure 7-23. A comparison of the average drag coefficients measured by control volume analysis in the Lagrangian and Eulerian models.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 A Summary of the Present Work

Image based modeling is attractive because of its potential to generate accurate descriptions of complex objects in a relatively simple manner. The purpose of a model is to elucidate some phenomenon that occurs in the physical world, with the model's degree of fidelity often dependent upon its degree of complexity.

Geometric models can be difficult to create with mathematical functions, especially when the thing being modeled exhibits highly complicated shapes or patterns of motion. In Chapter 2 of this thesis, mathematically defined moving boundaries were shown to effect highly tortuous fluid flow patterns in some 2-D channel flows, which led to rapid mixing. Mixing is of great interest to engineering and biological sciences, as it is prevalent in many important phenomena such as chemical reactions, pollutant transport, homogenization, heat transfer, digestion, etc. Indeed, the models described in Chapter 2 were inspired by questions about mechanisms observed in the human antro-duodenal junction that have not been answered to complete satisfaction. And, some insightful results were achieved with the use of those simple geometries. However, modeling the human GI tract and its concerted kinematic actions, even over a small portion of it, with fidelity, would require mathematical surface building using functions that are of a much higher order of complexity and with a great deal of built-in uncertainty.

Chapter 3 offered a way around this difficulty by offering the creation of geometric models from images as an alternate route. Imaged objects were used to generate curves describing their outlines, and the curves were populated with a set of Lagrangian points, thus allowing them to be tracked spatially and temporally, completing the physical description. This method was applied to a video sequence of an American eel swimming in an experimental apparatus, and to another video sequence of a guinea pig duodenal segment contracting in vitro., exemplifying both internal and external fluid flow behaviors, respectively. This method turned out to nicely facilitate the generation of these models, which would have otherwise been a difficult task using mathematical functions alone. However, the Lagrangian point-based nature of this method came with its own unique set of problems. One-to-one point correspondence was required to construct surface velocities on the objects, which required equal spacing of the points on the surface curves regardless of how the curves were actually evolving. This had the undesirable effect of introducing unphysical surface velocities, which in turn led to incorrect boundary conditions in the fluid flow environment. Point placement and tracking also turned out to be highly tedious and required compromises between accuracy and reliability, such as limiting the search for interface crossings to a single direction that had to be defined by the user a priori based on image characteristics in order for the point population process to proceed in a robust fashion.

Work outlined in the rest of the chapters was conducted thenceforth with the primary goal of dispensing of the need for Lagrangian points, and thus hopefully averting the multitude of problems encountered with them. Some crude segmentation and denoising techniques were introduced in Chapter 3, and sufficed for the purposes of those models because the point smoothing that followed Lagrangian surface meshing generated acceptable, if not ideal, geometries. However, such surface smoothing would not be possible in the new Eulerian framework. This meant that all image smoothing would have to take place *before* segmentation and the subsequent conversion to a level set surface description. So, in preparation for the transition away from points, denoising and segmentation were revisited in detail in Chapter 4, with a full investigation of denoising techniques provided therein. This allowed for the selection of a suitable image denoising method for our purposes, in turn allowing for the creation of smooth segmentation contours amenable to generating well-behaved level set descriptions of imaged objects.

With object boundaries cast in the level set formulation, an Eulerian description of their locations in space at pixel resolution was effected for each image frame in the sequence, nearly completing a coarsely defined description of the model in space and time. Now the task was to find a way to apply boundary conditions—the final missing piece of a complete physical description—to the modeled surfaces, and to increase the level of spatio-temporal resolution so that they could be coupled with a flow solver for CFD simulations.

To this end, the computer vision technique of optical flow was introduced in Chapter 5. Because optical flow is designed to deduce object motion based on changing brightness patterns in image sequences, it was felt to hold promise in providing the missing Eulerian boundary condition information being sought for the level set representations of segmentation curves discussed in Chapter 4. A survey of optical flow methods reported in the literature over the past 30 years was performed, showing recently developed nonlinear methods to offer a potential for highly accurate descriptions of imaged object motions.

In Chapter 6, the optical flow method algorithms described in Chapter 5 were applied, first to a set of synthetically created test images, then to a set of real images by revisiting the swimming American eel described in Chapter 3, and finally applying the methods to flowing particle fields and PIV imagery. The nonlinear optical flow method proved to be quite promising with respect to its ability to produce accurate boundary motions in an Eulerian setting; the errors produced by it were quite low in the majority of test cases, particularly near object boundaries where a small error was of greatest importance. However, attempts to advect the eel image from one frame in the sequence to another under the influence of optical flow vectors ultimately failed, despite the seemingly high accuracy of the optical flow solution. This was found to be a problem not with the optical flow field itself, but with the assumption that optical flow velocities project onto image gradients when invoking the standard advection method. Thus, optical flow vectors were retained as a means to supply boundary conditions, but an alternative method needed to be developed for effecting imaged object motion at a sub-frame-rate scale.

Particle and PIV results were mixed, with the nonlinear optical flow method giving feasible vector fields in some cases, but completely unrealistic results in others. In particular, image sequences fraught with large average intensity variations between frames, or with large radial intensity gradients within frames, gave poor results that could not compete with existing standard PIV methods. However, a significant amount of work remains to be done with regard to tuning optical flow parameters to match the demands of individual image sequences, as well as investigating the inclusion of physically significant effects in the constraint equations applied to the optical flow minimization functional.

Because of the problems encountered when trying to move boundaries using optical flow, Chapter 7 began with a discussion of image morphing, giving an overview of level set-based morphing methods that were felt to match well with the framework being developed here. In the end, a morphing method based on elastic force modeling was decided upon, as it gave the best results out of the methods tested in the form of smooth, consistent interface motion. Now the facility was in place for describing motion on arbitrarily fine time intervals, even when presented with relatively coarse image acquisition rates in video sequences.

The remainder of Chapter 7 was concerned with spatially mapping level set representations of imaged objects onto flow meshes of arbitrary scale, and simulating fluid flow around them. Bilinear interpolation was used to map level set information from a morphed image onto its corresponding CFD mesh, and optical flow vectors were mapped onto the resulting immersed interface in order to supply boundary conditions to the flow. Applied to the swimming American eel, the results turned out to be quite different from those obtained using the Lagrangian method outlined in Chapter 3; wake structures were markedly different between the two cases. The greatest contribution to this disparity was likely the large difference in the quality of surface velocity information. While the Lagrangian method consistently gave unphysical surface vectors resulting from the continuous redistribution of points to maintain connectivity, the Eulerian method relied on the displacement of brightness patterns to describe surface motion. Optical flow vectors were in agreement with observed motion, whereas Lagrangian surface vectors generally did not correlate with observed motion at all.

8.2 Scope of Future Work and Possible Extensions

While the overall goals envisioned at the start of this thesis work were achieved, there is much left to be investigated along the paths set forth; many more questions arose along the way. Denoising methods deserve a closer look in the way of further examining the optimization of parameters involved with methods such as SRAD. Wavelet based denoising methods also deserve a great deal more investigation, through the use of different wavelet types, and perhaps through different methods of filtering and diffusing in wavelet space.

The nonlinear optical flow techniques employed here were found to work quite well, but coupling them with level set segments along the lines of Amiaz and Kiryati [9] would probably improve accuracy further, and would not require much extension beyond what is already in place. Even the algorithm based on the work of Brox et al. that is already in place might be improved through one-way coupling with the image segment; the optical flow field could be solved independently inside and outside of the segmentation curve, possibly improving the quality of the solution in both places.

Although the particle results were mediocre, only a small range of tunable parameters was investigated to get the optical flow solutions provided herein, and so it is likely that a great deal of improvement could be made in the direction of particle vector quantification. While the large amount of user input required for optimizing optical flow for PIV images in this way may appear to make the method less attractive than standard correlation-based methods already well established, optical flow does offer two major advantages over standard PIV algorithms. First, unlike correlation-based methods, which use-multiple pixel windows to obtain vector fields that are significantly coarser than image resolution, optical flow provides a vector for each pixel in an image sequence. This can be of great use when trying to quantify wall shear flows or other phenomena that are marked by large velocity gradients. Second, optical flow has the ability to quantify motion everywhere in an image domain, including regions where there are no particles. This opens up the possibility of quantifying fluid interactions with moving boundaries, as both surface motion and particle motion can be tracked simply by finding the displacements of imaged brightness patterns. Of course, as was seen, this requires high-quality images that are free of large overall intensity variations in order to produce a reliable result. However, introducing physically meaningful metrics into the constraint equations may well provide the robustness required to overcome this present limitation in the method.

One of the most promising outlooks for the methodology outlined here lies in the realm of 4-D image based modeling. For example, 3-D ultrasound data sequences in time could be denoised and segmented using the methods described in Chapter 4, then a 3-D optical flow field could be calculated between the resulting smooth image data to generate a complete physical description of moving level set surfaces. Such tools would be powerful, indeed, and with a little extension to this work and given enough computational power, should be possible to achieve in relatively short order.

8.3 Contributions of the Present Thesis Work

Work performed for this thesis represents, to our knowledge, the first of its kind, coupling the computer vision methodologies of denoising, segmentation, optical flow, and morphing with a level set framework to provide a seamless transition from video images to CFD simulations—all within an Eulerian setting that completely dispenses with point placement and surface meshing. The facility to model complex moving boundaries in a simple, straightforward fashion has thus been established. Ongoing work in the direction of extending these methods to 3-D (still) and 4-D (moving) data sets will allow for modeling highly complex surfaces, such as organ systems or microstructures imaged via X-ray CT or ultrasound, with a level of detail that might not be possible to achieve any other way. Thus, the work presented in this thesis holds great potential for application to a wide array of problems in biofluid mechanics, hydrodynamics with moving boundaries (as in swimming and flapping flight) and computation in the presence of complex imaged microstructures.

REFERENCES

1. Simpson, J.A., E.S.C. Weiner, and Oxford University Press., *The Oxford English Dictionary*. 2nd ed. 1989, Oxford ; New York: Clarendon Press ;Oxford University Press.

2. Lighthill, J., *Mathematical Biofluiddynamics*. 1975, Philadelphia: Society for Industrial and Applied Mathematics.

3. Thompson, D., *On Growth and Form: A New Edition*. 1942, Cambridge: Cambridge University Press.

4. Chandran, K., A. Yoganathan, and S. Rittgers, *Biofluid Mechanics: The Human Circulation*. 2007, Boca Raton: CRC Press.

5. Schulze, K., *Imaging and Modelling of Digestion in the Stomach and the Duodenum.* Neurogastroenterology and Motility, 2006. **18**: p. 172-183.

6. Adkins, D. and Y.Y. Yan, *CFD Simulation of Fish-like Body Moving in Viscous Liquid.* Journal of Bionic Engineering, 2006. **3**: p. 147-153.

7. Liu, H., *Simulation-based Biological Fluid Dynamics in Animal Locomotion.* Applied Mechanics Reviews, 2005. **58**: p. 269-282.

8. Alvino, C., et al., *Multigrid Computation of Rotationally Invariant Non-Linear Optical Flow*, in *IEEE*. 2005.

9. Amiaz, T. and N. Kiryati, *Piecewise Smooth Dense Optical Flow via Level Sets.* International Journal of Computer Vision, 2006. **68**(2): p. 111-124.

10. Black, M. and P. Anandan, *The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields.* Computer Vision and Image Understanding, 1996. **63**(1): p. 75-104.

11. Borshukov, G., et al., *Motion Segmentation by Multistage Affine Classification.* IEEE Transactions on Image Processing, 1997. **6**(11): p. 1591-1594.

12. Brox, T., et al. *High Accuracy Optical Flow Estimation Based on a Theory for Warping*. in *8th European Conference on Computer Vision*. 2004.

13. Chan, T. and L. Vese, *Active Contours Without Edges.* IEEE Transactions on Image Processing, 2001. **10**(2): p. 266-277.

14. Ha, J., et al. *Active Contours and Optical Flow for Automatic Tracking of Flying Vehicles*. in *2004 American Control Conference*. 2004.

15. Horn, B. and B. Schunck, *Determining Optical Flow.* Artificial Intelligence, 1981. **17**: p. 185-203.

16. Jhunjhunwala, P. and S. Rajagopalan. *Optical Flow based Volumetric Spatio-Temporal Interpolation*. in *30th Annual International IEEE EMBS Conference*. 2008.

17. Kichenassamy, S., et al., *Conformal Curvature Flows: From Phase Transitions to Active Vision.* Archive for Rational Mechanics and Analysis, 1996. **134**: p. 275-301.

18. Gibou, F. and R. Fedkiw. *A Fast Hybrid k-Means Level Set Algorithm for Segmentation*. in *4th Annual Hawaii International Conference on Statistics, Mathematics, and Related Fields*. 2005.

19. Chan, T. and S. Esedoglu. *A Multiscale Algorithm for Mumford-Shah Image Segmentation*.   [cited 2009 July]; Available from: ftp://ftp.math.ucla.edu/pub/camreport/cam03-77.pdf.

20. Pitas, I., *Digital image processing algorithms and applications*. 2000, New York: Wiley. 419 p.

21. Newland, D.E., *An introduction to random vibrations, spectral and wavelet analysis*. 3rd ed. 1993, Harlow, Essex, England; New York: Longman Scientific & Technical ;Wiley. xxix, 477 p.

22. Donoho, D.L. and I.M. Johnstone, *Ideal Spatial Adaptation by Wavelet Shrinkage.* Biometrika, 1994. **81**(3): p. 425-455.

23. Strang, G. and T.Q. Nguyen, *Wavelets and Filter Banks*. 1996, Wellesley: Wellesley-Cambridge Press. 490.

24. Sethian, J., *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. 1999, New York: Cambridge University Press.

25. Osher, S. and J. Sethian, *Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations.* Journal of Computational Physics, 1988. **79**(1): p. 12-49.

26. Liu, X., R. Fedkiw, and M. Kang, *A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains.* Journal of Computational Physics, 2000. **160**(1): p. 151-178.

27. Ye, T., et al., *An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries.* Journal of Computational Physics, 1999. **156**(2): p. 209-240.

28. Zang, Y., R. Street, and J. Koseff, *A Non-Staggered Grid, Fractional Step Method for Time-Dependent Incompressible Navier-Stokes Equations in Curvilinear Coordinates.* Journal of Computational Physics, 1994. **114**(1): p. 18-33.

29. Liu, H., et al., *Sharp Interface Cartesian Grid Method II: A Technique for Simulating Droplet Interactions with Surfaces of Arbitrary Shape.* Journal of Computational Physics, 2005. **210**: p. 32-54.

30. Sethian, J., *Evolution, Implementation, and Application of Level Set and Fast Marching Methods for Advancing Fronts.* Journal of Computational Physics, 2001. **169**(2): p. 503-555.

31. Sethian, J. and P. Smareka, *Level Set Methods for Fluid Interfaces.* Annual Review of Fluid Mechanics, 2003. **35**: p. 341-372.

32. Sussman, M. and E. Fatemi, *An Efficient, Interface-Preserving Level Set Redistancing Algorithm and its Application to Interfacial Incompressible Fluid Flow.* SIAM Journal on Scientific Computing, 1999. **20**(4): p. 1165-1191.

33. Sussman, M., et al., *An Improved Level Set Method for Incompressible Two-Phase Flows.* Computers & Fluids, 1998. **27**(5-6): p. 663-680.

34. Fadlun, E., et al., *Combined Immersed Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations.* Journal of Computational Physics, 2000. **161**(1): p. 35-60.

35. Kim, J., D. Kim, and H. Choi, *An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries.* Journal of Computational Physics, 2001. **171**(1): p. 132-150.

36. Glowinski, R., et al., *A Fictitious Domain Approach to the Direct Numerical Simulation of Incompressible Viscous Flow past Moving Rigid Bodies: Application to Particulate Flow.* Journal of Computational Physics, 1999. **169**(2): p. 363-426.

37. Udaykumar, H., R. Mittal, and P. Rampunggoon, *Interface Tracking Finite Volume Method for Complex Solid-Fluid Interactions on Fixed Meshes.* Communications in Numerical Methods in Engineering, 2002. **18**(2): p. 89-97.

38. Udaykumar, H., et al., *A Sharp Interface Cartesian Grid Method for Simulating Flows with Complex Moving Boundaries.* Journal of Computational Physics, 2001. **174**(1): p. 345-380.

39. Leveque, R. and Z. Li, *The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources.* SIAM Journal on Numerical Analysis, 1994. **31**(4): p. 1019-1044.

40. Leveque, R. and Z. Li, *The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources* SIAM Journal on Numerical Analysis, 1995. **32**(5): p. 1704-1704.

41. Brackbill, J., D. Kothe, and C. Zemach, *A Continuum Method for Modeling Surface Tension.* Journal of Computational Physics, 1991. **100**(2): p. 335-354.

42. Anderson, D., G. McFadden, and A. Wheeler, *A Phase-Field Model of Solidification with Convection.* Physica D, 2000. **135**(1-2): p. 175-194.

43. Marella, S., et al., *Sharp Interface Cartesian Grid Method I: An Easily Implemented Technique for 3D Moving Boundary Computations.* Journal of Computational Physics, 2005. **210**: p. 1-31.

44. Yang, Y. and H. Udaykumar, *Sharp Interface Cartesian Grid Method III: Solidification of Pure Materials and Binary Solutions.* Journal of Computational Physics, 2005. **210**: p. 55-74.

45. Ramkumar, D. and K. Schulze, *The Pylorus.* Neurogastroenterology and Motility, 2005. **17**: p. 22-30.

46. Schulze-Delrieu, K., H. Ehrlein, and A. Blum, *Mechanics of the Pylorus*, in *Gastric and Gastroduodenal Motility*. 1984, Praeger Publishers. p. 87-102.

47. Malbert, C. and Y. Ruckebusch, *Passage of Chyme and Contractile Patterns at the Antro-Duodenal Junction in the Dog*, in *Gastro-Pyloro-Duodenal Coordination*. 1990, Wrightson Biomedical Publishing Ltd. p. 197-208.

48. Schulze-Delrieu, K. and C. Brown, *Emptying of Saline Meals by the Cat Stomach as a Function of Pyloric Resistance.* American Journal of Physiology (Gastrointestinal and Liver Physiology), 1985. **249**(12): p. G725-G732.

49. Gibou, F., et al., *A level set approach for the numerical simulation of dendritic growth.* Journal of Scientific Computation, 2003. **19**: p. 183-199.

50. Horner, M., et al., *Transport Enhancement Mechanisms in Open Cavities.* Journal of Fluid Mechanics, 2002. **452**: p. 199-229.

51. Howes, T. and P. Shardlow, *Simulation of Mixing in Unsteady Flow through a Periodically Obstructed Channel.* Chemical Engineering Science, 1997. **52**(7): p. 1215-1225.

52. Ottino, J., *The Kinematics of Mixing: Stretching, Chaos, and Transport*. 1989, New York: Cambridge University Press.

53. Krishnan, S., S. Marella, and H. Udaykumar, *Sharp Interface Cartesian Grid Method IV: Local Mesh Refinement*. 2006.

54. Mackley, M. and R. Veves Saraiva, *The Quantitative Description of Fluid Mixing using Lagrangian and Concentration-Based Numerical Approaches.* Chemical Engineering Science, 1999. **54**: p. 159-170.

55. Jeffrey, B., H. Udaykumar, and K. Schulze, *Flow Fields Generated by Peristaltic Reflex in Isolated Guinea Pig Ileum: Impact of Contraction Depth and Shoulders.* American Journal of Physiology (Gastrointestinal and Liver Physiology), 2003. **285**: p. 907-918.

56. Drazin, P., *Nonlinear Systems*. 1992, New York: Cambridge University Press.

57. Baker, G. and J. Gollub, *Chaotic Dynamics*. 2nd Edition ed. 1996, New York: Cambridge University Press.

58. Tytell, E. and G. Lauder, *The Hydrodynamics of Eel Swimming I: Wake Structure.* Journal of Experimental Biology, 2004. **207**: p. 1825-1841.

59. Press, W., et al., *Numerical Recipes in Fortran 77: The Art of Scientific Computing*. 2nd Edition ed. 1992, New York: Cambridge University Press.

60. Press, W., et al., *Numerical Recipes in Fortran 90: The Art of Parallel Scientific Computing*. 1999, New York: Cambridge University Press.

61. Percival, D. and A. Walden, *Wavelet Methods for Time Series Analysis.* . 2000, New York: Cambridge University Press.

62. Resnikoff, H. and R. Wells, *Wavelet Analysis: The Scalable Structure of Information*. 1998, New York: Springer-Verlag.

63. Walker, J., *A Primer on Wavelets and their Scientific Applications*. 2000, Boca Raton: CRC Press.

64. Drucker, E. and G. Lauder, *Locomotor Forces on a Swimming Fish: Three-Dimensional Vortex Wake Dynamics Quantified using Digital Particle Image Velocimetry*. Journal of Experimental Biology, 1999. **202**: p. 2393-2412.

65. Drucker, E. and G. Lauder, *Experimental Hydrodynamics of Fish Locomotion: Functional Insights from Wake Visualization.* International Society for Computational Biology, 2002. **42**: p. 243-257.

66. Fish, F. and G. Lauder, *Passive and Active Flow Control by Swimming Fishes and Mammals.* Annual Review of Fluid Mechanics, 2006. **38**: p. 193-224.

67. Shen, L., et al., *Turbulent Flow over a Flexible Wall Undergoing a Streamwise Travelling Wave Motion.* Journal of Fluid Mechanics, 2003. **484**: p. 197-221.

68. Buchholz, J.H.J. and A.J. Smits, *The wake structure and thrust performance of a rigid low-aspect-ratio pitching panel.* Journal of Fluid Mechanics, 2008. **603**: p. 331-365.

69. Jones, K.D., C.M. Dohring, and M.F. Platzer, *Experimental and computational investigation of the Knoller-Betz effect.* Aiaa Journal, 1998. **36**(7): p. 1240-1246.

70. Netter, F., *Atlas of Human Anatomy*. 2nd Edition ed. 1997, East Hanover: Novartis.

71. Russ, J.C., *The image processing handbook*. 5th ed. 2007, Boca Raton: CRC/Taylor and Francis. 817 p.

72. Black, M.J., et al., *Robust anisotropic diffusion.* IEEE Transactions on Image Processing, 1998. **7**(3): p. 421-432.

73. Zhong, J.M. and H.F. Sun, *Wavelet-Based Multiscale Anisotropic Diffusion With Adaptive Statistical Analysis for Image Restoration.* Ieee Transactions on Circuits and Systems I-Regular Papers, 2008. **55**(9): p. 2716-2725.

74. *Philips Ultrasound Image Library*. Available from: http://www3.medical.philips.com/en-us/secure/images_site/largeImage.asp?size=blowup&classcode=03&appcode=a&imagename=0084-HD11-C5-2-ABD&systemcode=c&div=ultra.

75. Greenwood, D., *Classical Dynamics*. 1977, New York: Dover Publcations.

76. Lanczos, C., *The Variational Principles of Mechanics*. 4th Edition ed. 1970, New York: Dover Publications.

77. Strang, G., *Linear Algebra and its Applications*. 4th Edition ed. 2006, Belmont: Thomson Brooks/Cole.

78. Osher, S. and J.A. Sethian, *Fronts Propagating with Curvature-Dependent Speed - Algorithms Based on Hamilton-Jacobi Formulations.* Journal of Computational Physics, 1988. **79**(1): p. 12-49.

79. Raffel, M., C.E. Willert, and J. Kompenhans, *Particle image velocimetry : a practical guide*. Experimental fluid mechanics. 1998, Berlin ; New York: Springer. xvi, 253 p.

80. Corpetti, T., E. Memin, and P. Perez, *Dense estimation of fluid flows.* Ieee Transactions on Pattern Analysis and Machine Intelligence, 2002. **24**(3): p. 365-380.

81. Corpetti, T., et al., *Fluid experimental flow estimation based on an optical-flow scheme.* Experiments in Fluids, 2006. **40**(1): p. 80-97.

82. Ruhnau, P., A. Stahl, and C. Schnorr, *On-line variational estimation of dynamical fluid flows with physics-based spatio-temporal regularization.* Pattern Recognition, Proceedings, 2006. **4174**: p. 444-454.

83. Liu, T.S. and L.X. Shen, *Fluid flow and optical flow.* Journal of Fluid Mechanics, 2008. **614**: p. 253-291.

84. Ruhnau, P., et al., *Variational optical flow estimation for particle image velocimetry.* Experiments in Fluids, 2005. **38**(1): p. 21-32.

85. Stanislas, M., K. Okamoto, and C. Kahler, *Main results of the First International PIV Challenge.* Measurement Science & Technology, 2003. **14**(10): p. R63-R89.

86. Stanislas, M., et al., *Main results of the second international PIV challenge.* Experiments in Fluids, 2005. **39**(2): p. 170-191.

87. Stanislas, M., et al., *Main results of the third international PIV Challenge.* Experiments in Fluids, 2008. **45**(1): p. 27-71.

88. Whitaker, R.T., *A level-set approach to image blending.* IEEE Transactions on Image Processing, 2000. **9**(11): p. 1849-1861.

89. Mukherjee, D. and N. Ray, *Contour Interpolation using Level Sets*, in *IET Image Processing*. 2005. p. 22.

APPENDIX A. THE EULER-LAGRANGE EQUATIONS [15, 75-76]

According to the Calculus of Variations, minimization of the functional

$$E^2 = \int \mathcal{F} dx \qquad \textbf{(A.1)}$$

over the mapping

$$\mathcal{F} = f\big(x, g(x), g'(x)\big) \qquad \textbf{(A.2)}$$

is achieved when $\mathcal{F}$ satisfies the Euler-Lagrange equation

$$\frac{\partial \mathcal{F}}{\partial g(x)} - \frac{d}{dx}\left(\frac{\partial \mathcal{F}}{\partial g\prime(x)}\right) = 0. \qquad \textbf{(A.3)}$$

Generalized to multiple dimensions,

$$E^2 = \int_{\Omega} \mathcal{F} d\boldsymbol{x} \qquad \textbf{(A.4)}$$

$$\mathcal{F} = f\big(x_i, g(x_i), g'(x_i)\big). \qquad \textbf{(A.5)}$$

The *n* Euler-Lagrange equations representing an *n*-dimensional space thus become

$$\frac{\partial \mathcal{F}}{\partial g(x_i)} - \sum_{i=0}^{n} \frac{d}{dx_i}\left(\frac{\partial \mathcal{F}}{\partial g\prime(x_i)}\right) = 0. \qquad \textbf{(A.6)}$$

Since energy E^2 is being minimized in $\Omega \subset \mathbb{R}^2$ for the purpose of determining optical flow variables u and v, $\mathcal{F} = \alpha^2 \varepsilon_s^2 + \varepsilon_b^2 = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{u}')$ and the Euler-Lagrange equations for this problem take the form

$$\frac{\partial \mathcal{F}}{\partial u} - \frac{d}{dx}\left(\frac{\partial \mathcal{F}}{u_x}\right) - \frac{d}{dy}\left(\frac{\partial \mathcal{F}}{u_y}\right) = 0 \qquad \textbf{(A.7)}$$

$$\frac{\partial \mathcal{F}}{\partial v} - \frac{d}{dx}\left(\frac{\partial \mathcal{F}}{v_x}\right) - \frac{d}{dy}\left(\frac{\partial \mathcal{F}}{v_y}\right) = 0. \qquad \textbf{(A.8)}$$

Expanding $\mathcal{F}$ yields

$$\mathcal{F} = I_x^2u^2 + I_y^2v^2 + I_t^2$$
$$+2\left(I_xI_yuv + I_xI_tu + I_yI_tv\right)$$
$$+\alpha^2\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right). \tag{A.9}$$

Derivatives of $\mathcal{F}$ in u, v, u', and v' are

$$\frac{\partial\mathcal{F}}{\partial u} = 2\left(I_x^2u + I_xI_yv + I_xI_t\right) \tag{A.10}$$

$$\frac{\partial\mathcal{F}}{\partial v} = 2\left(I_y^2v + I_xI_yu + I_yI_t\right) \tag{A.11}$$

$$\frac{\partial\mathcal{F}}{\partial u_x} = 2\alpha^2u_x \tag{A.12}$$

$$\frac{\partial\mathcal{F}}{\partial u_y} = 2\alpha^2u_y \tag{A.13}$$

$$\frac{\partial\mathcal{F}}{\partial v_x} = 2\alpha^2v_x \tag{A.14}$$

$$\frac{\partial\mathcal{F}}{\partial v_y} = 2\alpha^2v_y. \tag{A.15}$$

The resulting Euler-Lagrange equations are then

$$I_x^2u + I_xI_yv + I_xI_t - \alpha^2u_{xx} - \alpha^2u_{yy} = 0 \tag{A.16}$$

$$I_y^2v + I_xI_yu + I_yI_t - \alpha^2v_{xx} - \alpha^2v_{yy} = 0, \tag{A.17}$$

or, more compactly,

$$I_x\left(I_xu + I_yv + I_t\right) - \alpha^2\nabla^2u = 0 \tag{A.18}$$

$$I_y\left(I_xu + I_yv + I_t\right) - \alpha^2\nabla^2v = 0. \tag{A.19}$$

APPENDIX B. ESTIMATING DERIVATIVES (1ST ORDER) AND LAPLACIANS [15]

Forward differencing for estimating spatial and temporal derivatives:

$$I_x \approx \frac{1}{4}\left[\left(I_{i,j+1,k} - I_{i,j,k}\right) + \left(I_{i+1,j+1,k} - I_{i+1,j,k}\right) + \left(I_{i,j+1,k+1} - I_{i,j,k+1}\right) + \left(I_{i+1,j+1,k+1} - I_{i+1,j,k+1}\right)\right] \quad \textbf{(B.1)}$$

$$I_y \approx \frac{1}{4}\left[\left(I_{i+1,j,k} - I_{i,j,k}\right) + \left(I_{i+1,j+1,k} - I_{i,j+1,k}\right) + \left(I_{i+1,j,k+1} - I_{i,j,k+1}\right) + \left(I_{i+1,j+1,k+1} - I_{i,j+1,k+1}\right)\right] \quad \textbf{(B.2)}$$

$$I_t \approx \frac{1}{4}\left[\left(I_{i,j,k+1} - I_{i,j,k}\right) + \left(I_{i+1,j,k+1} - I_{i+1,j,k}\right) + \left(I_{i,j+1,k+1} - I_{i,j+1,k}\right) + \left(I_{i+1,j+1,k+1} - I_{i+1,j+1,k}\right)\right] \quad \textbf{(B.3)}$$

Flow velocity Laplacian estimates:

$$\nabla^2 u \approx \bar{u}_{i,j,k} - u_{i,j,k} \quad \textbf{(B.4)}$$

$$\nabla^2 v \approx \bar{v}_{i,j,k} - v_{i,j,k} \quad \textbf{(B.5)}$$

where

$$\bar{u}_{i,j,k} = \frac{1}{6}\left[u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\right] + \frac{1}{12}\left[u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\right] \quad \textbf{(B.6)}$$

$$\bar{v}_{i,j,k} = \frac{1}{6}\left[v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\right] + \frac{1}{12}\left[v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\right] \quad \textbf{(B.7)}$$

APPENDIX C. THE OPTICAL FLOW FIELD EQUATIONS

The Euler-Lagrange equations can be recast in a more convenient form for obtaining solutions to *u* and *v* [15]:

$$(\alpha^2 + I_x^2)u + I_x I_y v = \alpha^2 \bar{u} - I_x I_t \tag{C.1}$$

$$I_x I_y u + \left(\alpha^2 + I_y^2\right)v = \alpha^2 \bar{v} - I_y I_t. \tag{C.2}$$

In matrix form, these become

$$\begin{bmatrix} \alpha^2 + I_x^2 & I_x I_y \\ I_x I_y & \alpha^2 + I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha^2 \bar{u} - I_x I_t \\ \alpha^2 \bar{v} - I_y I_t \end{bmatrix}. \tag{C.3}$$

At this stage, Cramer's rule can be used to easily find solutions for *u* and *v* [77]:

$$u = \frac{\det \boldsymbol{A_1}}{\det \boldsymbol{A}} \tag{C.4}$$

$$v = \frac{\det \boldsymbol{A_2}}{\det \boldsymbol{A}}. \tag{C.5}$$

A is the coefficient matrix, and ***A₁***, ***A₂*** are defined below.

$$\boldsymbol{A} = \begin{bmatrix} \alpha^2 + I_x^2 & I_x I_y \\ I_x I_y & \alpha^2 + I_y^2 \end{bmatrix} \tag{C.6}$$

$$\boldsymbol{A_1} = \begin{bmatrix} \alpha^2 \bar{u} - I_x I_t & I_x I_y \\ \alpha^2 \bar{v} - I_y I_t & \alpha^2 + I_y^2 \end{bmatrix} \tag{C.7}$$

$$\boldsymbol{A_2} = \begin{bmatrix} \alpha^2 + I_x^2 & \alpha^2 \bar{u} - I_x I_t \\ I_x I_y & \alpha^2 \bar{v} - I_y I_t \end{bmatrix} \tag{C.8}$$

Determinants are calculated to be

$$\det \boldsymbol{A} = \begin{vmatrix} \alpha^2 + I_x^2 & I_x I_y \\ I_x I_y & \alpha^2 + I_y^2 \end{vmatrix} = \alpha^2\left(\alpha^2 + I_x^2 + I_y^2\right) \tag{C.9}$$

$$\det \boldsymbol{A_1} = \begin{vmatrix} \alpha^2 \bar{u} - I_x I_t & I_x I_y \\ \alpha^2 \bar{v} - I_y I_t & \alpha^2 + I_y^2 \end{vmatrix}$$

$$= \alpha^2\left(\alpha^2 \bar{u} + I_y^2 \bar{u} - I_x I_y \bar{v} - I_x I_t\right) \tag{C.10}$$

$$\det \boldsymbol{A_2} = \begin{vmatrix} \alpha^2 + I_x^2 & \alpha^2\bar{u} - I_x I_t \\ I_x I_y & \alpha^2\bar{v} - I_y I_t \end{vmatrix}$$

$$= \alpha^2\left(\alpha^2\bar{v} + I_x^2\bar{v} - I_x I_y \bar{u} - I_y I_t\right). \quad \textbf{(C.11)}$$

Applying Cramer's rule gives *u* and *v* as

$$\left(\alpha^2 + I_x^2 + I_y^2\right)u = \left(\alpha^2 + I_y^2\right)\bar{u} - I_x I_y \bar{v} - I_x I_t \quad \textbf{(C.12)}$$

$$\left(\alpha^2 + I_x^2 + I_y^2\right)v = (\alpha^2 + I_x^2)\bar{v} - I_x I_y \bar{u} - I_y I_t \quad \textbf{(C.13)}$$

Further algebraic manipulation leads to

$$\left(\alpha^2 + I_x^2 + I_y^2\right)(u - \bar{u}) = -I_x\left(I_x\bar{u} + I_y\bar{v} + I_t\right) \quad \textbf{(C.14)}$$

$$\left(\alpha^2 + I_x^2 + I_y^2\right)(v - \bar{v}) = -I_y\left(I_x\bar{u} + I_y\bar{v} + I_t\right) \quad \textbf{(C.15)}$$

and finally,

$$u = \bar{u} - I_x\left(I_x\bar{u} + I_y\bar{v} + I_t\right)/\left(\alpha^2 + I_x^2 + I_y^2\right) \quad \textbf{(C.16)}$$

$$v = \bar{v} - I_y\left(I_x\bar{u} + I_y\bar{v} + I_t\right)/\left(\alpha^2 + I_x^2 + I_y^2\right). \quad \textbf{(C.17)}$$

APPENDIX D. CENTRAL DIFFERENCING APPROXIMATIONS IN THE HORN-SCHUNCK FORMULATION

Central differencing for estimating spatial and temporal derivatives:

$$\begin{aligned} I_x \approx \frac{1}{8}\big[&\left(I_{i-1,j+1,k-1} - I_{i-1,j-1,k-1}\right) \\ &+\left(I_{i+1,j+1,k-1} - I_{i+1,j-1,k-1}\right) \\ &+\left(I_{i-1,j+1,k+1} - I_{i-1,j-1,k+1}\right) \\ &+\left(I_{i+1,j+1,k+1} - I_{i+1,j-1,k+1}\right)\big] \end{aligned} \tag{D.1}$$

$$\begin{aligned} I_y \approx \frac{1}{8}\big[&\left(I_{i+1,j-1,k-1} - I_{i-1,j-1,k-1}\right) \\ &+\left(I_{i+1,j+1,k-1} - I_{i-1,j+1,k-1}\right) \\ &+\left(I_{i+1,j-1,k+1} - I_{i-1,j-1,k+1}\right) \\ &+\left(I_{i+1,j+1,k+1} - I_{i-1,j+1,k+1}\right)\big] \end{aligned} \tag{D.2}$$

$$\begin{aligned} I_t \approx \frac{1}{8}\big[&\left(I_{i-1,j-1,k+1} - I_{i-1,j-1,k-1}\right) \\ &+\left(I_{i+1,j-1,k+1} - I_{i+1,j-1,k-1}\right) \\ &+\left(I_{i-1,j+1,k+1} - I_{i-1,j+1,k-1}\right) \\ &+\left(I_{i+1,j+1,k+1} - I_{i+1,j+1,k-1}\right)\big] \end{aligned} \tag{D.3}$$

APPENDIX E. THE MULTIGRID METHOD [14]

For illustrative purposes, a linear operator is defined as acting on the flow velocity components *u* and *v*.

$$\mathcal{L}_1(u,v) = \alpha^2\left(I_x u + I_y v\right)I_x - \nabla^2 u \tag{E.1}$$

$$\mathcal{L}_2(u,v) = \alpha^2\left(I_x u + I_y v\right)I_y - \nabla^2 v \tag{E.2}$$

Simplified as such, the Euler-Lagrange equations can be written in vector form.

$$\begin{bmatrix} \mathcal{L}_1(u,v) \\ \mathcal{L}_2(u,v) \end{bmatrix} = \begin{bmatrix} -\alpha^2 I_t I_x \\ -\alpha^2 I_t I_y \end{bmatrix} \tag{E.3}$$

From here, the multigrid algorithm proceeds as follows:

i) Start with an initial guess solution $(\hat{u}, \hat{v})$.

ii) Use the gradient descent equations for some n_1 iterations to refine $(\hat{u}, \hat{v})$ closer to their actual solution.

iii) Compute the residual *r* on the refined initial guess:

$$r = \begin{bmatrix} \mathcal{L}_1(\hat{u},\hat{v}) \\ \mathcal{L}_2(\hat{u},\hat{v}) \end{bmatrix} + \begin{bmatrix} \alpha^2 I_t I_x \\ \alpha^2 I_t I_y \end{bmatrix}. \tag{E.4}$$

iv) Down sample the residual *r* to obtain r_{coarse}, the residual on a coarse mesh.

v) Solve for functions f_c and g_c on the coarse mesh, where f_c and g_c satisfy

$$r_{coarse} = \begin{bmatrix} \mathcal{L}_{1,coarse}(f_c, g_c) \\ \mathcal{L}_{2,coarse}(f_c, g_c) \end{bmatrix}. \tag{E.5}$$

vi) Interpolate f_c and g_c onto the original grid to get *f* and *g*.

vii) Add *f* and *g* to the refined guess $(\hat{u}, \hat{v})$:

$$\hat{u} = \hat{u} + f \tag{E.6}$$

$$\hat{v} = \hat{v} + g. \tag{E.7}$$

viii) Refine $(\hat{u}, \hat{v})$ a final time using gradient descent for n_2 iterations.

Coarsening is performed recursively until the resulting linear equation may be solved exactly. Then, the grid is recursively refined, injecting each coarse level solution to the next finer level until the original grid density is reached once again. At this point, the residual is checked for convergence; the process is repeated with the new solution as the initialization if the residual remains too high.

APPENDIX F. NUMERICALLY APPROXIMATING THE TWO-PHASE OPTICAL FLOW FIELD EQUATIONS [9]

Linearization and numerical discretization of the two-phase Euler-Lagrange equations is supplied in the vein of Brox et al. as follows:

$$0 = H_\Delta(\kappa\varphi)\Psi'\left[\left(I_z^{k+1,+}\right)^2 + \gamma\left(I_{xz}^{k+1,+}\right)^2 + \gamma\left(I_{yz}^{k+1,+}\right)^2\right]$$
$$\cdot\left[I_x^{k,+}I_z^{k+1,+} + \gamma I_{xx}^{k,+}I_{xz}^{k+1,+} + \gamma I_{xy}^{k,+}I_{yz}^{k+1,+}\right]$$
$$-\mu\nabla\cdot\left[H_\Delta(\varphi)\Psi'(|\nabla u^{k+1,+}|^2 + |\nabla v^{k+1,+}|^2)\nabla u^{k+1,+}\right] \qquad \textbf{(F.1)}$$

$$0 = H_\Delta(-\kappa\varphi)\Psi'\left[\left(I_z^{k+1,-}\right)^2 + \gamma\left(I_{xz}^{k+1,-}\right)^2 + \gamma\left(I_{yz}^{k+1,-}\right)^2\right]$$
$$\cdot\left[I_x^{k,-}I_z^{k+1,-} + \gamma I_{xx}^{k,-}I_{xz}^{k+1,-} + \gamma I_{xy}^{k,-}I_{yz}^{k+1,-}\right]$$
$$-\mu\nabla\cdot\left[H_\Delta(-\varphi)\Psi'(|\nabla u^{k+1,-}|^2 + |\nabla v^{k+1,-}|^2)\nabla u^{k+1,-}\right] \qquad \textbf{(F.2)}$$

$$0 = H_\Delta(\kappa\varphi)\Psi'\left[\left(I_z^{k+1,+}\right)^2 + \gamma\left(I_{xz}^{k+1,+}\right)^2 + \gamma\left(I_{yz}^{k+1,+}\right)^2\right]$$
$$\cdot\left[I_y^{k,+}I_z^{k+1,+} + \gamma I_{yy}^{k,+}I_{yz}^{k+1,+} + \gamma I_{xy}^{k,+}I_{xz}^{k+1,+}\right]$$
$$-\mu\nabla\cdot\left[H_\Delta(\varphi)\Psi'(|\nabla u^{k+1,+}|^2 + |\nabla v^{k+1,+}|^2)\nabla v^{k+1,+}\right] \qquad \textbf{(F.3)}$$

$$0 = H_\Delta(-\kappa\varphi)\Psi'\left[\left(I_z^{k+1,-}\right)^2 + \gamma\left(I_{xz}^{k+1,-}\right)^2 + \gamma\left(I_{yz}^{k+1,-}\right)^2\right]$$
$$\cdot\left[I_y^{k,-}I_z^{k+1,-} + \gamma I_{yy}^{k,-}I_{yz}^{k+1,-} + \gamma I_{xy}^{k,-}I_{xz}^{k+1,-}\right]$$
$$-\mu\nabla\cdot\left[H_\Delta(-\varphi)\Psi'(|\nabla u^{k+1,-}|^2 + |\nabla v^{k+1,-}|^2)\nabla v^{k+1,-}\right]. \qquad \textbf{(F.4)}$$

In the foregoing equations, H_Δ is a numerical approximation to the Heaviside function:

$$H_\Delta(x) = \frac{1}{2}\left[1 + \frac{2}{\pi}\arctan\left(\frac{x}{\Delta}\right)\right], \qquad \textbf{(F.5)}$$

where Δ represents Eulerian grid spacing. After iterating in *k*, 1st order Taylor expansion is used to approximate $I_*^{k+1,\pm}$ in the same manner as before.

$$I_z^{k+1,\pm} \approx I_z^{k,\pm} + I_x^{k,\pm} du^{k,\pm} + I_y^{k,\pm} dv^{k,\pm} \tag{F.6}$$

$$I_{xz}^{k+1,\pm} \approx I_{xz}^{k,\pm} + I_{xx}^{k,\pm} du^{k,\pm} + I_{xy}^{k,\pm} dv^{k,\pm} \tag{F.7}$$

$$I_{yz}^{k+1,\pm} \approx I_{yz}^{k,\pm} + I_{xy}^{k,\pm} du^{k,\pm} + I_{yy}^{k,\pm} dv^{k,\pm} \tag{F.8}$$

$$u^{k+1,\pm} = u^{k,\pm} + du^{k,\pm} \tag{F.9}$$

$$v^{k+1,\pm} = v^{k,\pm} + dv^{k,\pm}. \tag{F.10}$$

Finally, making use of the "data term robustness" and "smoothness diffusivity" definitions put forth by Brox et al. (Equations 1.52 and 1.53), and utilizing their second nested fixed-point iteration scheme to linearize Ψ', the final form of the Euler-Lagrange equations becomes

$$\begin{aligned} 0 = {} & H_\Delta(\pm\kappa\varphi^l)(\Psi')_b^{k,l,\pm} \cdot \left\{I_x^{k,\pm}\left[I_z^{k,\pm} + I_x^{k,\pm} du^{k,l+1,\pm} + I_y^{k,\pm} dv^{k,l+1,\pm}\right]\right\} \\ & + \gamma H_\Delta(\pm\kappa\varphi^l)(\Psi')_b^{k,l,\pm}\left\{I_{xx}^{k,\pm}\left[I_{xz}^{k,\pm} + I_{xx}^{k,\pm} du^{k,l+1,\pm} + I_{xy}^{k,\pm} dv^{k,l+1,\pm}\right]\right. \\ & \left. + I_{xy}^{k,\pm}\left[I_{yz}^{k,\pm} + I_{xy}^{k,\pm} du^{k,l+1,\pm} + I_{yy}^{k,\pm} dv^{k,l+1,\pm}\right]\right\} \\ & - \mu\nabla \cdot \left\{H_\Delta(\pm\kappa\varphi^l)(\Psi')_s^{k,l,\pm}\, \nabla(u^{k,\pm} + du^{k,l+1,\pm})\right\} \end{aligned} \tag{F.11}$$

and

$$\begin{aligned} 0 = {} & H_\Delta(\pm\kappa\varphi^l)(\Psi')_b^{k,l,\pm} \cdot \left\{I_y^{k,\pm}\left[I_z^{k,\pm} + I_x^{k,\pm} du^{k,l+1,\pm} + I_y^{k,\pm} dv^{k,l+1,\pm}\right]\right\} \\ & + \gamma H_\Delta(\pm\kappa\varphi^l)(\Psi')_b^{k,l,\pm}\left\{I_{yy}^{k,\pm}\left[I_{yz}^{k,\pm} + I_{xy}^{k,\pm} du^{k,l+1,\pm} + I_{yy}^{k,\pm} dv^{k,l+1,\pm}\right]\right. \\ & \left. + I_{xy}^{k,\pm}\left[I_{xz}^{k,\pm} + I_{xx}^{k,\pm} du^{k,l+1,\pm} + I_{xy}^{k,\pm} dv^{k,l+1,\pm}\right]\right\} \\ & - \mu\nabla \cdot \left\{H_\Delta(\pm\kappa\varphi^l)(\Psi')_s^{k,l,\pm}\, \nabla(v^{k,\pm} + dv^{k,l+1,\pm})\right\}. \end{aligned} \tag{F.12}$$

APPENDIX G. THE DISCRETE LEVEL SET EVOLUTION EQUATION FOR 2-PHASE OPTICAL FLOW [9]

Adopting the notations of Chan and Vese [13],

$$C_1 = \left[\left(\frac{\varphi^l_{i+1,j} - \varphi^l_{i,j}}{h}\right)^2 + \left(\frac{\varphi^l_{i,j+1} - \varphi^l_{i,j-1}}{2h}\right)^2\right]^{-\frac{1}{2}} \qquad \textbf{(G.1)}$$

$$C_2 = \left[\left(\frac{\varphi^l_{i,j} - \varphi^l_{i-1,j}}{h}\right)^2 + \left(\frac{\varphi^l_{i-1,j+1} - \varphi^l_{i-1,j-1}}{2h}\right)^2\right]^{-\frac{1}{2}} \qquad \textbf{(G.2)}$$

$$C_3 = \left[\left(\frac{\varphi^l_{i+1,j} - \varphi^l_{i-1,j}}{2h}\right)^2 + \left(\frac{\varphi^l_{i,j+1} - \varphi^l_{i,j}}{h}\right)^2\right]^{-\frac{1}{2}} \qquad \textbf{(G.3)}$$

$$C_4 = \left[\left(\frac{\varphi^l_{i+1,j-1} - \varphi^l_{i-1,j-1}}{2h}\right)^2 + \left(\frac{\varphi^l_{i,j} - \varphi^l_{i,j-1}}{h}\right)^2\right]^{-\frac{1}{2}} \qquad \textbf{(G.4)}$$

$$m = \frac{\Delta t}{h^2}\delta_\Delta(\varphi_{i,j})\nu \qquad \textbf{(G.5)}$$

$$C = 1 + m(C_1 + C_2 + C_3 + C_4). \qquad \textbf{(G.6)}$$

Discretization of brightness and smoothness terms, respectively, gives

$$f^{l,+}_{i,j} = \Psi\left[\left(I^{l,+}_{z\,i,j}\right)^2 + \gamma\left(I^{l,+}_{xz\,i,j}\right)^2 + \gamma\left(I^{l,+}_{yz\,i,j}\right)^2\right] \qquad \textbf{(G.7)}$$

$$f^{l,-}_{i,j} = \Psi\left[\left(I^{l,-}_{z\,i,j}\right)^2 + \gamma\left(I^{l,-}_{xz\,i,j}\right)^2 + \gamma\left(I^{l,-}_{yz\,i,j}\right)^2\right] \qquad \textbf{(G.8)}$$

$$g^{l,+}_{i,j} = \Psi\left[\left(\frac{u^{l,+}_{i+1,j} - u^{l,+}_{i-1,j}}{2h}\right)^2 + \left(\frac{u^{l,+}_{i,j+1} - u^{l,+}_{i,j-1}}{2h}\right)^2 + \left(\frac{v^{l,+}_{i+1,j} - v^{l,+}_{i-1,j}}{2h}\right)^2 + \left(\frac{v^{l,+}_{i,j+1} - v^{l,+}_{i,j-1}}{2h}\right)^2\right] \qquad \textbf{(G.9)}$$

$$g_{i,j}^{l,-} = \Psi\left[\left(\frac{u_{i+1,j}^{l,-}-u_{i-1,j}^{l,-}}{2h}\right)^2 + \left(\frac{u_{i,j+1}^{l,-}-u_{i,j-1}^{l,-}}{2h}\right)^2 + \left(\frac{v_{i+1,j}^{l,-}-v_{i-1,j}^{l,-}}{2h}\right)^2 + \left(\frac{v_{i,j+1}^{l,-}-v_{i,j-1}^{l,-}}{2h}\right)^2\right]. \quad \textbf{(G.10)}$$

Finally, the discrete two-phase level set equation is

$$\varphi_{i,j}^{l+1} = \frac{1}{C}\left[\varphi_{i,j}^{l} + m\left(C_1\varphi_{i+1,j}^{l} + C_2\varphi_{i-1,j}^{l} + C_3\varphi_{i,j+1}^{l} + C_4\varphi_{i,j-1}^{l}\right)\right.$$
$$-\Delta t\mu\delta_\Delta\left(\varphi_{i,j}^{l}\right)\cdot\left(g_{i,j}^{l,+} - g_{i,j}^{l,-}\right)$$
$$\left.-\Delta t\kappa\delta_\Delta\left(\kappa\varphi_{i,j}^{l}\right)\cdot\left(f_{i,j}^{l,+} - f_{i,j}^{l,-}\right)\right. \quad \textbf{(G.11)}$$

APPENDIX H. COMPUTING OPTICAL FLOW DERIVATIVES AND MATRIX COEFFICIENTS

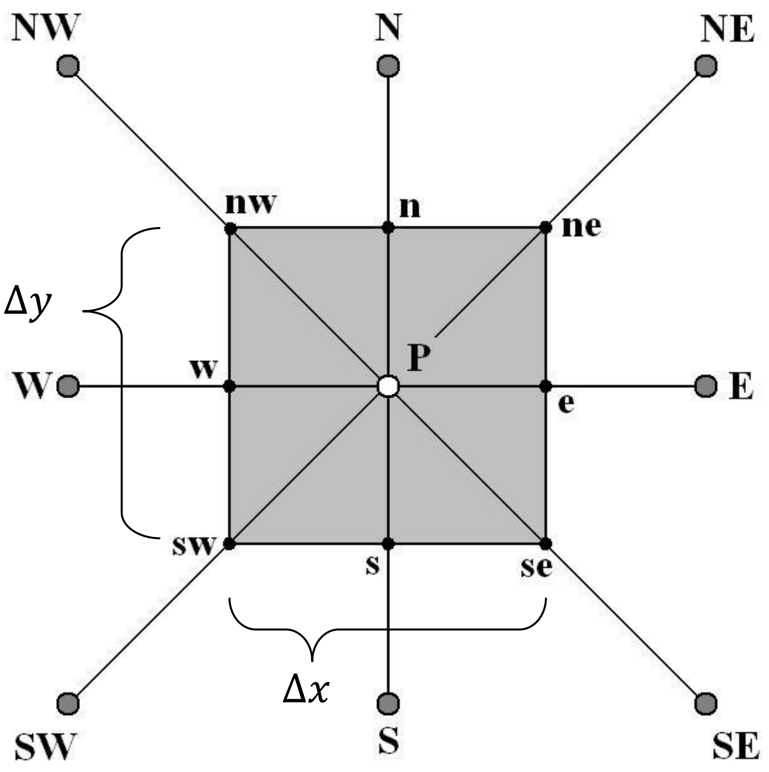


Figure H1. The stencil used to compute derivatives: A pixel (usually considered to be of unit area, but of dimension Δx by Δy more generically) is shown shaded in grey, with pixel center values labeled using uppercase letters and pixel face values labeled with lowercase letters.

All differencing approximations used for the optical flow field computations follow standard CFD methods. Away from image boundaries, derivatives are computed using central differencing:

First derivatives-

$$\varphi_x \approx \frac{\varphi_E - \varphi_W}{2\Delta x} \tag{H.1}$$

$$\varphi_y \approx \frac{\varphi_N - \varphi_S}{2\Delta y} \tag{H.2}$$

Second derivatives-

$$\varphi_{xx} \approx \frac{\varphi_E - 2\varphi_P + \varphi_W}{(\Delta x)^2} \quad \textbf{(H.3)}$$

$$\varphi_{yy} \approx \frac{\varphi_N - 2\varphi_P + \varphi_S}{(\Delta y)^2} \quad \textbf{(H.4)}$$

Mixed derivatives-

$$\varphi_{xy} \approx \frac{\varphi_{NE} - \varphi_{SE} - \varphi_{NW} + \varphi_{SW}}{4\Delta x \Delta y} \quad \textbf{(H.5)}$$

On domain edges, one-sided difference approximations are used:

East boundary-

$$\varphi_x \approx \frac{\varphi_P - \varphi_W}{\Delta x} \quad \textbf{(H.6)}$$

$$\varphi_{xx} = 0 \quad \textbf{(H.7)}$$

$$\varphi_{xy} \approx \frac{\varphi_N - \varphi_S - \varphi_{NW} + \varphi_{SW}}{2\Delta x \Delta y} \quad \textbf{(H.8)}$$

West boundary-

$$\varphi_x \approx \frac{\varphi_E - \varphi_P}{\Delta x} \quad \textbf{(H.9)}$$

$$\varphi_{xx} = 0 \quad \textbf{(H.10)}$$

$$\varphi_{xy} \approx \frac{\varphi_{NE} - \varphi_{SE} - \varphi_N + \varphi_S}{2\Delta x \Delta y} \quad \textbf{(H.11)}$$

North boundary-

$$\varphi_y \approx \frac{\varphi_P - \varphi_S}{\Delta y} \quad \textbf{(H.13)}$$

$$\varphi_{yy} = 0 \quad \textbf{(H.14)}$$

$$\varphi_{xy} \approx \frac{\varphi_E - \varphi_W - \varphi_{SE} + \varphi_{SW}}{2\Delta x \Delta y} \quad \textbf{(H.15)}$$

South boundary-

$$\varphi_y \approx \frac{\varphi_N - \varphi_P}{\Delta y} \quad \textbf{(H.16)}$$

$$\varphi_{yy} = 0 \quad \textbf{(H.17)}$$

$$\varphi_{xy} \approx \frac{\varphi_{NE}-\varphi_{NW}-\varphi_{E}+\varphi_{W}}{2\Delta x\Delta y} \quad \textbf{(H.18)}$$

In the domain corners, one-sided differencing is used in two directions:

Northeast corner-

$$\varphi_{x} \approx \frac{\varphi_{P}-\varphi_{W}}{\Delta x} \quad \textbf{(H.19)}$$

$$\varphi_{y} \approx \frac{\varphi_{P}-\varphi_{S}}{\Delta y} \quad \textbf{(H.20)}$$

$$\varphi_{xx} = \varphi_{yy} = 0 \quad \textbf{(H.21)}$$

$$\varphi_{xy} \approx \frac{\varphi_{P}-\varphi_{S}-\varphi_{W}+\varphi_{SW}}{\Delta x\Delta y} \quad \textbf{(H.22)}$$

Northwest corner-

$$\varphi_{x} \approx \frac{\varphi_{E}-\varphi_{P}}{\Delta x} \quad \textbf{(H.23)}$$

$$\varphi_{y} \approx \frac{\varphi_{P}-\varphi_{S}}{\Delta y} \quad \textbf{(H.25)}$$

$$\varphi_{xx} = \varphi_{yy} = 0 \quad \textbf{(H.26)}$$

$$\varphi_{xy} \approx \frac{\varphi_{E}-\varphi_{SE}-\varphi_{P}+\varphi_{S}}{\Delta x\Delta y} \quad \textbf{(H.27)}$$

Southeast corner-

$$\varphi_{x} \approx \frac{\varphi_{P}-\varphi_{W}}{\Delta x} \quad \textbf{(H.28)}$$

$$\varphi_{y} \approx \frac{\varphi_{N}-\varphi_{P}}{\Delta y} \quad \textbf{(H.29)}$$

$$\varphi_{xx} = \varphi_{yy} = 0 \quad \textbf{(H.30)}$$

$$\varphi_{xy} \approx \frac{\varphi_{N}-\varphi_{P}-\varphi_{NW}+\varphi_{W}}{\Delta x\Delta y} \quad \textbf{(H.31)}$$

Southwest corner-

$$\varphi_{x} \approx \frac{\varphi_{E}-\varphi_{P}}{\Delta x} \quad \textbf{(H.32)}$$

$$\varphi_{y} \approx \frac{\varphi_{N}-\varphi_{P}}{\Delta y} \quad \textbf{(H.33)}$$

$$\varphi_{xx} = \varphi_{yy} = 0 \quad \textbf{(H.34)}$$

$$\varphi_{xy} \approx \frac{\varphi_{NE}-\varphi_E-\varphi_N+\varphi_P}{\Delta x \Delta y} \tag{H.35}$$

Pixel face values are simply taken as the average of the pixel-center value φ_P and that of its neighbor in the appropriate direction:

$$\varphi_n = \frac{\varphi_N+\varphi_P}{2} \tag{H.36}$$

$$\varphi_s = \frac{\varphi_S+\varphi_P}{2} \tag{H.37}$$

$$\varphi_e = \frac{\varphi_E+\varphi_P}{2} \tag{H.38}$$

$$\varphi_w = \frac{\varphi_W+\varphi_P}{2} \tag{H.39}$$

When there is no neighboring pixel in a given direction, as is the case on image boundaries, the face value in that direction is taken to be equal to φ_P.