DISSERTATION

SUBMITTED

TO THE

COMBINED FACULTIES FOR THE NATURAL SCIENCES AND FOR MATHEMATICS

OF THE

RUPERTO-CAROLA UNIVERSITY OF HEIDELBERG, GERMANY FOR THE DEGREE OF DOCTOR OF NATURAL SCIENCES

Put forward by
Dipl.–Mat, M.Sc. Dorotea Dudaš
Born in: Osijek, Croatia
Oral examination:

Vortex Extraction Of Vector Fields

Advisors: Prof. Dr. Rolf Rannacher

PD Dr. Christoph S. Garbe

Abstract

Spinning, turbulent structures swirling around its centers within various flow media are known as vortices. The capability of locating and extracting vortical structures in flow data is crucial for understanding the flow. Vortices also have a strong impact on flow control and transport processes.

Real-time vortex extraction methods are presented, offering immediate notion of the shape and location of the vortex structures. Using a real-time fluid simulation based on Navier-Stokes equations presented in [46], several vortex extraction methods are interactively performed in real-time. Following vortex extraction methods are implemented using the GPU: vorticity threshold, Q criterion, λ_2 criterion, the eigenvector method via parallel vectors operator (PVO) and the eigenvector method via coplanar vectors operator (CVO).

Diffusional methods outputting flow fields with preserved/enhanced vortical structures are also presented. Such methods are useful for obtaining an alternative insight into vortices within a flow field and can also be used within the real-time simulation.

Using a number of human performed gestures for human-computer interaction, special ensemble flow fields are produced. Detecting vortices from these gesture ensemble range flows is introduced as aid for gesture classification. Gesture range data is recorded using the Microsoft Kinect device. Range or scene flow is a 3D vector field describing movement within a scene. Range data consists of images (color channels) and corresponding depth images (depth channels) in which the distance of objects is recorded as a grayscale image. Ensemble range flow is estimated from gesture videos. Ensemble flow describes the overall flow within the scene and is obtained by averaging the structure tensor throughout the scene. Vortices are extracted from an ensemble range flow of the gestures. Their

number and location is offering an additional parameter for gesture classification.

Collection of methods for detecting vortices and obtaining vector fields with emphasized vortices are introduced in this thesis. Real-time execution of vortex extraction methods offers an instant notion of the nature of the flow. Diffusional methods can serve as a processing step within the real-time vortex extraction. As an additional application, gesture ensemble flow is presented. By detecting its vortices, a parameter for gesture classification is introduced.

Zusammenfassung

In einer Strömung werden rotierende, turbulente Strukturen, die sich um ein Zentrum drehen als Wirbel bezeichnet. Deren Charakterisierung und Lokalisierung ist essentiell für das Verständnis einer Strömung. Weiterhin haben Wirbel einen bedeutenden Einfluss auf das Verhalten einer Flüssigkeit und Transportprozesse innerhalb der Flüssigkeit.

Hier werden Methoden für die Extraktion von Wirbeln in Echtzeit vorgestellt, die die sofortige Ansicht von Ort und Form der Wirbelstruktur ermöglichen. Mehrere Wirbelextraktionsmethoden werden interaktiv und in Echtzeit ausgeführt und Daten aus einer Simulation der Navier-Stokes-Gleichungen in Echt-zeit ([46]) angewendet. Dabei handelt es sich um folgende Methoden: Vortizitätsschwellwert, Q Kriterium, λ_2 Kriterium, Eigenvektorenmethode über parallele Vektoroperatoren (PVO) und Eigenvektorenmethode über Koplanare Vektoroperatoren (CVO).

Weiterhin werden auf Diffusionsprozessen basierte Methoden zur Generierung von Strömungsfeldern präsentiert mit erhaltenen/verstärkten Wirbelstrukturen. Diese Methoden ermöglichen einen alternativen Einblick in Wirbelstrukturen innerhalb des Geschwindigkeitsfeldes und können auch in der Echtzeitsimulation eingesetzt werden.

Ensemble Geschwindigkeitsfelder wurden von einer Auswahl menschlicher Gesten im Rahmen der Mensch-Computer-Interaktion generiert. Die Vortexdetektion aus diesen gesteninduzierten Geschwindigkeitsfeldern wird als Hilfe in der Gestenklassifikation vorgestellt. Die nötigen Daten wurden mit der Microsoft Kinect Kamera aufgenommen. Das aufgenommene Geschwindikeitsfeld ist ein 3D Vektorfeld, das die Bewegung innerhalb eine Szene beschreibt. Die zugehörigen Daten bestehen aus 3-Kanal Farbbildern und dazugehörigen

Tiefenkarten, in denen der Abstand der Objekte als Grauwertbild gespeichert wird. Ein *Ensemble* Geschwindigkeitsfeld wird aus jedem Gestenvideo berechnet. Es beschreibt die durchschnittliche Bewegung und wird durch die Mittelung des Strukturtensors über die gesamte Szene bestimmt. Im Anschluss werden die Wirbel aus den resultierenden *Ensemble* Geschwindigkeitsfeldern bestimmt. Ort und Anzahl der Wirbel stellen einen zusätzlichen Parameter für die Gestenklassifikation dar.

In dieser Arbeit wird eine Auswahl von Methoden zur Detektion von Wirbeln und Erzeugung von Vektorfeldern mit Schwerpunkt auf Wirbeln vorgestellt. Die Ausführung der Methoden der Wirbelextraktion in Echtzeit ermöglicht die sofortige Kenntnis der Art des Geschwindigkeitsfeldes. Die Anwendung der Diffusionsmethoden kann einen Schritt innerhalb der Vortexextraktion in Echtzeit darstellen. Als eine zusätzliche Anwendung wird die Generierung von *Ensemble* Geschwindigkeitsfeldern aus Gesten vorgestellt. Über die Detektion der entstehenden Wirbel wird ein zusätzlicher Parameter für die Gestenklassifikation eingeführt.

Contents

Al	ostrac	et			i	
Zι	ısamr	nenfass	sung		iii	
1	Intr	oductio	n		1	
2	Flov	v Data			7	
	2.1	Introd	uction		7	
	2.2	Simul	ated Data		8	
		2.2.1	Analytic	al Vortex Examples	9	
			2.2.1.1	Vector Field Topology	9	
		2.2.2	Real-Tin	ne Flow Simulation on the GPU	16	
			2.2.2.1	Navier-Stokes equations	17	
			2.2.2.2	Implementation	19	
	2.3	Recor	ded Data		25	
		2.3.1		ft Kinect	25	
3	Diff	usional	and Varia	ational Vortex Preserving	27	
	3.1	Introduction				
	3.2	Diffusion				
	3.3	3.3 Variational Methods			32	
	3.4 Vortex Preserving Diffusion and Variational Processes				34	
		3.4.1	Vortex P	reserving Diffusion	36	
		3.4.2	Variation	nal Formulation of Vortex Preserving	46	
		3.4.3	Results		49	

4	Real	l-Time	Vortex Ex	traction	53	
	4.1	Introd	Introduction			
	4.2	Overv	iew of the	Vortex Extraction Methods	55	
		4.2.1	Vortex R	Region Extraction	55	
			4.2.1.1	Vorticity Threshold	56	
			4.2.1.2	Q Criterion (Okubo-Weiss)	56	
			4.2.1.3	λ_{2} Method (Jeong-Hussain)	56	
		4.2.2	Vortex C	Core Extraction	57	
			4.2.2.1	Minimal Bending Energy Method	57	
			4.2.2.2	Sujudi-Haimes Method (Eigenvector Method) .	61	
			4.2.2.3	Parallel Vectors Operator	62	
			4.2.2.4	Feature Flow Fields	64	
			4.2.2.5	Coplanar Vectors Operator	65	
	4.3	Station	nary Flow	Parallel Vectors Operator on the GPU	68	
		4.3.1	Results	-	74	
	4.4	Real-T	ime Flow	Vortex Extraction on the GPU	76	
		4.4.1 Simulation Enhancement Through Nonlinear Isotropic				
		Diffusion				
		4.4.2	4.4.2 Implementation of Vortex Detection Methods			
			4.4.2.1	Vorticity Threshold	79	
			4.4.2.2	Q Criterion	79	
			4.4.2.3	λ_2 Criterion	79	
			4.4.2.4	Parallel Vectors Operator	79	
			4.4.2.5	Coplanar Vectors Operator	80	
		4.4.3	Results		81	
5	Gest			n by Detecting Vortices in Ensemble Flow	91	
	5.1	Introduction				
5.2 Range and Optic Flow					93	
	5.3	Correc	ction of Ra	nge Flow Computation - Combined Local-Global		
		Range	Flow		95	
		5.3.1	Results		99	
	5.4	Ensem	ble Range	e Flow	101	

	5.5	Vortex Detection of Ensemble Range Flow for Gesture Classification 10									
		5.5.1	Results	107							
		5.5.2	Vortex Preserving Diffusion of the Ensemble Range Flow	110							
6	Cone	clusion	and Future Work	113							
	6.1	Future	Work	114							
Ap	pend	ices		115							
A	Matl	nematio	cal Formulations Leading to a Discrete Explicit Scheme	117							
	A.1	Finite	Difference Derivative Approximations	117							
	A.2	Bound	ary Conditions	118							
	A.3	Linear	Isotropic Diffusion	119							
	A.4	Backw	vard Linear Isotropic Diffusion	119							
A.5 Nonlinear Isotropic Diffusion											
A.6 Backward Nonlinear Isotropic Diffusion											
						A.9 Discriminant Steered Energy Functional Requesting Similarity to the Original Vector Field					
A.10 Optic Flow											
	A.11	Range	Flow	128							
В	Ense	mble R	Range Flow of the Fluid Data With Estimated Depth	131							
		B.0.1	Estimating Depth From Single Monocular Images	132							
		B.0.2	Results	133							
Ac	know	ledgem	nent	135							
Af	fidavi	t		137							
Lis	st Of]	Figures		139							
Lis	st Of '	Fables		145							
Bil	bliogr	aphy		147							

Chapter 1

Introduction

Due to extreme climatic changes, importance of understanding vortices increases. Figure 1.1 shows different instances of vortical structures.







Figure 1.1: Vortices. Upper, left: World tallest artificial tornado serving as a smoke draining system in Mercedes-Benz museum in Stuttgart ([23], Copyright Daimler AG). Upper, right: Tornado storm in the "tornado alley" area in the central USA (Orchard, Iowa, June 2008 [25]). Lower: A fire tornado near Alice Springs, Australia (Sep. 18. 2012 [4]).

A collection of methods for detecting vortices and obtaining vector fields with emphasized vortices are introduced in this thesis. Real-time vortex extraction methods are presented, offering immediate notion of the shape and location

of the vortex structures. Diffusional methods outputting flow fields with preserved/enhanced vortical structures are also introduced. Detecting vortices from gesture ensemble range flows is introduced as a possible application of the developed methods to gesture classification.

Chapter 2 Chapter 2 is an introductory chapter presenting some of the background knowledge necessary for understanding the other chapters. Obtaining flow data, whether by designing analytical flow fields, by simulating the flow or by recording the flow, is crucial for the study of vortices. Chapter 2 is divided into sections on simulated (Section 2.2) and recorded data (Section 2.3).

Section 2.2 on simulated data is divided into "Analytical Vortex Examples" (Section 2.2.1) and "Real-Time Flow Simulation on the GPU" (Section 2.2.2) sections.

For designing analytical vector fields suitable for studying vortices, it is necessary to have some knowledge about the vector field topology. Certain types of first order critical points are considered to be vortex cores in 2D. Analytical 2D flow fields containing such critical points, i.e. vortex cores, are extensively used to test diffusion and diffusion-reaction processes in Chapter 3. Analytical 3D flows are designed to test 3D extraction methods presented in Chapter 4. Eigenvector method for vortex extraction ([47]) is a standard vortex extraction method. To understand where the motivation for its design originated from, first order 3D critical points are presented. Parallel vectors operator method, extensively used in Chapter 4, was motivated by the eigenvector method.

Simulating fluid in real-time is presented in Section 2.2.2. Using this fluid simulation in Chapter 4, vortex extraction methods are implemented in real-time. The simulation is based on Navier-Stokes equations which describe the motion of fluid substances. Implementation was done on a GPU using the CUDA framework. Interactive 3D real-time fluid simulation allows user to input additional force into the data volume, so creating flow data in real-time.

Section 2.3 presents the range data used in Chapter 5 for detecting vortices withing the gesture flow. The data is recorded using the Microsoft Kinect device. The device has a standard camera and an infrared sensor that records the depth information. Range data consists of images (color channels) and corresponding

depth images (depth channels) in which the distance of objects is recorded as a grayscale image. Such range data is used to create the ensemble range flow (Section 5.4) used for classifying gestures (Section 5.5).

Chapter 3 Chapter 3 presents diffusion and diffusion-reaction (variational) processes which produce vector fields with preserved/emphasized vortices.

Sections 3.2 and 3.3 give an introduction into diffusional and variational processing framework. Basic diffusion and diffusion-reaction processes are presented in order to facilitate the understanding of the upcoming sections.

Vortex preserving diffusion and variational processes are presented in Section 3.4.

Vortex preserving diffusion (Section 3.4.1) results in vector fields with preserved vortex areas where rest of the flow field is dampened. Diffusion produces simplified and information-reduced output. By choosing diffusivity functions that avoid vortex areas, diffusion processes are steered to preserve the information within them. Other features are diminished. It is also preferred, but not guaranteed that no new features are created. Different binary and continuous diffusivity functions are designed. One choice for designing diffusivity functions is making them dependent on the discriminant d of the characteristic polynomial of the Jacobian matrix. Negative d indicates an existence of a swirling area that possibly contains swirling critical points i.e. critical points that can be classified as vortex cores in 2D. Other choice for diffusivity functions includes considering area around detected vortex cores or regions. All approaches output vector fields with emphasized vortex areas.

Variational processes which give vector fields with emphasized/preserved vortex areas are presented in Section 3.4.2. A function H that causes preservation of certain areas and destruction of others is used to preserve vortices. H is again steered either by discriminant d, or by the areas around vortex cores or regions.

Designed processes offer an alternative insight into the structure of the flow and are useful for the further study of vortices.

Chapter 4 Chapter 4 presents real-time vortex detection and extraction methods. Various vortex core and region extraction methods exist. They operate as a post-processing step for locating vortex regions/cores. Here, standard vortex extraction methods are performed in real-time and are implemented within the real-time fluid simulation based on Navier-Stokes equations (presented in Section 2.2.2).

Overview of the vortex extraction methods is given in Section 4.2. Vortex core and region extraction methods needed for testing or for real-time implementation are covered. Following vortex region detection methods are presented: vorticity threshold, Q criterion and λ_2 criterion. Vortex core extraction includes minimal bending energy vortex extraction, the eigenvector method and the parallel vector operator (PVO). Eigenvector method is one of the standard vortex core extraction methods and is a predecessor of the PVO. Coplanar vectors operator (CVO) method, which is a generalization of the PVO, can be expressed through PVO.

GPU implementation of the parallel vectors operator method (PVO) for stationary vector fields is presented in Section 4.3. OpenCL framework is used for programming. PVO is used as a testing method for vortex extraction from 3D vector fields obtained by pausing the real-time simulation. This way, it is possible to verify the results from the real-time extraction. GPU implementation is optimized by considering only limited amount of faces within a volume cell of the 3D data.

Section 4.4 introduces real-time vortex core/region extraction. CUDA is used as a programming environment. Extraction methods are implemented within a real-time fluid simulation which is interactively influenced by the user. User is able to input additional force into the flow field in real-time (see Section 2.2.2) and so form, e.g. a helical-like flow suitable for testing of the real-time extraction methods. The results of the real-time extraction are color coded within the arrow or within the density plot of the real-time simulation. Different extraction methods can be compared in real-time by using different colors to depict them. Following vortex extraction methods are implemented: vorticity threshold, Q criterion, λ_2 criterion, the eigenvector method via parallel vectors operator (PVO) and the eigenvector method via coplanar vectors operator (CVO). Vortex preserving diffusion methods presented in Section 3.4.1, can be used as a replacement for the diffusion step of the fluid simulation, producing flows with slightly empha-

sized swirling areas.

Chapter 5 Chapter 5 deals with detecting vortices within the ensemble range flow of the gesture videos. The designed method is intended as a improvement of the existing gesture classification methods.

Section 5.2 is an introductory section presenting basics of optic and range flow. Range or scene flow is a 3D vector field describing movement within a scene. It requires image pairs of color images and of depth images as input. Depth images are encoding the distance of objects in an observed scene using grayscale values. Data recorded with Kinect device is used for testing.

Depth data obtained by Kinect contains many artifacts. Section 5.3 introduces a range flow method that removes those artifacts. Improved global-local range flow algorithm is presented. Magnitude of the derivatives of the input depth images is thresholded, thus removing the influence of artifacts to the resulting flow.

Ensemble range flow is introduced in Section 5.4. It describes the overall flow within the considered scene. The structure tensors of images are averaged throughout the entire data set.

Section 5.5 shows how vortex detection of the gesture ensemble range flows can aid gesture classification. Five volunteers recorded nine gestures using the Microsoft Kinect device. Ensemble range flow is calculated for every gesture. Vortex cores extracted from these ensemble range flows, i.e. their position and number can serve as a addition to certain gesture classification methods. Although this approach can be used as a standalone indicator, it is intended as an improvement of gesture classification methods that operate on data of reduced dimensions. A draft for a newly designed gesture classification is also proposed. The technique is based on mapping of the ensemble range flow magnitudes to 1D, so allowing fast classification. Gesture classification requires only the number of obtained vortices. Diffusion techniques from Section 3.4.1 can be used to create a flow field with emphasized vortices.

Appendix Different mathematical formulations leading to discrete explicit schemes are presented in Appendix A. Calculation of the ensemble range flow of the fluid data by estimating the depth data is given in Appendix B.

Chapter 2

Flow Data

2.1 Introduction

In order to deal with flow data, it is important to understand its origin and basic topological structure. Different approaches for obtaining flow data as well as some theoretical background of the flow topology and flow simulation are presented in this chapter.

2D flow fields and their topology are presented in order to better understand diffusion and diffusion-reaction processes covered in chapter 3. Since only certain types of first order critical points are considered to be vortex cores, the ability to recognize such features is essential for their processing and extraction. During an iterative process on a 2D vector field, two critical points sometimes collapse into one, or vice versa. These special events are called bifurcations and should also be considered when dealing with 2D flow data.

In order to understand where the idea for the 3D eigenvector vortex extraction method came from, one should also be familiar with first order 3D critical points. If 3D critical points of the saddle focus type are considered, the direction of the vortex core coincides with the direction of the only "real eigenvector" of the Jacobian matrix of that point. The eigenvector method led to the development of the parallel vectors operator used in chapter 4. 3D vector fields specially designed to test vortex extraction methods, such as a helical flow field or bent helical flow field are also presented.

Real-time fluid simulation, used in Chapter 4 to implement and test the real-time vortex extraction methods, is based on simulating the Navier-Stokes equations. These partial differential equations describe the motion of fluid substances. The real-time simulation, and later the vortex detection methods, are implemented on the GPU using the CUDA framework. Parallel vector operator for stationary fields is also implemented on the GPU using the OpenCL framework.

Range flow methods developed in chapter 5 rely on data recorded using special equipment, e.g. Microsoft Kinect device. Such devices produce not only color video sequences, but also, by utilizing an infrared projector, provide depth sequences i.e. data that contains information about distance of objects to the device.

Section 2.2 gives some insights into simulation of the flow data. Analytical examples are presented in Section 2.2.1. Section 2.2.1.1 gives an introduction into the vector field topology important for understanding the structure of the flow. Simulating flow in real-time using the graphic processing units (GPUs) is covered in Section 2.2.2. Section 2.2.2.1 explains the governing Navier-Stokes equations. Section 2.2.2.2 reveals some details of the simulation implementation.

Section 2.3 is about the acquisition of the flow data. Acquisition of range data using the Microsoft Kinect is covered in Section 2.3.1.

The vector field data, techniques and theories presented here are used throughout the rest of the thesis.

2.2 Simulated Data

Simulated flow data in the thesis is produced by using 2D and 3D analytical flow examples or by simulation of the Navier-Stokes equations.

2.2.1 Analytical Vortex Examples

In order to test vortex extraction methods or design processes that preserve vortices, it is necessary to have analytical examples for which the exact solution of the given problem is known. Producing such examples requires certain knowledge about the main features of the vector field i.e. about its topology.

2.2.1.1 Vector Field Topology

First Order Critical Points The main feature of a vector field are its critical points. Definition of a first order critical point is given as: $\mathbf{x_0}$ is a first order critical point of the vector field $\mathbf{v}(\mathbf{x})$ if and only if $\mathbf{v}(\mathbf{x_0}) = \mathbf{0}$ and $|J(\mathbf{x_0})| \neq 0$, where J is the Jacobian matrix of the flow in a point.

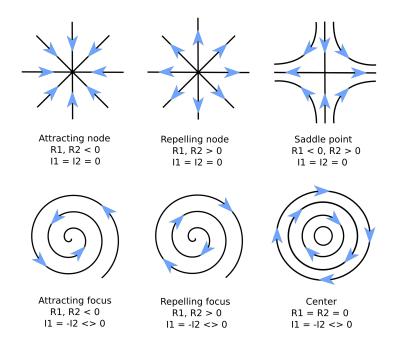


Figure 2.1: First order critical points in 2D linear vector field.

The flow pattern in a 2D vector field around the critical point is characterized by eigenvalues and eigenvectors of the Jacobian matrix in that point and can be

classified (Figure 2.1) as:

```
repelling node: 0 < Re(\lambda_1) < Re(\lambda_2), Im(\lambda_1) = Im(\lambda_2) = 0, attracting node: Re(\lambda_1) < Re(\lambda_2) < 0, Im(\lambda_1) = Im(\lambda_2) = 0, saddle point: Re(\lambda_1) < 0 < Re(\lambda_2), Im(\lambda_1) = Im(\lambda_2) = 0, repelling focus: 0 < Re(\lambda_1) = Re(\lambda_2), Im(\lambda_1) = -Im(\lambda_2) \neq 0, attracting focus: Re(\lambda_1) = Re(\lambda_2) < 0, Im(\lambda_1) = -Im(\lambda_2) \neq 0, center: 0 = Re(\lambda_1) = Re(\lambda_2), Im(\lambda_1) = -Im(\lambda_2) \neq 0.
```

First order critical points in 2D can therefore be divided into a saddle points, nodes, foci and centers. Repelling node and focus can also be called sources, while attracting node and focus are also called sinks. If eigenvalues are imaginary numbers then the critical point (focus or center) can be characterized as a vortex i.e. a swirling critical point.

Separatrices Next to critical points, separatrices are also significant features of the flow. Separatrices are streamlines that separate regions of different flow behavior. In parabolic region all tangent curves originate or end in the critical point. In hyperbolic region all tangent curves go by the critical point, except the two that end/begin in it. In elliptic sector all tangent curves originate and end in the critical point. Each streamline that starts/ends in a critical point which separates two sectors is a separatrix. Special cases of isolated streamlines and stream lines through boundary switch points are also separatrices.

Bifurcations Sudden changes of flow structure at a certain time are called bifurcations. Fold bifurcations is an event where a saddle collapses with a source/sink/center and both of them disappear (Figure 2.2) or a reversed process. Visualizations were made using the Mathematica toolkit ([21]). Hopf bifurcation is switching of the repelling node to an attracting node via center or reversed. Saddle connection is another type of the bifurcation where a separatrix of one saddle ends in a separatrix of another saddle. Collapsing of the separatrix of a saddle into the same saddle is called a blue sky bifurcation. Cyclic fold bifurcation is collapsing of two isolated closed streamlines into each other.

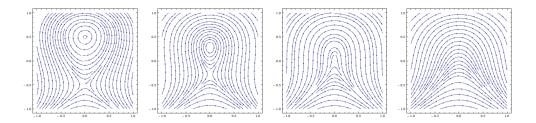


Figure 2.2: Four stages of a fold bifurcation. A saddle and a center collapse into each other and disappear with vector fields ranging from $u(x,y) = \begin{pmatrix} y^2 - 0.25 \\ -x \end{pmatrix}$ to $u(x,y) = \begin{pmatrix} y^2 + 0.25 \\ -x \end{pmatrix}$.

2D Examples Analytical models containing a single critical point are easiest to consider for testing purposes. Jacobian matrices producing respectively saddle, attracting node, repelling node, center, attracting focus, repelling focus are:

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \\ \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}.$$
 (2.1)

Following the fact that linear 2D or 3D vector fields can be simplified as $\mathbf{v} = J\mathbf{x}$ (where \mathbf{x} is a 2D/3D vector and J a constant Jacobian matrix of the flow field), the corresponding vector fields are:

$$\begin{pmatrix} x \\ -y \end{pmatrix}, \qquad \begin{pmatrix} -x \\ -y \end{pmatrix}, \qquad \begin{pmatrix} x \\ y \end{pmatrix}, \\ \begin{pmatrix} y \\ -x \end{pmatrix}, \qquad \begin{pmatrix} y \\ -x-y \end{pmatrix}, \qquad \begin{pmatrix} y \\ -x+y \end{pmatrix}.$$
 (2.2)

Creating vector fields with several critical points is also useful for testing different methods (Figure 2.3).

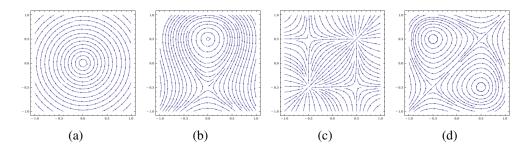


Figure 2.3: Creating a 2D vector field by sampling the analytical function: (a) $u(x,y) = \begin{pmatrix} y \\ -x \end{pmatrix}$, (b) $u(x,y) = \begin{pmatrix} y^2 - 0.25 \\ -x \end{pmatrix}$, (c) $u(x,y) = \begin{pmatrix} x^2 - 0.25 \\ y^2 - 0.25 \end{pmatrix}$, (d) $u(x,y) = \begin{pmatrix} y^2 - 0.25 \\ x^2 - 0.25 \end{pmatrix}$.

3D First Order Critical Points 3D topology is more complicated to consider. In 3D classification of the first order critical points is as follows:

sources:
$$0 < Re(\lambda_1) \le Re(\lambda_2) \le Re(\lambda_3)$$
, repelling saddles: $Re(\lambda_1) < 0 < Re(\lambda_2) \le Re(\lambda_3)$, attracting saddles: $Re(\lambda_1) \le Re(\lambda_2) < 0 < Re(\lambda_3)$, sinks: $Re(\lambda_1) \le Re(\lambda_2) \le Re(\lambda_3) < 0$.

Each of the four cases is subdivided by considering the imaginary parts:

foci :
$$Im(\lambda_1) = 0$$
, $Im(\lambda_2) = -Im(\lambda_3) \neq 0$,
nodes : $Im(\lambda_1) = Im(\lambda_2) = Im(\lambda_3) = 0$,

where $\lambda_1, \lambda_2, \lambda_3$ are the eigenvalues of the Jacobian matrix. Figure 2.4 shows some of the possible first order 3D critical points. Critical points containing a vortex are the ones with two conjugated complex eigenvalues that have opposite signs of real parts when compared to the third eigenvalue i.e. focus saddles. In critical points of the focus saddle type the direction of the vortex core coincides with the direction of the only "real eigenvector" of the Jacobian matrix of that point. The eigenvector method for vortex extraction, which led to the parallel vectors operator, is a generalization of that fact to the entire flow.

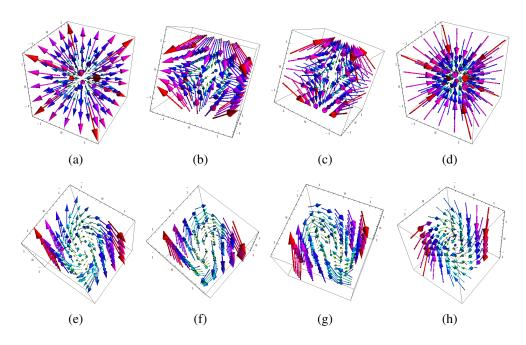


Figure 2.4: First order 3D critical points: (a) source node, (b) repelling saddle node, (c) attracting saddle node, (d) sink node, (e) source focus, (f) repelling saddle focus, (g) attracting saddle focus, (h) sink focus.

3D Examples Jacobian matrices producing respectively (Figure 2.4): source node, repelling saddle node, attracting saddle node, sink node, source focus, repelling saddle focus, attracting saddle focus, sink focus are e.g.:

$$\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}, \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
-1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & 1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
-1 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & -1
\end{pmatrix}, \begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 0 & -1
\end{pmatrix}$$

with corresponding flow fields being:

$$(x,y,z)^T$$
, $(x,y,-z)^T$, $(x,-y,-z)^T$, $(-x,-y,-z)^T$, $(y,-x+y,z)^T$, $(y,-x+y,-z)^T$, $(y,-x-y,z)^T$, $(y,-x-y,-z)^T$.

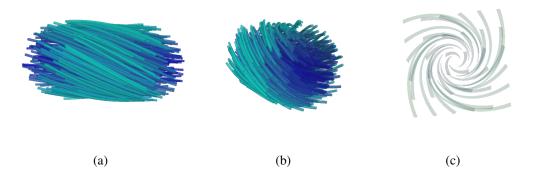


Figure 2.5: Side and front view of the 3D helical vector field. Visualizations were made using the ParaView scientific visualization application ([29]).

Some other common examples of 3D vector fields used for testing are the following. In 2D a center can also be considered as a velocity field of a solid body rotation. Its Jacobian matrix is:

$$\left(\begin{array}{cc}
0 & -\omega \\
\omega & 0
\end{array}\right),$$
(2.5)

where ω is a rotational speed of the rotation of the solid body around the origin. When a constant motion perpendicular to the 2D rotation plane is added a **helical** flow field (Figure 2.5) is obtained. Its Jacobian matrix is:

$$\begin{pmatrix}
0 & -\omega & 0 \\
\omega & 0 & 0 \\
0 & 0 & \gamma
\end{pmatrix},$$
(2.6)

where γ is the speed of the constant motion perpendicular to the rotation plane. Using the speed zero the following vector field is obtained $\mathbf{v}(x,y,z)=(-y,x,1)^T$.

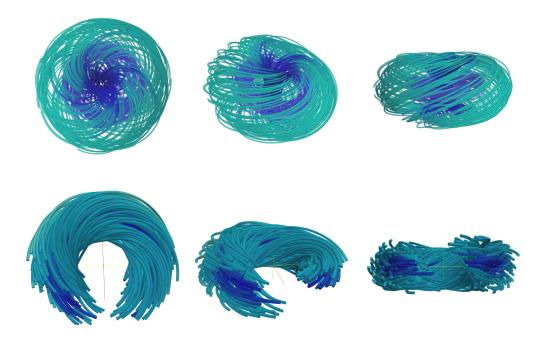


Figure 2.6: Circular helical vector field. Upper row: Side and front view of the bent helical vector field with circular vortex core. Lower row: Side and front view of the bent helical vector field with circular vortex core - a closer look at the streamlines around the core.

In order to construct a bent helical vector field with circular vortex ([37]) (Figure 2.6), a helical vortex is translated and bent around an axis resulting in a following vector field:

$$\mathbf{v} = \begin{pmatrix} -\frac{\omega x z R}{r^2} - \frac{\Omega y}{r} \\ -\frac{\omega y z R}{r^2} + \frac{\Omega x}{r} \\ \left(R - \frac{R^2}{r}\right) \omega \end{pmatrix},$$

where $r=\sqrt{x^2+y^2}$. The vortex core is a circle of radius R around the z axis. ω is the rate of the rotation around the core, Ω is the large-scale rotation around the z axis.

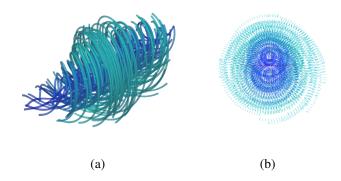


Figure 2.7: Side and front view of a helical vortex with core dislocated from z = -1 to z = 1 via a blending function. The dislocated core is clearly visible in Figure (b).

Helical vortex with "dislocated" core (Figure 2.7) is constructed by using:

$$\mathbf{v} = \begin{pmatrix} y + f(z) \\ -x \\ 1 \end{pmatrix},\tag{2.7}$$

where function

$$f(z) = \begin{cases} -1, & z < -1 \\ x, & -1 \le z \le 1 \\ 1, & z > 1 \end{cases}$$
 (2.8)

is added to one vector field component, such that a vortex core is dislocated away from the plane center by the value of -1 on one side of the spiraling plane, and value of 1 on the other side of the plane, except "near" the center, according to f(z).

2.2.2 Real-Time Flow Simulation on the GPU

Vortex extraction methods are incorporated within a real-time simulation of the flow in Chapter 4. Real-time simulation is implemented by following the paper "Stable Fluids" [46]. It is based on simulating the Navier-Stokes equations.

2.2.2.1 Navier-Stokes equations

Navier-Stokes equations are a set of differential equations that describe the motion of fluid substances. The Navier-Stokes equation for incompressible flow of Newtonian fluids in vector form is:

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} \right) = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}, \tag{2.9}$$

where ρ is the density of flow medium, μ dynamic tenacity, p pressure and \mathbf{f} represents body forces such as gravity or centrifugal force. The meaning of the individual terms in the equation (2.9) is the following:

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v}\right) \qquad \text{inertia,}$$

$$\frac{\partial \mathbf{v}}{\partial t} \qquad \text{unsteady acceleration,}$$

$$(\mathbf{v} \cdot \nabla)\mathbf{v} \qquad \text{convective acceleration,}$$

$$-\nabla p + \mu \nabla^2 \mathbf{v} \qquad \text{divergence of stress,}$$

$$-\nabla p \qquad \text{pressure gradient,}$$

$$\mu \nabla^2 \mathbf{v} \qquad \text{viscosity,}$$

$$\mathbf{f} \qquad \text{other forces.}$$

$$(2.10)$$

The incompressible Navier-Stokes equation (2.9) is a differential algebraic equation (DAE) which has no explicit mechanism for advancing the pressure in time. It is desired for pressure to be eliminated from the equations and thus create a pressure-free Navier-Stokes velocity formulation.

Helmholtz decomposition states that any sufficiently smooth, rapidly decaying vector field in three dimensions can be resolved into the sum of an irrotational (curl-free) vector field and a solenoidal (divergence-free) vector field. The incompressible Navier-Stokes equation (2.9) can be written as a sum of two orthogonal equations, a pressure-free governing equation for the velocity (2.11) and a func-

tional of the velocity related to pressure Poisson equation (2.12):

$$\frac{\partial \mathbf{u}}{\partial t} = \Pi^{S} \left(-(\mathbf{u} \cdot \nabla) \mathbf{u} + \mu \nabla^{2} \mathbf{u} \right) + \mathbf{f}^{S},$$

$$\rho^{-1} \nabla p = \Pi^{I} \left(-(\mathbf{u} \cdot \nabla) \mathbf{u} + \mu \nabla^{2} \mathbf{u} \right) + \mathbf{f}^{I},$$
(2.11)

$$\rho^{-1}\nabla p = \Pi^{I}\left(-(\mathbf{u}\cdot\nabla)\mathbf{u} + \mu\nabla^{2}\mathbf{u}\right) + \mathbf{f}^{I}, \tag{2.12}$$

where are Π^S , Π^I are solenoidal and irrotational projection operators with Π^S + $\Pi^I = 1$, and are \mathbf{f}^S , \mathbf{f}^I the nonconservative and conservative parts of the body force.

Pressure-free Navier-Stokes equation is used to implement the real-time fluid simulation.

Real-Time Fluid Simulation The real-time fluid simulation method presented in paper Stable Fluids [46] was initially designed to move the density particles along a stationary vector field (Figure 2.8). Particles are influenced by the additional density inputed into a density scalar field, they are moved into the direction of the vector field and also there is a certain amount of diffusion (dissipation) happening.

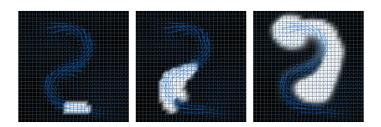


Figure 2.8: Advection i.e. moving of the density (denoted white) through the static vector field (denoted by a blue arrow plot) ([46]).

Pressure-free Navier-Stokes equation for velocity that governs such behavior is given as:

$$\frac{\partial \mathbf{u}}{\partial t} = P\Big(-(\mathbf{u} \cdot \nabla)\mathbf{u} + \mu \nabla^2 \mathbf{u} + \mathbf{f}\Big), \tag{2.13}$$

with projection operator P that projects a vector field onto its divergence-free component. The following notation for density and velocity equations shall be

used for simplicity:

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho + \kappa \nabla^2 \rho + \mathbf{S}, \tag{2.14}$$

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho + \kappa \nabla^2 \rho + \mathbf{S}, \qquad (2.14)$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \mu \nabla^2 \mathbf{u} + \mathbf{f}, \qquad (2.15)$$

where ρ is the density, **u** vector field, **S** additional density, **f** additional force. The meaning of the individual terms in the equation (2.14) is the following:

$$-(\mathbf{u}\cdot\nabla)\rho$$
 movement along the vector field,
$$\kappa\nabla^2\rho$$
 diffusion of the density, (2.16)
 \mathbf{S} input of additional density.

Similar equation is also used for velocity (2.15), where individual terms can be interpreted as:

$$-(\mathbf{u}\cdot\nabla)\mathbf{u}$$
 self-advection,
$$\mu\nabla^2\mathbf{u}$$
 viscous diffusion, (2.17)
$$\mathbf{f}$$
 additional forces.

A simulation of a swirling divergence free (without sources or sinks) vector field was accomplished in the following way. As already mentioned, every velocity field can via Helmholtz decomposition be represented as a sum of a divergence free field and a gradient field. We want to project a vector field onto its divergence free part containing no sinks or sources, but containing many swirls (Figure 2.9). The projection onto a divergence free space is achieved by subtracting the gradient field from the original vector field. The gradient field is obtained by solving a Poisson equation (2.19).

2.2.2.2 **Implementation**

Fluid simulation runs in real-time and has a scalable flow field grid. By using a right mouse click the user can input a desired amount of density into the plane (in 2D) or first slice of the volume (in 3D). Left mouse click adds forces to the vector field (first slice of the vector field in 3D). The forces spread the density around

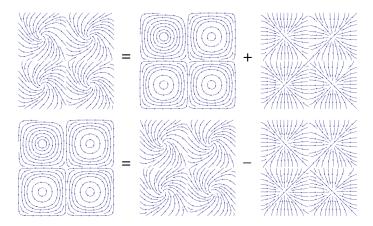


Figure 2.9: Upper row: original vector field is decomposed into a mass conserving field and a gradient field. Lower row: by subtracting the gradient field from the original field we get the visually appealing divergence free field ([46]).

the volume (Figure 2.10). The view can be switched between an arrow plot and the density plot of the vector field. Pseudo-volume rendering i.e. rendering of transparent textures is used to visualize the density of the volume. Simulation is needed in order to test the real-time performance of the vortex extraction methods presented in chapter 4. Quantities indicating a vortex are visualized instead of density. It is possible to pause a simulation and save a stationary flow data. Data can also be saved in regular time periods thus producing a 3D time dependent data.

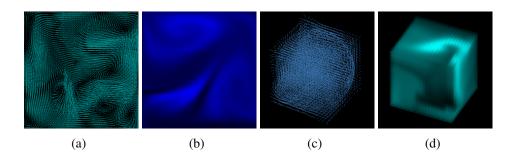


Figure 2.10: 2D and 3D real-time fluid simulation. (a) 2D simulation, arrow plot of the vector field, (b) 2D simulation, density (denoted blue) plot, (c) 3D simulation, arrow plot of the vector field (sphere is to test the camera movement and not related to the simulation), (d) 3D simulation, density (denoted cyan) pseudo-volume.

Implementation Detail Starting point of the implementation is a 2D C code provided by the author of a Stable Fluid paper ([46]). The code was expanded to a 3D CUDA GPU version. The finite grid size is set. The solver is started with an initial state of density and velocity which are then updated according to the environment and newly inputed density/forces. Simulation is based on four functions: source input, diffusion, advection and projection acting as follows:

- Source input adds additional density into the density field or additional forces into the velocity field. The adding of the forces is preformed by the user by using the mouse pointer movement.
- Diffusion procedure is a stable backward diffusion process (given in Appendix A.4) that results in a sparse linear system solved by using a Gauss-Seidel relaxation (2.25).
- Advection procedure forces density/velocity to follow a given velocity field.
 Linear backtracing is used to trace particles back in time. The new density/velocity value it then obtained by bi/trilinear interpolation. Trilinear interpolation is an extension of linear interpolation to 3D (Figure 2.11):

$$\mathbf{v}(x, y, z) = (1 - x)(1 - y)(1 - z)\mathbf{v}_{0,0,0} + x(1 - y)(1 - z)\mathbf{v}_{1,0,0} + (1 - x)y(1 - z)\mathbf{v}_{0,1,0} + (1 - x)(1 - y)z\mathbf{v}_{0,0,1} + xy(1 - z)\mathbf{v}_{1,1,0} + (1 - x)yz\mathbf{v}_{0,1,1} + x(1 - y)z\mathbf{v}_{1,0,1} + xyz\mathbf{v}_{1,1,1}.$$
(2.18)

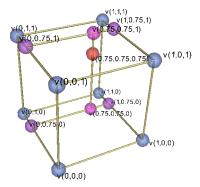


Figure 2.11: Trilinear interpolation of the vector field \mathbf{v} in a 3D cell. The value of the flow field at the location of the red "point" is obtained using (2.18).

Projection routine forces velocity field to be mass conserving. Visually it
produces a swirly flow without sources or sinks. Mass conserving i.e. divergence free vector field is obtained by subtracting the gradient field from the
original vector field. The gradient field is obtained by solving the Poisson
equation:

$$\Delta \mathbf{u} = \mathbf{f}.\tag{2.19}$$

Poisson equation is a large sparse linear system. For example, in 2D the equation (2.19) is rewritten as:

$$\mathbf{f} = \partial_{xx} u + \partial_{yy} u, \tag{2.20}$$

$$\mathbf{f} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}, \quad (2.21)$$

$$h^{2}\mathbf{f} = u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j},$$
 (2.22)

and yields a system $A\mathbf{u} = \mathbf{b}$, where a system matrix is of the form:

$$A = \begin{pmatrix} D & -I & 0 & \cdots & & & 0 \\ -I & D & -I & 0 & \cdots & & 0 \\ 0 & -I & D & -I & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -I & D & -I & 0 \\ 0 & & \cdots & 0 & -I & D & -I \\ 0 & & & \cdots & 0 & -I & D \end{pmatrix}, \quad (2.23)$$

where I is the identity matrix, and D of the following form:

$$D = \begin{pmatrix} 4 & -1 & 0 & \cdots & & & 0 \\ -1 & 4 & -1 & 0 & \cdots & & 0 \\ 0 & -1 & 4 & -1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 4 & -1 & 0 \\ 0 & & \cdots & 0 & -1 & 4 & -1 \\ 0 & & & \cdots & 0 & -1 & 4 \end{pmatrix}. \tag{2.24}$$

The Poisson equation is solved by using a red-black Gauss-Seidel scheme:

$$u_{i,j}^{k+1} = \frac{u_{i+1,j}^k + u_{i-1,j}^{k+1} + u_{i,j+1}^k + u_{i,j-1}^{k+1}}{4} - \frac{h^2 \mathbf{f}}{4}$$
 (2.25)

Red-black denotes an alternating usage of pixel grid in each iteration i.e. a grid is divided into a red-black chess-like pattern and only red or only black pixels are used in one iteration.

These four routines combined produce a fluid simulation. Density is simulated by iteration of adding additional density, diffusing and advecting. Velocity is simulated by iteration of adding force, diffusing, advecting, projecting. Boundary conditions are set so that the velocity component perpendicular to the cell border is zero. The routines are implemented as CUDA kernels operating in parallel on GPU. OpenGL was used for visualization of vector field/density/vortex measures. Additional GPU routines are implemented for vortex extraction in Chapter 4. They are visualized by connecting the vortex indicators to the color of the volume/arrow plot.

CUDA Implementation and Optimization The simulation is implemented by using OpenGL and nVidia CUDA and is running on the GPU. nVidia CUDA (Compute Unified Device Architecture) is a parallel computing architecture developed by nVidia for graphics processing. C for CUDA (C with nVidia extensions and certain restrictions) is used to code algorithms for execution on the graphics processing units (GPUs). The main feature of using CUDA (or OpenCL) is the ability of parallel computation on many cores allowing very large speedups if used properly.

Optimizing code for GPU requires certain knowledge about the hardware architecture and is sometimes contraintuitive. Main opportunities for optimization (other than changing the algorithm) lie in **optimizing the grid/block sizes** and **optimizing the memory access**.

CUDA computation is organized in a grid. A grid has several blocks which have a a fixed number of threads (Figure 2.12). GPU kernels can be run using different block and grid sizes. A grid has the same dimension as the blocks in it,

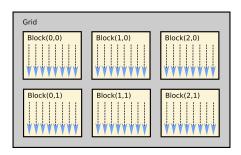


Figure 2.12: CUDA grid containing a number of blocks which contain a number of threads.

usually 1,2 or 3. Grid and block sizes are specified during the launch of the kernel. Choosing the optimal size has a great influence on the speed of the execution. The design of problem and the GPU hardware determine the optimal setup. Number of treads in a block should be a multiple of 32 (multiple of warp size). Most of the kernels used in the flow simulation use the 32×32 block size with 64 threads per block. The simulation is performed using nVidia GeForce GTX 285 graphics card.

Memory management is extremely important when programming CUDA and OpenCL kernels (Section 4.3). Choosing the "right" memory for the task (global, shared, texture etc.) and dimensioning the data (padding) to get the consecutive chunks of memory thrown onto a kernel (coalescent reads) helps to optimize the task. Parallel vectors operator used in Section 4.3 pads the loaded data with zeros to achieve coalescent reads from memory. In the real-time simulation the size is chosen by the user and can easily be set to multiples of 32.

Further speedup of the algorithm could be achieved by taking care of the OpenGL and CUDA interoperability. By taking into account the ability of OpenGL to render directly from the device memory, the unnecessary copying from device to host can and back can be avoided. OpenGL buffers are mapped into the CUDA address space and then used as global memory. This is done via Vertex Buffer Objects (VBO) or Pixel Buffer Objects (PBO).

Recorded Data 25

2.3 Recorded Data

Flow data measured from different sources is essential for testing performance of the detection methods within the flow. Data can be obtained from different sources such as flow inside mechanical parts, blood flows, flow around a plane, around an insect, hurricane data etc. The quality of so obtained data is sometimes low and might require preprocessing.

2.3.1 Microsoft Kinect

Kinect is a multi sensing device by Microsoft. It consists of a RGB camera, and a depth sensor, allowing inclusion of the third dimension into different algorithms. Depth sensor is an infrared projector that projects light grid onto the environment. Depth images recorded with the Kinect contain significant invalid areas and unstable edges (Figure 2.13) which are a consequence of occlusions which occur because of the shift between the source of active illumination and the infrared camera, caused by the *structured light depth estimation* approach [1], [51].

Data obtained by Kinect is used in Chapter 5 to record human gestures. Five volunteers preformed nine gestures each. Each gesture data consists of 60 frames of color and depth images. Color and depth data is first calibrated by using backward warping. Post processing technique is developed to get rid of the input data artifacts (Section 5.3). Ensemble range flow of the gesture videos is created (Section 5.4) and finally, vortex cores are detected within created gesture ensemble range flow in order to aid gesture classification (Section 5.5).

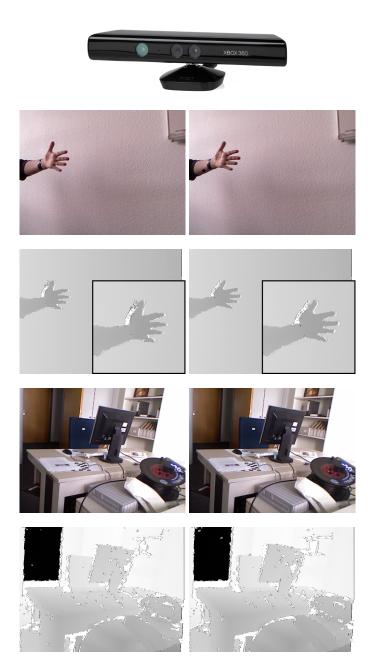


Figure 2.13: Upper row: Microsoft Kinect device ([24]). Second row: two consecutive image frames (color channels) of a scene that shows a human hand moving right, up and back. Third row: depth channels of the images above (with close-up frame of the hand) obtained by Kinect. The depth images are aligned to the corresponding color channels using backward warping. Invalid areas (white) are visible in depth images. Fourth row: color channels of a passive scene of an office with camera moving towards the table. Last row: depth channels of the images above.

Chapter 3

Diffusional and Variational Vortex Preserving

3.1 Introduction

Diffusion and diffusion-reaction processes in which the swirling motion i.e vortices are preserved will be presented in this chapter. Diffusion processes produce simplified, information-reduced output. Such processes are utilized to preserve vortex areas and diminish the rest of the flow field. Vector fields with emphasized swirling areas i.e 2D swirling critical points (vortices) are wanted as the result. Other types of features should be diminished and no new features should be created. Several processes are designed in order to preserve vortical structures. The discriminant d of the characteristic polynomial of the Jacobian matrix is used as an indicator of a swirling area. Nonlinear isotropic diffusion with diffusivity steered by d is used to process the vector fields. Detecting vortices or vortex regions and preserving a circular area around them is another option for obtaining the wanted vector fields. Variational processes which give similar results are also presented here.

Section 3.2 explains diffusion process and its discretization. Section 3.3 gives an introduction into the variational energy minimization. Section 3.4 introduces diffusion and diffusion-reaction processes designed to preserve the swirling

i.e vortical structures. Nonlinear isotropic diffusion processes steered either by discriminant d or by the location of the vortex cores/region are presented. Binary and continuous diffusivities are tested. Some processes are based on utilizing the discriminant d of the characteristic polynomial of the Jacobian matrix as an indicator of the existence of swirling structures i.e. vortex cores. As an alternative approach, a circular area around detected vortex cores or vortex regions is preserved while the rest of the flow is blurred. These processes can also be expressed as a nonlinear isotropic diffusion with binary diffusivity based on the location of vortex cores/regions. Variational methods that preserve swirling structures are presented in Section 3.4.2.

The processes produce vector fields with emphasized i.e. preserved swirling and vortex structures. Other kinds of critical points are diminished, however, their destruction cannot be guaranteed. Resulting flow fields are suitable for a alternative insight into the flow structure, especially when one is interested in study of vortices. They can also be used as a processing step within the presented real-time techniques (Chapter 4, Section 4.4.1).

Further improvements include determination of optimal parameters, using advanced anisotropic regularization and formulation of algorithms in 3D. Non-iterative fast numerical schemes could be used to speed up the algorithms up to the real-time performance.

3.2 Diffusion

Prior to introduction of vortex area preserving diffusion in Section 3.4.1, basic theory and classification of diffusion processes will be presented in this section.

The diffusion or heat equation describes a process that equilibrates concentration and preserves mass. It can be described by the following equation:

$$\partial_t u = \operatorname{div}(D\nabla u). \tag{3.1}$$

Different diffusion processes are suitable for various applications. In 2D image processing linear isotropic diffusion is sufficient for simple blurring. Edge enhancing diffusion can be more suitable for e.g denoising an image, while e.g. fingerprint recognition can benefit from coherence enhancing diffusion.

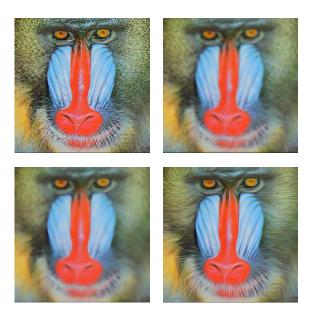


Figure 3.1: Diffusion of a 2D image. (a) original image, (b) linear isotropic diffusion i.e. Gaussian blurring ([15], [56]), (c) nonlinear isotropic diffusion (Perona-Malik [32]), (d) coherence enhancing anisotropic diffusion (using joint color channels in structure tensor) ([53]).

Different processes can be formulated as follows (Figure 3.1):

• Linear isotropic diffusion ([15], [56]):

$$\partial_t u = \operatorname{div}(1 * \nabla u) = \Delta u, \tag{3.2}$$

• Nonlinear isotropic diffusion ([32]):

$$\partial_t u = \operatorname{div}(g(|\nabla u|^2)\nabla u),$$
 (3.3)

• Nonlinear anisotropic diffusion ([52]):

$$\partial_t u = \operatorname{div}(D(J_\rho(\nabla u_\sigma))\nabla u). \tag{3.4}$$

Following sections will tangle each of them. Finite difference derivative approximations and boundary conditions are explained in Appendix A.1 and A.2.

Linear Isotropic Diffusion Linear diffusion process can be expressed as:

$$\partial_t u = \Delta u,$$

$$u(x,0) = f(x),$$
(3.5)

and is equivalent to Gaussian convolution. It has the unique solution:

$$u(x,t) = \begin{cases} f(x) & (t=0) \\ (K_{\sqrt{2t}} * f)(x) & (t>0) \end{cases},$$
(3.6)

where K_{σ} is a Gaussian with standard deviation σ :

$$K_{\sigma}(x) := \frac{1}{2\pi\sigma^2} \exp\left(-\frac{|x|^2}{2\sigma^2}\right). \tag{3.7}$$

The unique solution depends continuously on the initial image f (well-posedness). The evolving image satisfies the minimum-maximum principle:

$$\inf_{\mathbb{R}^2} f \le u(x,t) \le \sup_{\mathbb{R}^2} f \qquad \forall x, \forall t > 0.$$
 (3.8)

Other theoretical results include average grey level invariance, Lyapunov sequences (i.e. transformation is simplifying, information-reducing) and convergence to a constant steady state. Mathematical formulation leading to a discrete explicit scheme is given in Appendix A.3.

Nonlinear Isotropic Diffusion Nonlinear isotropic diffusion process can be written as:

$$\partial_t u = \operatorname{div}(g(|\nabla u|^2)\nabla u),$$
 (3.9)

where $g(|\nabla u|^2)$ is a diffusivity function governing the nature of the diffusion. Contrast parameter λ that separates forward and backward diffusion needs to be set. The function is formed so that the smoothing (forward diffusion) happens for $|\nabla u| < \lambda$ and edge-enhancing (backward diffusion) for $|\nabla u| > \lambda$.

Some usual diffusivities are:

Perona-Malik diffusivity:
$$g(s^2) = \frac{1}{1 + \frac{s^2}{\lambda^2}}$$
, (3.10)

Charbonnier diffusivity:
$$g(s^2) = \frac{1}{\sqrt{1 + \frac{s^2}{\lambda^2}}},$$
 (3.11)

Exponential diffusivity:
$$g(s^2) = \exp\left(\frac{-s^2}{2\lambda^2}\right)$$
. (3.12)

Properties of function g are g>0, $g\in C^{\infty}$, g(0)=1, g decreasing on $[0,\infty)$, $\lim_{s^2\to\infty}g(s^2)=0$. Mathematical formulation leading to a discrete explicit scheme is given in Appendix A.5.

Nonlinear Anisotropic Diffusion Nonlinear anisotropic diffusion process can be written as:

$$\partial_t u = \operatorname{div} \left(D(\nabla u) \nabla u \right)$$

$$= \operatorname{div} \left(\begin{pmatrix} v_{1,x} & v_{2,x} \\ v_{1,y} & v_{2,y} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} v_{1,x} & v_{1,y} \\ v_{2,x} & v_{2,y} \end{pmatrix} \nabla u \right), \tag{3.13}$$

where v_1, v_2 are eigenvectors such that $v_1 || \nabla u, v_2 \bot \nabla u$ and λ_1, λ_2 eigenvalues chosen such that $\lambda_2 = 1$ denotes full diffusion along the edge, and $\lambda_1 = g(|\nabla u|^2)$ adjustable diffusion across the edge so forming an **edge enhancing diffusion**.

For applications that require a more general structure direction description, the matrix $\nabla u \nabla u^T$ can be used instead of ∇u :

$$\partial_{t} u = \operatorname{div} \left(D(K * \nabla u \nabla u^{T}) \nabla u \right)
= \operatorname{div} \left(D \left(K * \begin{pmatrix} u_{x} u_{x} & u_{x} u_{y} \\ u_{x} u_{y} & u_{y} u_{y} \end{pmatrix} \right) \nabla u \right)
= \operatorname{div} \left(\begin{pmatrix} v_{1,x} & v_{2,x} \\ v_{1,y} & v_{2,y} \end{pmatrix} \begin{pmatrix} \lambda_{1} & 0 \\ 0 & \lambda_{2} \end{pmatrix} \begin{pmatrix} v_{1,x} & v_{1,y} \\ v_{2,x} & v_{2,y} \end{pmatrix} \nabla u \right),$$
(3.14)

where v_1, v_2 are eigenvectors of the $K*\nabla u\nabla u^T$ and eigenvalues λ_1, λ_2 are chosen

Variational Methods 32

such that the smoothing increases with structure coherence i.e.:

$$\lambda_1 = \alpha, \tag{3.15}$$

$$\lambda_2 = \begin{cases} \alpha & \mu_1 = \mu_2 \\ \alpha + (1 - \alpha) \exp\left(\frac{-C}{(\mu_1 - \mu_2)^2}\right) & else \end{cases} , \tag{3.16}$$

where $\alpha > 0$ is a small value, μ_1, μ_2 eigenvalues of the structure tensor $K * \nabla u \nabla u^T$ and C contrast parameter. Such **coherence enhancing diffusion** has a better sense of "overall direction" since, due to the structure tensor, the cancellation of gradients with opposite orientations in not an issue. Mathematical formulation leading to a discrete explicit scheme is given in Appendix A.7.

3.3 Variational Methods

Prior to introduction of vortex area preserving variational methods in Section 3.4.2, variational processes will be presented in this section.

Minimizing an energy functional of the form:

$$E_f(u) = \int_{\Omega} \left((u - f)^2 + \alpha \Psi \left(|\nabla u|^2 \right) \right) dx, \tag{3.17}$$

is an elegant way of describing a minimization model and its constraints. The first term, known as data or similarity term is rewarding the similarity to the original image f, while the second term, know as smoothness term, regulariser or penaliser, penalises deviations from smoothness.

If energy E_f is minimized by function v, then v satisfies **Euler-Lagrange equations**. A minimizer of the 2D functional:

$$E(u) = \int F(x, y, u, u_x, u_y) dx dy$$
 (3.18)

necessarily satisfies the Euler-Lagrange equation:

$$0 = F_u - \partial_x F_{u_x} - \partial_y F_{u_y}. \tag{3.19}$$

Variational Methods 33

By using the notation:

$$F = \left((u - f)^2 + \alpha \Psi \left(|\nabla u|^2 \right) \right),$$

$$F_u = 2(u - f),$$

$$F_{u_x} = 2\alpha u_x \Psi' \left(|\nabla u|^2 \right),$$

$$F_{u_y} = 2\alpha u_y \Psi' \left(|\nabla u|^2 \right),$$

$$(3.20)$$

the following equation is obtained:

$$0 = (u - f) - \alpha \partial_x \left(\Psi' \left(|\nabla u|^2 \right) u_x \right) - \alpha \partial_y \left(\Psi' \left(|\nabla u|^2 \right) u_y \right) \text{ giving}$$

$$0 = \operatorname{div} \left(\Psi' (|\nabla u|^2) \nabla u \right) - \frac{u - f}{\alpha}. \tag{3.21}$$

A choice of how to discretize has to be made.

Discretizing the Euler-Lagrange equations If Euler-Lagrange equations are discretized a non-linear system of equations is obtained. Note that the same system is obtained if discrete energy functional and its minimizing condition (vanishing gradient) is considered directly, without using the Euler-Lagrange equations (approach used in graphical models):

$$E(u) = \sum_{i=1}^{N} (u_i - f_i)^2 + \alpha \sum_{i=1}^{N-1} \Psi\left(\frac{(u_{i+1} - u_i)^2}{h^2}\right), \quad (3.22)$$

$$\frac{\partial E}{\partial u_i} = 0, \quad \text{for } i = 1, \dots, N , \qquad (3.23)$$

where $f = (f_1, \dots, f_N)^T$ is the signal to be restored, $u = (u_1, \dots, u_N)^T$ minimizer of E(u). The system is then solved by using some root finding algorithm.

Euler-Lagrange equations as Parabolic PDEs Another option for discretization is formulating and solving the Euler-Lagrange equation (3.21) as a steady state of a parabolic partial differential equation i.e. forming a diffusion-reaction system:

$$\frac{\partial u}{\partial t} = \operatorname{div}\left(\Psi'(|\nabla u|^2)\nabla u\right) - \frac{u - f}{\alpha}.$$
(3.24)

Such diffusion-reaction system can then be solved in a similar fashion as a diffusion process. Mathematical formulation leading to a discrete explicit scheme is given in Appendix A.8.

3.4 Vortex Preserving Diffusion and Variational Processes

Diffusion based vortex preserving processes are designed in this section. The goal is a vector field in which vortex cores are preserved and rest of the flow dampened. The destruction of other types of features is desired, however, no new features should be created.

Methods for smoothing the vector field that pay attention to the topology of the field have been designed before. Papers like [3] rely on complex combinatorial topology to process the vector field ([36]). In [27], user is expected to choose the features to be preserved. Here, an automatic method is designed in which the emphasis is on vortex preserving.

Figure 3.2 shows the color coding used for depicting the flow magnitude. Arrow plots are also used to obtain a better insight into the flow behavior.

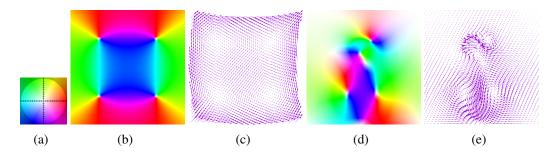


Figure 3.2: (a) color coding used to depict the vector direction (color) and magnitude (saturation), (b) color coded magnitude of a analytical test flow field with two centers and two saddles, (c) arrow plot of the vector field from (b) (also in Figure 2.3 (d)). (d) color coded magnitude of a flow field saved from a paused 2D real-time simulation, (e) arrow plot of the vector field from (d).

Section 3.4.1 presents different diffusion methods for preserving the swirling features. Nonlinear isotropic diffusion is used with different diffusivities designed to preserve vortices. Diffusivities are based on using the discriminant d of the characteristic polynomial of the Jacobian matrix in each point of the flow.

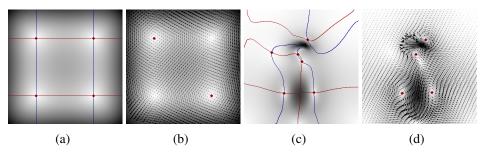


Figure 3.3: Critical points detected in the flow fields from Figure 3.2. (a), (c) Critical points detected as intersection of curves where flow components change sign. (b), (d) Vortex cores i.e. critical points where d < 0 i.e. swirling critical points, foci or centers (denoted as red diamonds).

In 2D stationary vector fields, the vortex/swirling critical point exists if the eigenvalues of the Jacobian matrix are two conjugated complex values i.e. if the discriminant of the quadratic equation:

$$\lambda_{1,2} = \frac{trJ \pm \sqrt{tr^2J - 4|J|}}{2},\tag{3.25}$$

where
$$J=\begin{pmatrix}u_x&u_y\\v_x&v_y\end{pmatrix}$$
, is less than zero i.e. $d=\begin{pmatrix}(u_x-v_y)^2+4u_yv_x\end{pmatrix}<0$.

Figure 3.3 shows critical points detected in 2D vector fields from Figure 3.2. The critical points are detected by locating the intersections of the curves where the components of the vector field change direction. In (b) and (d) only the points where d < 0 i.e. vortices are shown. Figure 3.4 shows the areas in the test vector fields where the quantity d < 0.

Binary diffusivity, binary diffusivity with additional edge blurring and continuous diffusivities are presented. Swirling area indicated by the negative discriminant d does not guarantee that there is a critical point within it. To always obtain only areas around vortices, binary diffusivities based on vortex (core or region) detection are also presented. Diffusion-reaction processes are presented in Section 3.4.2. Resulting flow fields are smoothed, but have emphasized/preserved

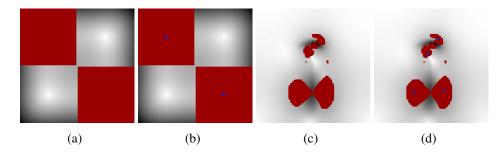


Figure 3.4: (a), (c) Areas in test vector fields where discriminant d < 0 denoted red, (b), (d) Swirling areas together with swirling critical points denoted blue.

swirling features i.e. vortices.

3.4.1 Vortex Preserving Diffusion

A diffusion processes are designed with diffusivity function that avoids swirling features or vortices. Nonlinear isotropic diffusion process is taken:

$$\partial_t u = \operatorname{div}(g(|\nabla u|^2)\nabla u),$$
 (3.26)

with function g being a diffusivity function governing the nature of the diffusion (see Section 3.2, Appendix A.5).

Wanted properties for the function g are that it is positive g>0 and smooth $g\in C^\infty[0,\infty)$. These properties ensure that the diffusion process is well-posed and regular, the average gray level is preserved, the minimum-maximum principle respected, that the process is simplifying and information-reducing, and that for $t\to\infty$ the image converges to the average gray level (average flow magnitude).

Selecting a diffusivity function which equals zero for certain cases, results in a process that does not converge to the average steady state. Flow areas for which g=0 do not get diffused, so for $t\to\infty$ the process converges to the average magnitude outside these areas, while the flow inside stays unchanged. A small variance of the flow magnitude within the blurred area is a good stopping criteria for all presented methods.

Binary Diffusivity Diffusivity g is set to strong in a non-swirling area (where $d \ge 0$) and set to weak where there are swirling features (d < 0). Following

diffusivity is constructed:

$$g(s^2) = \begin{cases} 0, & d < 0 \\ 1, & d \ge 0 \end{cases}$$
 (3.27)

Note that for g=1 nonlinear isotropic diffusion equals the linear isotropic diffusion i.e. Gaussian blurring. Figure 3.5 shows the diffusion process on the two test vector fields for different number of iterations.

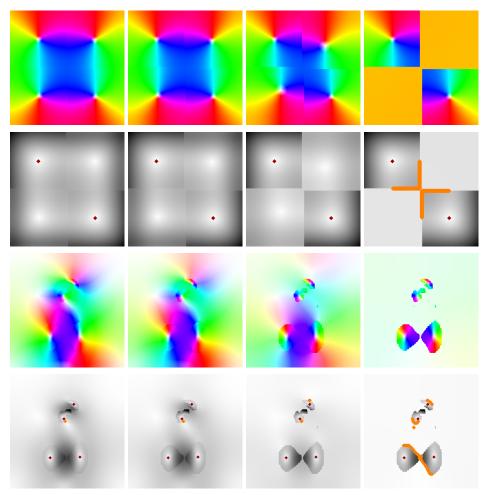


Figure 3.5: Nonlinear isotropic diffusion process with diffusivity (3.27) on an analytical test flow field (upper two rows) and simulated test flow field (lower two rows) for iteration number: 500, 1000, 5000, 50000 (left to right). Upper, third row: color coded magnitude of the flow. Second, lower row: gray scale flow magnitude with vortex cores. Red critical points are the points originally present in the vector field. Orange critical points are newly created.

Figure 3.6 shows the arrow plots of the processed flow fields for high iteration number (50000).

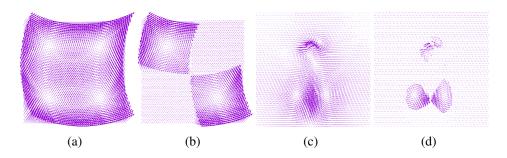


Figure 3.6: (a), (c) arrow plots of the two test vector fields, (b), (d) arrow plots of the diffused vector fields from Figure 3.5 right (50000 iterations).

Swirling structures are preserved and the rest of the field blurred, however, new critical points are introduced at the borders of the swirling and non-swirling ares (orange points in Figure 3.5).

Additional Blurring of the Swirling Structure Edges Newly introduced critical points need to be removed. They appear at the places where discriminant d changes sign. Figure 3.7 shows the areas where $d > -\epsilon$, where ϵ is small. The ϵ threshold chooses the area where d > 0 plus edge of the d < 0 area (Figure 3.4).

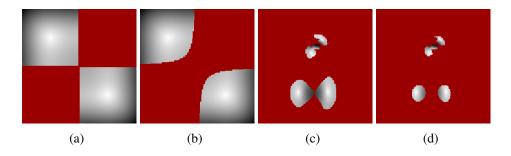


Figure 3.7: Areas where discriminant (a), (c) d > -0.0001 and (b), (d) d > -0.01. The area where d > 0 is chosen plus the edge of the d < 0 area (Figure 3.4).

Additional Gaussian blurring of the area with d > -0.01 is performed after the diffusion process with the diffusivity (3.27). This removes the created invalid critical points. Figure 3.8 shows such process.

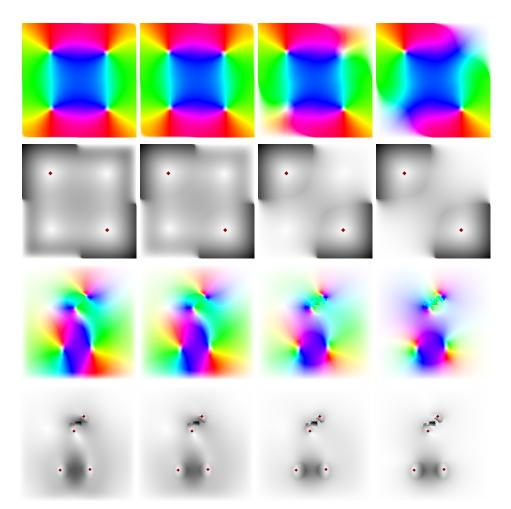


Figure 3.8: Nonlinear isotropic diffusion process with diffusivity (3.27) followed by blurring for iteration number: 500, 1000, 5000, 50000 (left to right). Upper, third row: color coded magnitude of the flow. Second, lower row: gray scale flow magnitude with vortex cores.

Figure 3.9 shows arrow plots of the processed flow fields for high iteration number. Swirling features are preserved and no new features are introduced. Sharp boundaries between the swirling and non swirling areas are blurred.

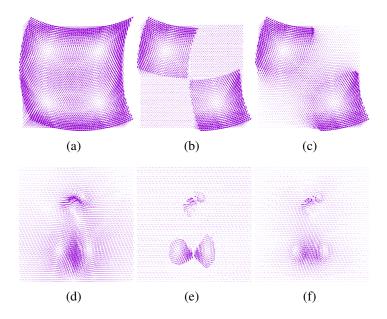


Figure 3.9: (a), (d) arrow plots of the two test vector fields, (b), (e) arrow plots of the diffused vector fields from Figure 3.5 right (50000 iterations), (c), (f) arrow plots of the diffused and additionally blurred fields from Figure 3.8 right (50000 iterations).

Continuous Diffusivity Previous process first blurs the area where d>0, and then, because the sharp blurring border introduces additional critical points, the area where d>-0.01 is blurred. This removes the introduced artifacts. Alternative solution is designing a diffusivity without sudden jumps. Following diffusivity is now considered:

$$g(s^2) = \begin{cases} 0, & d < 0 \\ d, & d \in [0, 1] \\ 1, & \text{else} \end{cases}$$
 (3.28)

where discriminant $d = \left((u_x - v_y)^2 + 4u_yv_x\right)$. This results in an graduate increase of blurring amount, from no blurring within the swirling area to full blurring (g=1) away form the swirling area. During the diffusion process, additional blurring iteration is made within the non-swirling area to further blur it out. Figure 3.10 shows the value of the discriminant of the two test vector fields mapped to grayscale. Sudden jumps in values are not excluded. Figure 3.11 shows the diffusion process. Figure 3.12 compares the arrow plots of this and two previous approaches.

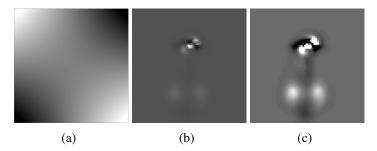


Figure 3.10: Value of the discriminant d of the two test vector fields mapped to grayscale. Negative values start from white, positive end with black. (a) analytical example, (b) frame from the real-time simulation test data, (c) middle image with adjusted contrast and brightness.

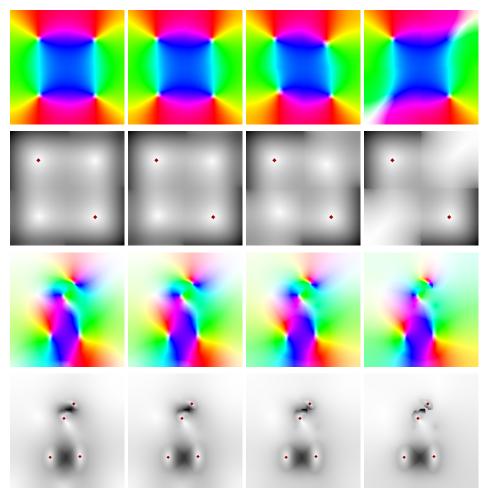


Figure 3.11: Nonlinear isotropic diffusion process with diffusivity (3.28) for iteration number: 500, 1000, 5000, 50000 (left to right). Upper, third row: color coded magnitude of the flow. Second, lower row: gray scale flow magnitude with vortex cores.

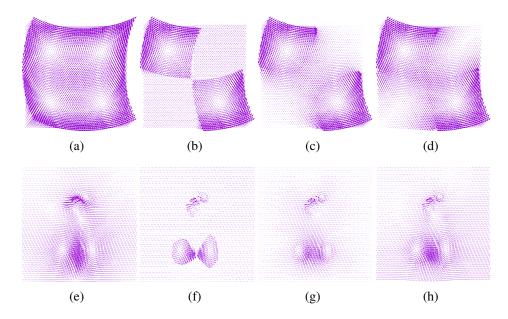


Figure 3.12: (a), (e) arrow plots of the two test vector fields, (b), (f) arrow plots of the diffused vector fields (3.27) from Figure 3.5 right (50000 iterations), (c), (g) arrow plots of the diffused and additionally blurred fields from Figure 3.8 right (50000 iterations). (d), (f) arrow plots of the diffused vector fields (3.28) from Figure 3.11, right (50000 iterations).

Shown processes output the vector fields with emphasized swirling features. No new swirling features are introduced, however, it is not always guaranteed that all other features are destroyed. Figure 3.13 shows the resulting vector fields with all critical points denoted. The diffusion with diffusivity (3.27) destroys all other features, but also introduces new critical points. Approach with additional blurring after the diffusion and the diffusion with diffusivity (3.28) do not guarantee the destruction of the non-swirling critical points. They do, however, emphasize the swirling features and produce relatively smooth vector fields without introducing new swirling features.

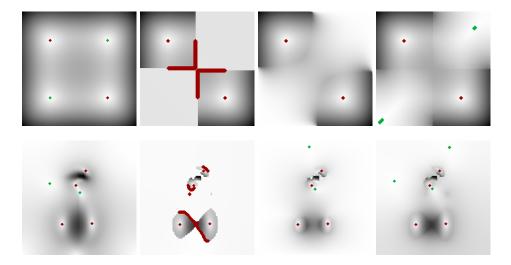


Figure 3.13: All critical points within the two processed vector fields with 50000 iterations. Swirling critical points are denoted red, non-swirling critical points green. Upper row: analytical flow field, lower row: frame from the real-time simulation. Left column: initial vector fields, Second column: diffusion with diffusivity (3.27), Third column: diffused with diffusivity (3.27) plus additional blurring, Right column: diffusion with diffusivity (3.28).

Binary Diffusivity Based on Vortex Detection Presented diffusion approaches are steered by the discriminant d of the characteristic polynomial of the Jacobian matrix. Areas where d < 0 do not always nicely indicate an area around vortex cores. There is also no guarantee that there is a swirling critical point present within the swirling area. This results in seemingly random patches of flow which are not smoothed. Figure 3.14 shows such an example. It represents another vector field saved from a 2D real-time simulation processed with nonlinear isotropic diffusion with diffusivity (3.27). Although the swirling areas are preserved, it is not the result one would necessarily desire as a field with emphasized vortices.

An alternative approach is to detect vortex cores, and to not blur the vector field in areas around them. This process could also be considered a nonlinear isotropic diffusion process with binary diffusivity steered by the location of vortex cores i.e. with diffusivity:

$$g(s^2) = \begin{cases} 0, & \text{areas around vortex cores} \\ 1, & \text{else} \end{cases}$$
 (3.29)

Figure 3.15 shows the result of such process for two different area sizes. Resulting flow fields have preserved swirling areas around the vortex cores. There are no newly created critical points.

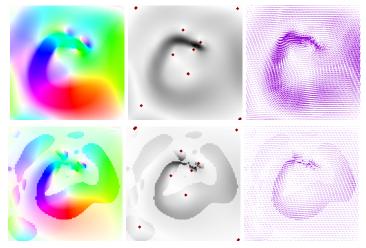


Figure 3.14: Nonlinear isotropic diffusion process with diffusivity (3.27) on a simulated flow field. Swirling areas without swirling critical points remain unblurred.

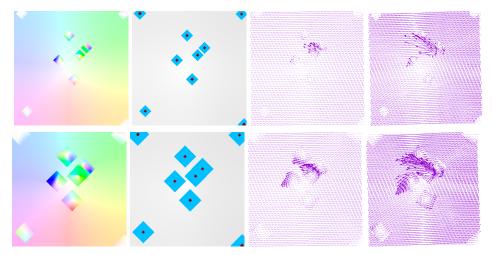


Figure 3.15: Nonlinear isotropic diffusion process with diffusivity (3.29) on a simulated flow field (same as in figure 3.14) for 50000 iterations. Upper row: smaller area surrounding the vortex cores. Lower row: bigger area surrounding the vortex cores. For better visibility, right column shows arrow plots of the vector field with increased magnitude. First column: color coded magnitude of the vector field. Second column: magnitude of the vector field together with swirling critical points (red) and area surrounding them (blue).

Considering vortex region detection gives another option for defining the vortex areas to be preserved. Q criterion for vortex region detection is taken as a vortex indicator ([14], Section 4.2.1). Areas where the vorticity tensor $A = \frac{J-J^T}{2}$ dominates the rate of strain tensor $S = \frac{J+J^T}{2}$ are considered (J Jacobian matrix). This is achieved by looking at the ratio of the Euclidean norm of the two matrices. The diffusivity is then defined as:

$$g(s^2) = \begin{cases} 0, & \text{areas around points where } \frac{|A|^2}{|S|^2} > 1\\ 1, & \text{else} \end{cases}$$
 (3.30)

Figure 3.16 shows the test vector fields processed with diffusion (3.30). In practice an area around $|A|^2 > |S|^2 + \epsilon$ with small ϵ is chosen to get the preserved vortex area (Figure 3.16, upper row).

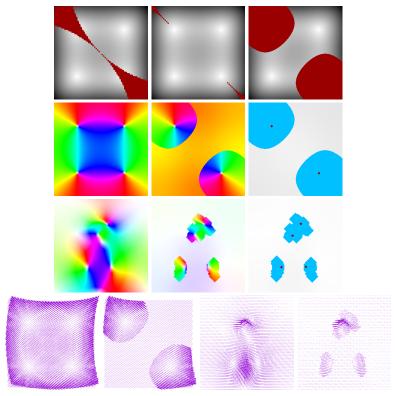


Figure 3.16: Nonlinear isotropic diffusion process with diffusivity (3.30) on test flow fields for 50000 iterations. Upper row: left: area where $|A|^2 > |S|^2$, middle: area where $|A|^2 > |S|^2 + \epsilon$, right: area surrounding $|A|^2 > |S|^2 + \epsilon$. Second, third row: diffusion on test flow fields. Left: original magnitude, middle: magnitude of the diffused vector field, right: magnitude of the vector field together with swirling critical points (red) and area surrounding them (blue). Lower row: arrow plots of the shown vector fields.

3.4.2 Variational Formulation of Vortex Preserving

Discriminant Steered Energy Functional Requesting Similarity to the Original Vector Field Variational approach that produces vector fields with preserved swirling features is wanted. The processed data should be as similar as possible to the initial data within the vortex areas whereas other locations within the flow should be destroyed. For that purpose the following energy functional is used:

$$E(u,v) = \int (H((u - u_{orig})^2 + (v - v_{orig})^2) + \alpha \Psi(|\nabla u|^2 + |\nabla v|^2)) dx dy, \quad (3.31)$$

where H is a function steering the diffusion and u_{orig} , v_{orig} initial vector field components. Mathematical formulation leading to a discrete explicit scheme is given in Appendix A.9. H needs to impose high similarity to the data within the swirling areas and allow low similarity outside them. H is chosen as:

$$H_{i,j} = \begin{cases} -a & d < 0 \\ b & d \ge 0 \end{cases} , \tag{3.32}$$

where a is a large value, b small value, and discriminant $d = ((u_x - v_y)^2 + 4u_yv_x)$. H is set so to prevent smoothing within the swirling structure, and allow smoothing elsewhere. Due to stability issues with negative H values, the resulting vector field components are normalized. This backward diffusion gives "edge detection" in a flow field sense, resulting in vector fields with emphasized "critical lines". These lines are connecting the swirl critical point with its matching non-swirling critical point pair (when present). The energy functional is producing normalized vector fields which are not smoothed within the swirling areas and are smoothed outside the swirling areas. Figure 3.17 shows the normalized vector field on which the energy functional (3.31) is minimized. The process introduces additional critical points at the places where "critical lines" are intersecting the swirling area. The swirling features in output vector fields are emphasized and visible, with the uniform magnitude.

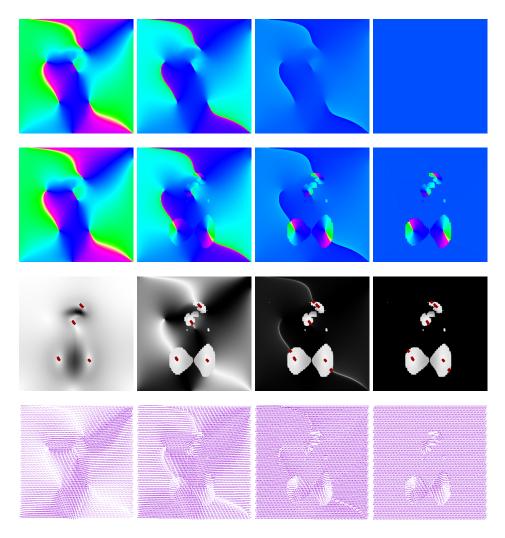


Figure 3.17: Minimization of the energy functional (3.31). Processing results of the test vector field for 1, 2, 4, 10 iterations (left to right). Upper row: nonlinear isotropic diffusion i.e. regularization only (Section 3.2). Vector fields with emphasized critical lines are obtained. Second row: color magnitude of the minimization result of the normalized vector field. Third row: grayscale magnitude and swirling critical points of the resulting fields. New critical points are introduced at the places where critical lines intersect the swirling area. Lower row: arrow plots of the resulting fields. Normalized vector fields with emphasized critical lines and preserved swirling areas are obtained.

Discriminant Steered Energy Functional Requesting Similarity to the Intermediate Vector Field Normalization of the vector field components produces vector fields with emphasized swirling structures, but destroys information about their magnitude. Requesting similarity to the intermediate vector field i.e. to the resulting field from the previous iteration, gives a stable process and still produces vector fields with preserved swirling areas. Non-swirling areas are blurred. Following energy functional is formulated:

$$E(u,v) = \int F(x, y, u, v, u_x, u_y, v_x, v_y) dx dy$$

$$= \int \left(H\left((u - u_{prev})^2 + (v - v_{prev})^2 \right) + \alpha \Psi\left(|\nabla u|^2 + |\nabla v|^2 \right) \right) dx dy,$$
(3.33)

with the same H as in (3.31), and u_{prev} , v_{prev} intermediate vector field components i.e. the result of the previous iteration. Vector field components are not normalized thus preserving the information about the magnitude. Figure 3.18 shows the result of the energy functional minimization. Flow field with preserved, but not strongly emphasized swirling features is produced.

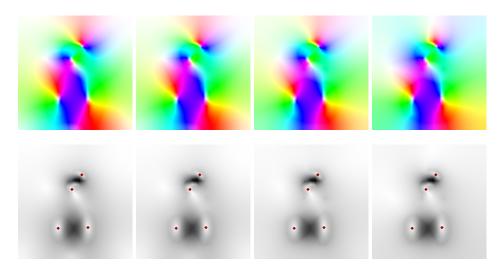


Figure 3.18: Minimization process of the energy functional (3.33) for iteration number: 500, 1000, 5000, 50000 (left to right). Upper row: color coded magnitude of the flow. Lower row: gray scale flow magnitude with vortex cores.

3.4.3 Results

Different approaches were used in order to construct a vortex preserving diffusivity in Section 3.4.1. Figures 3.19, 3.20 show the summary of the designed diffusion methods. Diffusion with diffusivity (3.27) nicely clears the non-swirling areas, but also introduces new swirling points on the boundaries of the swirling areas. Additionally blurring non-swirling area and also small swirling area border produced vector fields with emphasized swirling features without introducing new swirling features. Using diffusivity (3.28) gives an alternative way of producing vector fields with preserved swirling features. Figure 3.20 shows diffusion processed with binary diffusivities based on vortex detection. Areas around the detected swirling critical points are preserved. Lower row of Figure 3.20 shows the Q criterion for vortex region detection used to determine area to be preserved. Diffusion based on vortex detection is e.g. suitable when approaches based on detection of swirling areas preserve patches without vortices (Figure 3.15).

Variational methods for preserving swirling features are presented in Section 3.4.2. Figure 3.21 shows the summary of the designed variational methods. Vector fields produced by the discriminant steered energy functional (3.31) where vector field components are normalized after each step, have emphasized swirling areas and uniform magnitude. Additional critical points are, however, introduced at places where "critical lines" are intersecting the swirling area. Energy functional (3.33) produces vector fields with unchanged swirling areas and blurred surroundings without destroying the magnitude of the flow or introducing new critical points. Resulting flow fields are similar to the diffusion with diffusivity (3.28). It is possible to consider other criteria for defining the function H, such as vortex core or region detection used to determine the preserved areas in Section 3.4.1 "Binary Diffusivity Based on Vortex Detection". Figure 3.22 shows a test vector field processed with the same energy functional (3.31), but using different H functions. Function H based on discriminant d preserves swirling areas regardless of them containing a vortex or not, whereas function H based on vortex core detection preserves only areas around vortices.

Further work includes investigation of the optimal parameters for the diffusion and diffusion-reaction processes. Fast 3D algorithms shall also be researched.

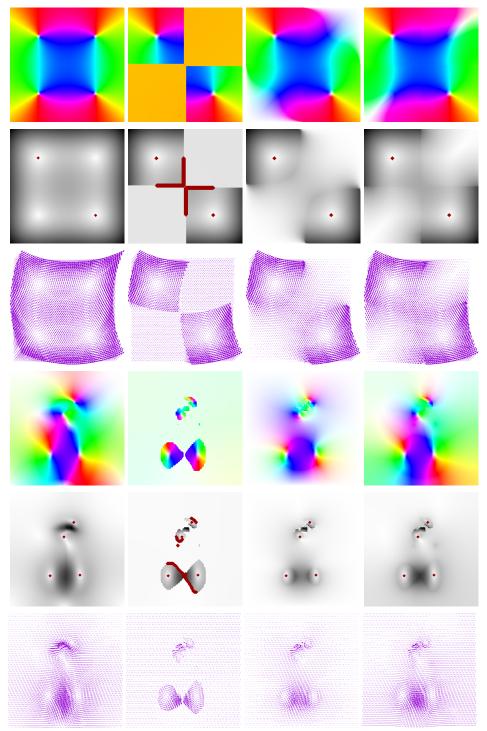


Figure 3.19: Vortex preserving diffusion processes based on discriminant d. Upper, fourth row: color coded magnitude of the vector fields. Second, fifth row: magnitude of the vector fields together with swirling critical points. Third, sixth row: arrow plot of the corresponding vector fields above. First column (left): original test vector fields. Second column: vector fields diffused by nonlinear isotropic diffusion using diffusivity (3.27). Third column: vector fields diffused using diffusivity (3.27) and additionally blurred in the edge areas of swirling structures. Last column (right): vector fields diffused by nonlinear isotropic diffusion using diffusivity (3.28).

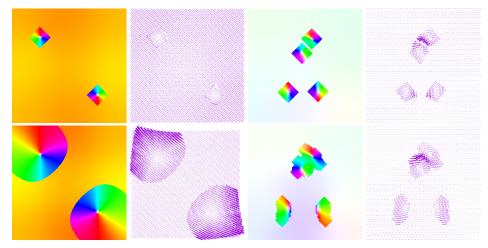


Figure 3.20: Vortex preserving diffusion processes based on vortex core (upper row) or vortex region (lower row) detection. Upper row: vector fields diffused using diffusivity (3.29). Lower row: vector fields diffused using diffusivity (3.30).

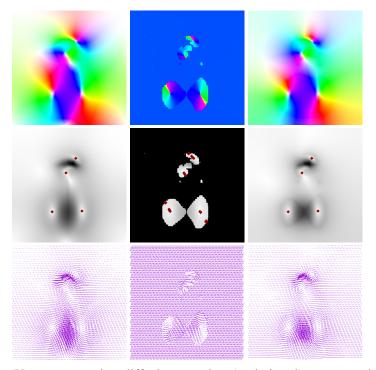


Figure 3.21: Vortex preserving diffusion-reaction (variational) processes based on discriminant d. Upper row: color coded magnitude of the vector field. Second row: magnitude of the vector field together with swirling critical points. Lower row: arrow plot of the corresponding vector field above. First column (left): original test vector field. Second column: vector field processed by discriminant steered energy functional (3.31) with normalized vector field components. Last column (right): vector field processed by discriminant steered energy functional (3.33).

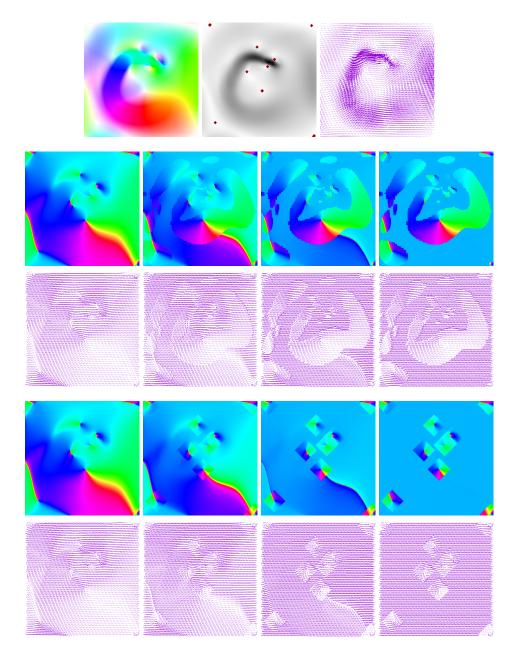


Figure 3.22: Vortex preserving diffusion-reaction (variational) processes based on discriminant d (rows two and three) or vortex core (lower two rows) detection. Minimization of the energy functional (3.31) on a simulated vector field shown in the first row (from Figure 3.14) for 1, 2, 4, 10 iterations (rows two to five, left to right). Second, Third row: color magnitude and arrow plots of the resulting fields from the diffusion-reaction process (3.31) based on discriminant d. Swirling patches without critical points are preserved. Fourth, lower row: color magnitude and arrow plots of the resulting fields from the diffusion-reaction process (3.31) based on vortex core detection. Only areas around the detected vortices are preserved.

Chapter 4

Real-Time Vortex Extraction

4.1 Introduction

This chapter will present vortex detection and extraction methods performing within the real-time simulation. As with the development of modern hardware the computing power is continuously growing, it will eventually be possible to observe or even simulate very high precision time dependent fluid flows in real-time. Feasible fluid simulation is presented in Section 2.2.2. Many tools for vortex extraction from stationary and time-dependent flow data have been developed. They all operate as a post-processing step for locating vortex regions/cores. Vortex detection methods that operate directly in real-time are presented here. Various vortex detection methods are encoded within the real-time simulation from Section 2.2.2. Following vortex region and vortex core detection methods are implemented: vorticity threshold, Q criterion, λ_2 criterion, the eigenvector method via parallel vectors operator (PVO) and the eigenvector method via coplanar vectors operator (CVO).

Diffusion and diffusion-reaction processes that preserve swirling features are presented in Chapter 3. They operate on 2D flow fields (although a straightforward formulation for 3D is possible) and are solved using iterative methods. Here, it is shown how a nonlinear isotropic diffusion presented in Section 3.4.1 can be used as a replacement for the diffusion step of the fluid simulation. Resulting flows have slightly emphasized swirling areas.

Introduction 54

Advantage of presented real-time methods is immediate notion of the shape and location of the vortex structure. Unlike with the iterative diffusion processes, the result is immediately visible, without the time delay. Using this real-time approach on not only simulated, but also real world data is planned as future work. Enhancing the flow simulation by implementing different specially designed diffusion and diffusion-reaction processes within the simulation is also planned.

Section 4.2 gives a short overview of the vortex extraction methods. They can be roughly divided into vortex region and vortex core extraction methods. Common region extraction methods are presented in Section 4.2.1: vorticity threshold, Q criterion and λ_2 criterion. Vortex core extraction methods are presented in Section 4.2.2. Previous work done on minimal bending energy vortex extraction is given in Section 4.2.2.1. The eigenvector method presented in Section 4.2.2.2 is the predecessor method of the parallel vectors operator (PVO) explaining the logic behind its design. Section 4.2.2.3 explains the parallel vectors operator (PVO) method used as a testing method for stationary 3D vector fields in Section 4.3. Feature flow fields (Section 4.2.2.4) are needed for formulation of the coplanar vectors operator (CVO) (Section 4.2.2.5) which can be reduced to a PVO formulation (Section 4.2.2.5).

Section 4.3 shows a GPU implementation of the parallel vectors operator for stationary flow data. OpenCL is used as the programming tool. Vortex cores are successfully detected using the implemented algorithm (Section 4.3.1). By decreasing the number of considered triangles within a cell, the method performance can be speeded up. The higher order PVO method is especially successful with curved vortex cores. Stationary PVO method is used as a check-up tool for the real-time vortex extraction. Real-time simulation can be paused and stationary flow fields obtained. In order to verify results from real-time simulation, vortices are extracted from such stationary fields and comparison is preformed between stationary and real-time extraction methods.

Section 4.4 introduces a vortex core/region extraction within a real-time fluid simulation. The methods are implemented using a GPU with CUDA as a programming environment. The simulation is interactively influenced by the user who is able to input additional force into the flow field in real-time (Section 2.2.2). Real-time simulation is "stirred" by the user using, for example, circular movements with the mouse which are inputing a force into the first slice of the simulation which then spreads through the volume. That way, a helical-like flow field is obtained, suitable for testing purposes. The results of the real-time extraction are color coded within the arrow or within the density plot of the real-time simulation. The vortex structures appear at the expected locations. By intelligent color coding different methods can be compared in real-time.

4.2 Overview of the Vortex Extraction Methods

Vortex is usually defined implicitly through the definition of the extraction method. The methods of detecting and localizing vortices differ in localizing vortex regions or vortex centers or cores. Most common method for determining the vortex region is the λ_2 method ([17]), while the *eigenvector* method ([47]) is the standard for vortex core extraction.

4.2.1 Vortex Region Extraction



Figure 4.1: Vortex region extraction. Horseshoe vortex region ([6]) extracted by the λ_2 method ([17]).

4.2.1.1 Vorticity Threshold

The simplest region based extraction is thresholding. Vorticity $\nabla \times \mathbf{v}$ is used as a thresholding value. A threshold is set on vorticity values. Area of high vorticity is considered to be the vortex region. Another option is putting threshold on helicity $(\nabla \times \mathbf{v}) \cdot \mathbf{v}$ (projection of vorticity onto the flow vector). A potential vortex is a contra-example for these definitions. Alternative approach is using pressure as a thresholding value.

4.2.1.2 Q Criterion (Okubo-Weiss)

In ([14]) Jacobian matrix J is decomposed into symmetric part S (the rate-of-strain tensor that measures the amount of stretching and folding):

$$S = \frac{J + J^T}{2},\tag{4.1}$$

and antisymmetric part A (the vorticity tensor):

$$A = \frac{J - J^T}{2},\tag{4.2}$$

the Q-criterion can be expressed as:

$$Q = \frac{|A|^2 - |S|^2}{2} > 0, (4.3)$$

i.e. the vortex is defined as a spatial region where the Euclidean norm of the vorticity tensor dominates that of the rate of strain ([11]).

4.2.1.3 λ_2 Method (Jeong-Hussain)

In ([17]) Jacobian matrix J is decomposed into symmetric part S and antisymmetric part A. The vortex region is region where at least two of the three eigenvalues of the $S^2 + A^2$ matrix are negative i.e.:

$$\lambda_2(S^2 + A^2) < 0, (4.4)$$

where $\lambda_2(X)$ is the intermediate eigenvalue of a symmetric tensor X (Figure 4.1(a)).

4.2.2 Vortex Core Extraction

4.2.2.1 Minimal Bending Energy Method

Minimal bending energy vortex extraction is based on detecting the streamlines with minimal bending energy which are then considered to be candidates for vortex cores ([7]).

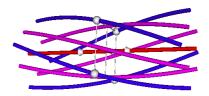


Figure 4.2: The center curve (red) has the minimal bending energy, purple curves higher and blue curves the highest bending energy.

Bending energy stored in a thin beam is proportional to the integral of the square of the curvature $\int k^2 ds$, where k is curvature and s arc length ([5], [13], [26]). A discrete version of the energy is then:

$$B = \sum k^2 d, (4.5)$$

where d is the distance between two consecutive polyline points of the curve approximation, and $|\dot{\mathbf{x}} \vee \ddot{\mathbf{v}}|$

$$k = \frac{|\dot{\mathbf{x}} \times \ddot{\mathbf{x}}|}{|\dot{\mathbf{x}}|^3},\tag{4.6}$$

is curvature where:

$$\dot{\mathbf{x}} = \mathbf{v}(x, y, z) = \begin{pmatrix} u(x, y, z) \\ v(x, y, z) \\ w(x, y, z) \end{pmatrix}, \tag{4.7}$$

$$\ddot{\mathbf{x}} = J_{\mathbf{v}} \dot{\mathbf{x}} = \mathbf{a}(x, y, z) = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix} \begin{pmatrix} u(x, y, z) \\ v(x, y, z) \\ w(x, y, z) \end{pmatrix}. \quad (4.8)$$

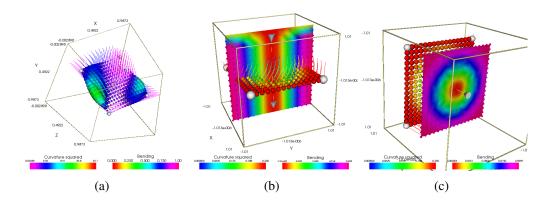


Figure 4.3: Interactive probe is set within a data set. From seed points on the chosen plane, bending energy values are accumulated on a plane. (a) Interactive plane probe can be manipulated within the volume. Curvature values are denoted by size and color of the seed points (spheres), (b), (c) Bending energy can be accumulated at chosen plane (here, in a helical flow field). Values of the bending energy are color coded on the plane (interpolated in-between seed points).

Using the minimal bending energy ensures that the integral curve, with the least energy invested in its bending, is chosen as a candidate vortex core (Figure 4.2).

An interactive plane is placed by the user within the data set volume. Streamlines are integrated from seed points on the plane. The values of the bending energy are accumulated into the seed points (Figure 4.3). The plane is subdivided in order to find the seed points with minimal bending energy. From the point/s with minimal bending energy streamline is integrated as vortex core (Figure 4.4). Optimal length of the vortex core is automatically determined by considering sudden value changes of bending energy along the curve (Figure 4.5).

The consequence of an integration based approach is that the method makes no assumptions about the shape of the vortex and thus has no problems with e.g. extracting bent vortex cores (Figure 4.6). The method produces (if desired) a single vortex core, making the post processing, in order to get rid of the many detected false, short vortex core segments, unnecessary. Utilizing pathlines enables the method to deal with the 2D time-dependent data by considering both time and space on equal terms (Figure 4.7(b)).

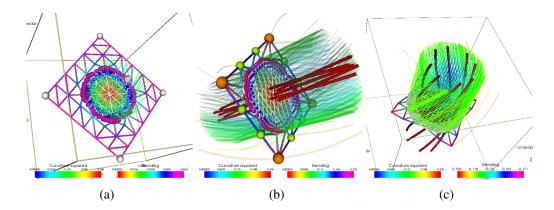


Figure 4.4: Minimal bending energy extraction of the vortex core from a helical flow field. (a) Subdivision is performed on the plane (different subdivision schemes possible), (b) The point/s with the minimal bending energy are chosen as seed point/s for vortex core line candidates to be integrated from these points as integral curves (streamlines/pathlines). (c) Additionally, only the streamlines that can be integrated to full length can be considered

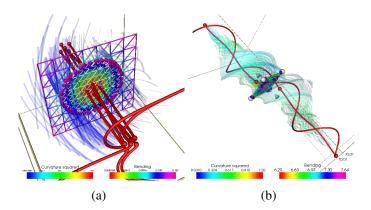


Figure 4.5: Optimal length of the vortex core candidate is determined by considering the bending energy values along the integral curve (e.g. for sudden jumps). (a) helical vortex core breakdown data, (b) dislocated helical vortex core.

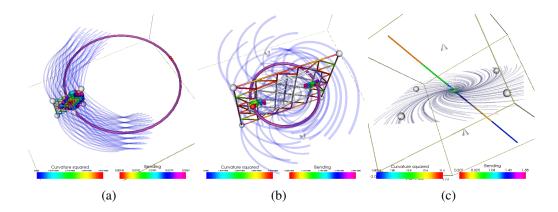


Figure 4.6: (a), (b) extraction of the circular vortex core from a bent helix set, (c) extraction of the vortex core from a vortex rotating in a plane perpendicular neither to its core, nor to the vorticity

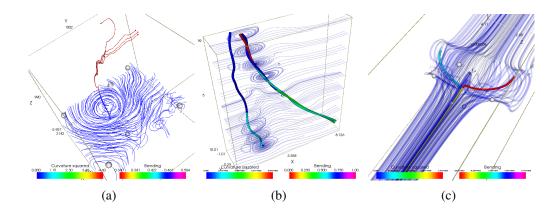


Figure 4.7: (a) Vortex core of the hurricane Isabel data set (b) 2D time dependent data of a simulated cavity flow. Core extraction is performed in a pathline version of the data, while streamlines are depicting the flow (blue ribbons). (c) Vortex cores extracted from a finite elements simulation of a flow behind a cylinder.



Figure 4.8: Vortex cores (yellow) extracted by the eigenvector method ([47]).

4.2.2.2 Sujudi-Haimes Method (Eigenvector Method)

The algorithm of Sujudi and Haimes, also known as the eigenvector method ([47]), is based on using the only real eigenvalue of Jacobian matrix to determine vortex segments. The flow field domain is decomposed into tetrahedra and within each tetrahedron a linear vortex segment is searched for.

The method was designed by generalizing the flow pattern around the focus saddle critical point to all flow points. Namely, [33] showed that if there is a critical point on a vortex core, than that point is a focus saddle. *Globus* [8] used this fact to search for vortex cores by locating focus saddles and extracting cores by integrating streamlines in the direction of the only real eigenvector for that critical point. When a swirling motion is present in 3D, a Jacobian matrix has two conjugate complex eigenvalues and one real eigenvalue. A matching vector is referred to as the real eigenvector.

Using the facts above, Sujudi and Haimes designed reduced velocities which are a projection of the velocity vector (flow vector) to the plane normal to the only eigenvector with the real eigenvalue (Figure 4.9). Requiring for such projection to be zero is equivalent to requiring for the flow vector to be parallel to eigenvector with the real eigenvalue. This fact was used when designing the Parallel Vectors Operator.

The results of the eigenvector method are shown in Figure 4.8. Many short

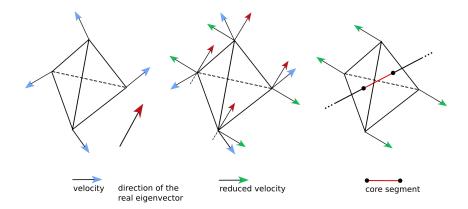


Figure 4.9: Extraction of the linear vortex core segment within a single tetrahedron using the eigenvector method (left to right). Reduced velocities (green arrows) are a projection of the velocities (blue arrows) to a plane perpendicular to the only "real eigenvector" (red arrows). By locating the zeros of the reduced velocity on sides of the tetrahedron, the vortex segment (red line) can be located.

false vortex segments are detected.

4.2.2.3 Parallel Vectors Operator

The Parallel Vectors operator (PVO) by *Peikert and Roth* ([31]) is a tool for extraction of features from vector fields. It generalizes some of the previously designed vortex extraction methods e.g. the eigenvector method. The parallel vectors operator for two n-dimensional vector fields \mathbf{v} and \mathbf{w} denoted as $\mathbf{v} \parallel \mathbf{w}$ is defined as the set:

$$S = \{\mathbf{x} : \mathbf{v}(\mathbf{x}) = 0\} \cup \{\mathbf{x} : \exists \lambda, \mathbf{w}(\mathbf{x}) = \lambda \mathbf{v}(\mathbf{x})\}$$
(4.9)

of all locations where either the two vectors have the same direction or one of them is zero. For 2D or 3D vector fields the set S can be written as:

$$S = \{ \mathbf{x} : \mathbf{v}(\mathbf{x}) \times \mathbf{w}(\mathbf{x}) = 0 \}. \tag{4.10}$$

Some vortex core detection methods can be reformulated using the concept of parallel vector fields.

One of the "standard" vortex extraction method, the eigenvector or Sujudi/Haimes method ([47]) can be obtained by taking $J\mathbf{v}$ as second vector field i.e. by considering the parallelity \mathbf{v} and $J\mathbf{v}$. The method can be formulated as following. If \mathbf{v} considered to be an eigenvector of Jacobian J:

$$J\mathbf{v} = \lambda \mathbf{v},\tag{4.11}$$

then, considering parallel vectors operator, that formulation is equivalent to:

$$\mathbf{v} \parallel J\mathbf{v}. \tag{4.12}$$

Connection between the eigenvector method and the PVO formulation (4.12) is as follows. In the eigenvector method, reduced velocity $\mathbf{r}(\mathbf{x})$ at a point \mathbf{x} is a projection of the steady flow field $\mathbf{v}(\mathbf{x})$ to the plane normal to the real eigenvector $\mathbf{e}(\mathbf{x})$:

$$\mathbf{r}(\mathbf{x}) = \mathbf{v}(\mathbf{x}) - (\mathbf{v}(\mathbf{x}) \cdot \mathbf{e}(\mathbf{x}))\mathbf{e}(\mathbf{x}). \tag{4.13}$$

Places where $\mathbf{r}(\mathbf{x}) = 0$ are vortex cores. PVO is equivalently stating that the locations of the core are at the position where the vector field is parallel to the only real eigenvector of the Jacobian (Figure 4.10): $\mathbf{v}(\mathbf{x}) \parallel \mathbf{e}(\mathbf{x})$.

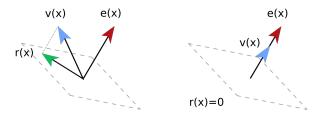


Figure 4.10: Reduced velocity $\mathbf{r}(\mathbf{x})$ (green) is a projection of the velocity $\mathbf{v}(\mathbf{x})$ (blue) to a plane perpendicular to the only "real eigenvector" $\mathbf{e}(\mathbf{x})$ (red). Left: $\mathbf{r}(\mathbf{x}) \neq 0$, Right: $\mathbf{r}(\mathbf{x}) = 0$ i.e. $\mathbf{v}(\mathbf{x}) \parallel \mathbf{e}(\mathbf{x})$.

Further interpretation of the parallel vectors operator indicates a similar logic behind the bending energy method and the PVO. $\mathbf{v} \parallel J\mathbf{v}$ can be rewritten as:

$$\mathbf{v} \parallel (\nabla \mathbf{v}) \mathbf{v}, \tag{4.14}$$

then further rewritten as:

$$\mathbf{v} \parallel \mathbf{a}. \tag{4.15}$$

The acceleration \mathbf{a} of a particle is parallel to its velocity \mathbf{v} , so the streamline through a point on the core has zero curvature at that point. The location of all points where the streamlines have zero curvature can than be defined as the **loci** of zero curvature and expressed in the same manner as the eigenvector method, as equation (4.14).

Depending on the choice of the second vector field, different methods can be implemented simply by testing the parallelity between the input vector field and the second chosen vector field derived from the input field. Encoding torsion $(\nabla \mathbf{a})\mathbf{v}$ gives a **higher order vortex core detection** ([30]) that detects **loci of zero torsion**. The second derivative following a particle in a flow field is $(\nabla \mathbf{a})\mathbf{v}$. The parallel vector formulation of the method is:

$$\mathbf{v} \parallel (\nabla \mathbf{a}) \mathbf{v}. \tag{4.16}$$

4.2.2.4 Feature Flow Fields

Feature flow fields [49] are vector fields modeled in way that a simple integration gives the wanted features. Formulations for tracking critical points and vortices in 2D/3D are given here. They are needed for formulation of the coplanar vectors operator (CVO) in Section 4.2.2.5.

Tracking of Critical Points in 2D Unsteady Vector Fields If $\mathbf{v}(x,y,t) = (u(x,y,t),v(x,y,t))^T = (\mathbf{v}_x,\mathbf{v}_y,\mathbf{v}_t)$ is a 2D unsteady vector field, the feature flow field for tracking the critical points is constructed as:

$$\mathbf{f}(x, y, t) = grad(u) \times grad(v) = \begin{pmatrix} det(\mathbf{v}_y, \mathbf{v}_t) \\ det(\mathbf{v}_t, \mathbf{v}_x) \\ det(\mathbf{v}_x, \mathbf{v}_y) \end{pmatrix}. \tag{4.17}$$

Tracking of Vortex Cores in 3D Unsteady Vector Fields If $\mathbf{v}(x, y, z, t)$ is a 3D unsteady vector field, and $\mathbf{v2}(x, y, z, t)$ the vector field constructed from \mathbf{v}

for applying the PVO, then the feature flow field for extracting the vortex core is constructed as:

$$\mathbf{f}(x, y, z, t) = \begin{pmatrix} +det(\mathbf{w}_y, \mathbf{w}_z, \mathbf{w}_t) \\ -det(\mathbf{w}_z, \mathbf{w}_t, \mathbf{w}_x) \\ +det(\mathbf{w}_t, \mathbf{w}_x, \mathbf{w}_y) \\ -det(\mathbf{w}_x, \mathbf{w}_y, \mathbf{w}_z) \end{pmatrix}, \tag{4.18}$$

where $\mathbf{w} = \mathbf{v} \times \mathbf{v2} = (\mathbf{w}_x, \mathbf{w}_y, \mathbf{w}_z, \mathbf{w}_t)$.

4.2.2.5 Coplanar Vectors Operator

Coplanar vectors operator (CVO) is a generalization of the PVO to the 3D unsteady case ([55]). CVO for 3D unsteady case can be formulated as a 3D PVO problem which makes the implementation straightforward. In order to obtain a formulation for 3D unsteady vector fields, a formulation for 2D unsteady case is first given.

2D Unsteady Path lines of 2D unsteady flow fields $\mathbf{v}(x, y, t)$ are stream lines of the following 3D vector field:

$$\mathbf{p}(x,y,t) = \begin{pmatrix} \mathbf{v}(x,y,t) \\ 1 \end{pmatrix} = \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \\ 1 \end{pmatrix}. \tag{4.19}$$

Jacobian matrix of the vector field **p** is of the form:

$$J(\mathbf{p}) = \begin{pmatrix} u_x & u_y & u_t \\ v_x & v_y & v_t \\ 0 & 0 & 0 \end{pmatrix},\tag{4.20}$$

with eigenvalues $e_1, e_2, 0$ and matching eigenvectors:

$$\begin{pmatrix} \mathbf{e}_1 \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{e}_2 \\ 0 \end{pmatrix}, \mathbf{f}, \tag{4.21}$$

where \mathbf{e}_1 , \mathbf{e}_2 are the eigenvectors of the spatial Jacobian and vector field \mathbf{f} is of the form (4.17). Field \mathbf{f} is designed so that the values of \mathbf{v} do not change along the stream lines of \mathbf{f} i.e. the directional derivative of field \mathbf{v} in direction of field \mathbf{f} is zero. Therefore $J(\mathbf{v}) \cdot \mathbf{f} = 0$ and therefore $J(\mathbf{p}) \cdot \mathbf{f} = 0 \cdot \mathbf{f}$. Field \mathbf{f} is an eigenvector of $J(\mathbf{p})$ with eigenvalue 0.

3D Unsteady Path lines of 3D unsteady flow field $\mathbf{v}(x, y, z, t)$ are stream lines of the following 4D vector field:

$$\mathbf{p}(x,y,z,t) = \begin{pmatrix} \mathbf{v}(x,y,z,t) \\ 1 \end{pmatrix} = \begin{pmatrix} u(x,y,z,t) \\ v(x,y,z,t) \\ w(x,y,z,t) \\ 1 \end{pmatrix}. \tag{4.22}$$

Jacobian matrix of the vector field p is of the form:

$$J(\mathbf{p}) = \begin{pmatrix} u_x & u_y & u_z & u_t \\ v_x & v_y & v_z & v_t \\ w_x & w_y & w_z & w_t \\ 0 & 0 & 0 & 0 \end{pmatrix},$$
 (4.23)

with eigenvalues $e_1, e_2, e_3, 0$ and matching eigenvectors:

$$\begin{pmatrix} \mathbf{e}_1 \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{e}_2 \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{e}_3 \\ 0 \end{pmatrix}, \mathbf{f},$$
 (4.24)

where e_1 , e_2 , e_3 are the eigenvectors of the spatial Jacobian and f is a 4D feature flow field for tracking critical points in 3D unsteady vector fields (4.18):

$$\mathbf{f}(x, y, z, t) = \begin{pmatrix} +det(\mathbf{v}_y, \mathbf{v}_z, \mathbf{v}_t) \\ -det(\mathbf{v}_z, \mathbf{v}_t, \mathbf{v}_x) \\ +det(\mathbf{v}_t, \mathbf{v}_x, \mathbf{v}_y) \\ -det(\mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z) \end{pmatrix}.$$
 (4.25)

The eigenvalue corresponding to field f is zero. Therefore one of the remaining eigenvalues is real and other two are either real or a pair of conjugated complex values. Swirling motion is present only when eigenvalues are conjugated complex. The two real eigenvectors f and e^S span a plane in which no swirling occurs. The cores of swirling motion are at the positions where 4D vectors f, e^S and p are coplanar i.e. the point x lies on a core if the flow vector p(x) lies in a plane of non-swirling flow i.e:

$$\lambda_1 \mathbf{p} + \lambda_2 \mathbf{e}^{\mathbf{S}} + \lambda_3 \mathbf{f} = 0, \quad \sum \lambda_i^2 > 0$$
 (4.26)

i.e, it is necessary to have three linearly dependent 4D vectors.

Formulation Of Coplanar Vectors Operator By Using Parallel Vectors Oper-

ator Coplanar vectors operator for 3D unsteady flow can be formulated as a 3D PVO problem. This makes the implementation straightforward. Expanding (4.26) reads:

$$\lambda_{1} \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix} + \lambda_{2} \begin{pmatrix} \mathbf{e_{1}^{S}} \\ \mathbf{e_{2}^{S}} \\ \mathbf{e_{3}^{S}} \\ 0 \end{pmatrix} + \lambda_{3} \begin{pmatrix} \mathbf{f}_{1} \\ \mathbf{f}_{2} \\ \mathbf{f}_{3} \\ \mathbf{f}_{4} \end{pmatrix} = \mathbf{0}. \tag{4.27}$$

By setting $\lambda_1 = -\lambda_3 \mathbf{f}_4$ the equation reformulates to:

$$\lambda_{2} \begin{pmatrix} \mathbf{e_{1}^{S}} \\ \mathbf{e_{2}^{S}} \\ \mathbf{e_{3}^{S}} \end{pmatrix} + \lambda_{3} \begin{pmatrix} \begin{pmatrix} \mathbf{f}_{1} \\ \mathbf{f}_{2} \\ \mathbf{f}_{3} \end{pmatrix} - \mathbf{f}_{4} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \end{pmatrix} = \mathbf{0}, \tag{4.28}$$

which is a 3D parallel vectors problem, with the two vector fields:

$$\mathbf{v}_{1} = \begin{pmatrix} \mathbf{e}_{1}^{\mathbf{S}} \\ \mathbf{e}_{2}^{\mathbf{S}} \\ \mathbf{e}_{3}^{\mathbf{S}} \end{pmatrix}, \quad \mathbf{v}_{2} = \begin{pmatrix} \begin{pmatrix} \mathbf{f}_{1} \\ \mathbf{f}_{2} \\ \mathbf{f}_{3} \end{pmatrix} - \mathbf{f}_{4} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \end{pmatrix}. \tag{4.29}$$

Applying PVO on 3D unsteady data produces a line sweeping over time i.e. a surface.

4.3 Stationary Flow Parallel Vectors Operator on the GPU

Parallel vectors operator vortex extraction method ([31]) for stationary vector fields is implemented on the GPU using OpenCL. Eigenvector ([47]) method and higher order method ([30]) via PVO are used to extract vortex cores from different analytical and simulated flow fields.

The algorithm proceeds as follows. Data domain, consisting of structured cells, is divided into triangles. Each cell space is covered by a number of strategically placed triangles. By considering different number of space covering elements i.e. triangles, computation can be speeded up. Using only 3 outer sides gives good results and decreases the number of triangles considered per cell. The goal of the algorithm is to find vortex core points within the cells. This is to be achieved by finding out where the two considered vector fields are parallel i.e. by finding zeros of the cross product of the two vector fields on every triangle within the cell. More detail on input data, the algorithm, implementation and covering of the cell with triangles will be given in the text to follow.

Algorithm Detail Input data for the algorithm is of the following form. As input, the algorithm requires a structured set of points and two sets of vectors which are going to be checked for parallelity. Second set of vectors is usually derived from the first set, either by using a separate program or within the real-time simulation (Section 4.4.2.4). Input data is a structured grid of the following form:

xdim	ydim	zdim						
xmin	xmax	ymin	ymax	zmin	zmax			
X	y	Z	u	v	W	u2	v2	w2
:								
X	y	Z	u	v	W	u2	v2	w2

where xdim, ydim, zdim are the dimension of the data in x, y, z direction

(the number of points N is given, spanning N-1 cells), [xmin, xmax], [ymin, ymax], [zmin, zmax] is the data domain, x, y, z point coordinates, u, v, w values of the first vector field and u2, v2, w2 values of the second vector field. The point coordinates x, y, z are given or calculated from data dimensions and data range.

The PVO algorithm for the eigenvector method (Section 4.2.2.3) proceeds as follows. First, the second vector field has to be produced. For every point (x, y, z) of the data, next to the already present vector field $\mathbf{v}(x, y, z) = (u(x, y, z) \ v(x, y, z) \ w(x, y, z))^T$, a new vector field is calculated. Since the method to be implemented is the eigenvector method, the second set of vectors equals $J\mathbf{v} = \mathbf{v}\mathbf{2}(x, y, z) = (u2(x, y, z) \ v2(x, y, z) \ w2(x, y, z))^T$, where J is the Jacobian matrix in a point. Partial derivatives for the Jacobian matrix (with (x, y, z) omitted in the notation):

$$J_{\mathbf{v}} = \nabla \mathbf{v} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix}$$
(4.30)

are approximated from the surrounding grid points (Appendix A.1). After multiplication of the each matrix with the matching vector, obtained vector field is encoded into the data file. For higher order method vortex extraction ([30]) the second vector field is $(\nabla(\mathbf{J}\mathbf{v}))\mathbf{v} = (\nabla\mathbf{a})\mathbf{v}$.

After the data file is complete, the vortex core calculation can commence. Zeros of the cross product of the two vector fields are determined on each available face within the data. Starting in the center of each triangle, a number of 2D Newton-Raphson steps are performed. Vector $\mathbf{c} = \mathbf{v} \times \mathbf{v2}$ is minimized on every triangle. Partial derivatives of the cross product with respect to 2 axes ξ_1, ξ_2 (the 2D triangle coordinate system) are calculated:

$$\frac{\partial \mathbf{c}}{\partial \xi_i} = \frac{\partial \mathbf{v}}{\partial \xi_i} \times \mathbf{v2} + \mathbf{v} \times \frac{\partial \mathbf{v2}}{\partial \xi_i},\tag{4.31}$$

thus obtaining the 3×2 matrix $\mathbf{J_c}$. In order to obtain the "correction" vector $\Delta \xi_i$ the system

$$\mathbf{J_c} \Delta \xi_i = -\mathbf{c} \tag{4.32}$$

needs to be solved. Since this is an overdetermined system least square solution is used and system:

$$\Delta \xi_i = \left(\mathbf{J_c}^T \mathbf{J_c} \right)^{-1} \left(-\mathbf{J_c}^T \mathbf{c} \right) \tag{4.33}$$

is solved to obtain the correction vector $\Delta \xi_i$ on the triangle face. After a certain number of iterations on the triangle the zero point (if found) is stored as a part of the vortex core. Optionally, the zeros within the tetrahedron can be connected in order to form a vortex core segment.

The parallel vectors operator is implemented on the GPU using OpenCL/C++ ([28]). The key feature of the problems being implemented on GPUs are the parallelization possibilities, i.e. the ability to divide the problem into many independent problems, all requiring same calculation. The loaded data is padded with zeros to achieve coalescent reads from memory.

Choosing a Number of Tetrahedrons Cells within the data are divided into tetrahedra i.e. into triangles. On each triangle an iterative algorithm attempts to find a zero of a PVO operator i.e. a point belonging to a vortex core. The cell/box can be uniformly divided into 5, 6 or 12 tetrahedra (Figure 4.11). Better coverage



Figure 4.11: A cell can be uniformly divided into 5, 6, or 12 tetrahedrons.

of space gives more precise results. Using less tetrahedrons i.e. triangles speeds up the calculation. Since the implementation does not connect the zeros found on the faces of a tetrahedron, single points are obtained as output. This allows basing the calculation on triangles as calculation units instead on tetrahedrons.

Versions of the algorithm using different parts of cell structure were tested (Table 4.1). The table shows the number of triangles which have to be processed by the algorithm per data cell. The number of triangles depends on the way the cell space was covered. The tests were made by using outer sides of the cell only or by using the division of the cell to 5 and 6 tetrahedra. Using 3 outer sides plus inner sides, or even using only 3 outer sides gives satisfactory results.

No.	Structures used within the cell	number of triangles		
		in the used structure		
1.	3 outer sides of the cell	6		
2.	6 outer sides of the cell	12		
3.	5 tetrahedra (all sides)	20		
4.	5 tetrahedra (unique sides only)	16		
5.	inner sides (cell divided to 5 tetrahedra)	4		
6.	3 outer sides + 4 inner triangles (5 tetrahedra)	10		
7.	6 tetrahedra (all sides)	24		
8.	6 tetrahedra (unique sides only)	18		
9.	inner sides (cell divided to 6 tetrahedra)	6		
10.	3 outer sides + 6 inner triangles (6 tetrahedra)	12		

Table 4.1: Number of triangles to be processed by the algorithm depends on the choice of the underlaying structure of the cell used for calculation. Table shows different configurations which were tested. Configuration 1 and 2 use only the outer sides of the cells and are independent of the division of the cell to tetrahedra. Configurations 3-6 use the division of the cell to 5 tetrahedra, and configurations 7-10 division to 6 tetrahedra.

Tables 4.2 and 4.3 show the resulting images of the vortex extraction on the helical and bent helical flow field using different space coverage setup. Figure 4.12 shows the close-up view of the vortex core extracted from the helical field. Configurations 8, 10 and 1 are shown in the images, processing consecutively 18, 12 and 6 triangles. Using only 3 outer neighboring sides of the cell yielded good results with analytical examples. This also ensures that no double calculation is performed between neighboring cells, since the neighbor cells will perform the calculation on the remaining sides.

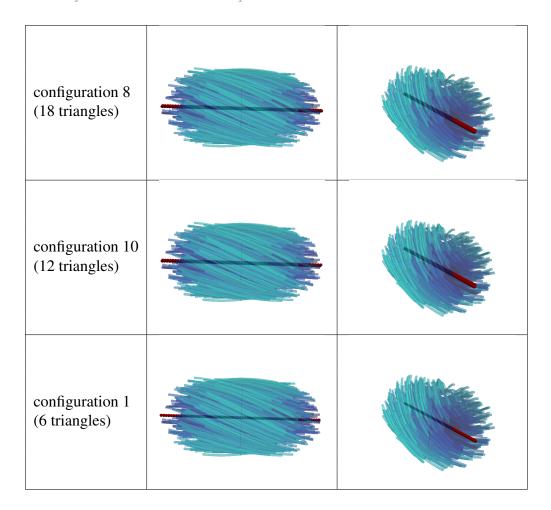


Table 4.2: Vortex core extraction from helical flow field using the eigenvector method via PVO. The algorithm is processing different numbers of triangles per cells. The left table column shows the used configuration from Table 4.1. Middle column shows the side view and right column the frontal view of the helical flow field with vortex core denoted by red spheres. All configurations give the expected vortex core as output. Slight differences in the extracted vortex core are shown in Figure 4.12.



Figure 4.12: A close up view of the vortex cores extracted from the helical flow field by using a PVO algorithm with different cell coverage (Table 4.1). Left: extraction by using configuration 8 gives triple row of spheres in the core. Middle: extraction by using configuration 10 gives a double row of spheres in the core. Right: extraction by using configuration 1 gives a single row of spheres in the core.

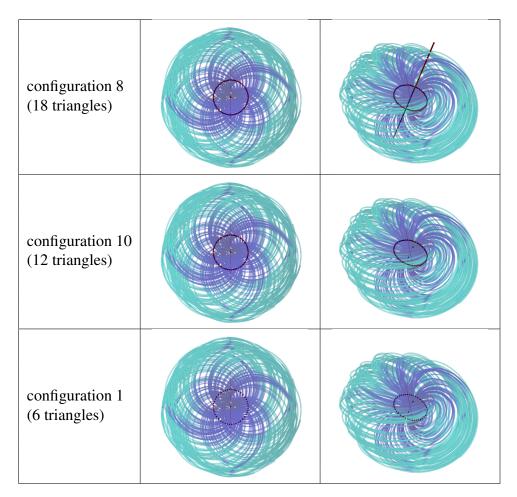


Table 4.3: Vortex core extraction from bent helix flow using the eigenvector method via PVO. The algorithm is processing different numbers of triangles per cells. The left table column shows the used configuration from Table 4.1. Middle column shows the top view and right column the side view of the bent helical flow field with vortex core denoted by red spheres. All configurations give the expected vortex core as output. Configurations with more triangles give a denser vortex core, while configurations with low triangle number used per cell give a sparser core. Note that the algorithm with configuration 1 still gives a good vortex extraction result although only 3 outer triangles are used per cell. A close-up of differences in the extracted vortex core are shown in Figure 4.13.

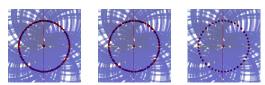


Figure 4.13: A close up view of the vortex cores extracted from the bent helical flow field by using a PVO algorithm with different cell coverage (Table 4.1). Right: extraction by using configuration 1 gives a sparser core, but it is still a good approximation.

4.3.1 Results

Figure 4.14 shows the results of the eigenvector method via PVO for different analytical flow fields. Obtained results, where vortex core points are denoted by red spheres, are at the expected locations and are noiseless. Visualizations were made using the ParaView scientific visualization application ([29]).

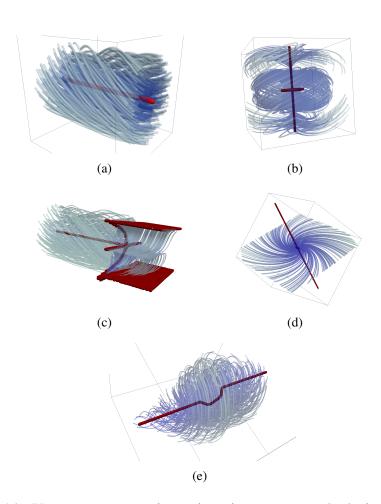


Figure 4.14: Vortex core extraction using eigenvector method via PVO implemented in OpenCL. (a) helical flow, (b) bent helix, (c) helical vortex breakdown, (d) vortex rotating in a plane perpendicular neither to its core, not to the vorticity, (e), (f) dislocated helical vortex.

Figure 4.15 shows the vortex extraction results for simulated flow fields. The central core structure is nicely visible (especially when rotating the result in 3D is possible). Results show noise typical for eigenvector method. Figure 4.16 shows the results of the higher order method via PVO. Curved vortex cores are more correctly detected using the higher order method.

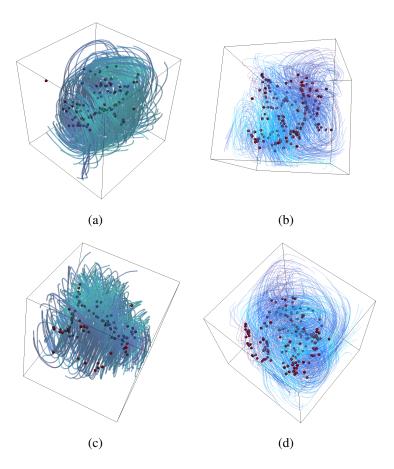


Figure 4.15: Vortex cores extracted from simulated flow data using eigenvector method via PVO. Left column and right column represent two data sets from different views. The data sets are two paused real-time simulation instances. When observing the data in 3D central structures are nicely visible (denoted red). First data set is a fairly straight core (green streamlines), second is curved and branches into several structures (blue streamlines). Presence of noise is characteristic for eigenvector method.

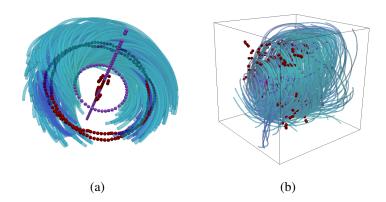


Figure 4.16: Vortex cores extracted using the higher order method ([30]) i.e. instead of locating zero curvature, zero torsion is located. (a) Bent vortex example. Purple points are extracted by the eigenvector method. Red points by the higher order method. Here it is visible how eigenvector method has problems with curved vortices which are overcomed by using higher derivatives. (b) Higher order method on a simulated data (center points are roughly denoted in magenta).

4.4 Real-Time Flow Vortex Extraction on the GPU

Vortex extraction methods are integrated into the real-time fluid simulation (Section 2.2.2, [46]). Methods implemented are the eigenvector method via PVO, eigenvector method via CVO, and also vortex region methods: threshold on vorticity, Q criterion, λ_2 criterion. Figure 4.17 shows the real-time implementation. Quasi-volume rendering and arrow plots (without the arrow head) are used for visualization of the result. The result is color coded within the simulation, allowing instant visualization of vortex structures and comparison of different methods. Area where the measured quantities are zero are color coded as black or white, depending on the background color. The best visibility is achieved when the simulation volume is directly rotated in 3D.

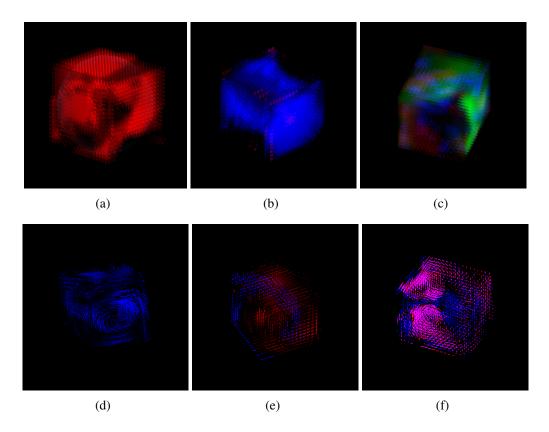


Figure 4.17: Real-time simulation is used to directly indicate a vortex region/core within the data volume while running a real-time simulation. Both arrow plots (without the arrow head) and quasi-volume rendering will be used, as alternating different techniques offers additional visibility. Upper row shows volume rendering with different quantities color coded within the simulation. Comparing two quantities is then easy, since a mixture of two colors can easily be interpreted. For example, image (b) shows a region of a swirling motion (discriminant less than zero) encoded blue, and a vortex core encoded red. Intersection of the two quantities appears purple (the center structure in the image is purple, as are some artifact border areas). Comparing more than two quantities is possible, but visual interpretation is then difficult (c). Lower row shows arrow plots of flow fields with different quantities encoded as line (arrow) color (using either continuous or thresholded color scaling).

4.4.1 Simulation Enhancement Through Nonlinear Isotropic Diffusion

Flow simulation is implemented using an approach presented in Section 2.2.2 from [46]. Details are presented in subsection 2.2.2.2 and present four steps which are continuously iterated in order to obtain the simulation: force input, diffusion, advection and projection. Diffusion serves as a smoothing and also as a transfer mechanism. In order to obtain a stable simulation, instead of doing a forward diffusion, the values for which backward diffusion gives the initial state are searched for (Appendix A.4).

Linear diffusion step is replaced with the nonlinear isotropic diffusion presented in Section 3.4.1 where binary diffusivity (3.27) is used in order to emphasize the swirling areas. The diffusion from Section 3.4.1 is an iterative process operating on 2D vector fields. Here, the same diffusion is applied to 3D time-dependent fields. In order to maintain the real-time execution, only one iteration is preformed per simulation step. Backward nonlinear isotropic diffusion is used to keep the simulation stable (Appendix A.6). Figure 4.18 shows the resulting flow simulation which exhibits slightly emphasized swirling structures when compared to the regular simulation.

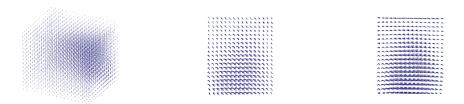


Figure 4.18: Left, Middle: side and top view of the arrow plot of the fluid simulation using linear diffusion. Right: top view of the simulation using nonlinear isotropic diffusion with diffusivity (3.27) shows a slightly emphasized swirling structure.

The standard simulation (with linear diffusion) shall be used throughout the rest of this chapter, since the vortex extraction methods, unlike the approach presented here, do not aim to change the flow simulation itself.

4.4.2 Implementation of Vortex Detection Methods

4.4.2.1 Vorticity Threshold

Magnitude of the vorticity vector:

$$\nabla \times \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix} \tag{4.34}$$

is used to color code the vortex region within a real-time simulation (Figure 4.20).

4.4.2.2 Q Criterion

Q criterion ([14]):

$$Q = \frac{|A|^2 - |S|^2}{2} > 0, (4.35)$$

where $A = \frac{J-J^T}{2}$, $S = \frac{J+J^T}{2}$, J Jacobian matrix, is used as a indicator of a vortex region (Figure 4.21). Euclidean norm of a matrix is the largest singular value of the matrix.

4.4.2.3 λ_2 Criterion

 λ_2 Criterion ([17]), i.e. the region where at least two eigenvalues of (S^2+A^2) are negative:

$$\lambda_2(S^2 + A^2) < 0 (4.36)$$

is used as a indicator of a vortex region (Figure 4.22). Q criterion and λ_2 criterion are compared in Figure 4.23.

4.4.2.4 Parallel Vectors Operator

Parallel vectors operator for stationary 3D fields is used to indicate vortex cores/regions. PVO for 3D stationary vector fields can be expressed as:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \times \begin{pmatrix} u_x & u_y & u_t \\ v_x & v_y & v_t \\ w_x & w_y & w_t \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tag{4.37}$$

left hand side giving the vector:

$$\begin{pmatrix} (w_{x}u + w_{y}v + w_{z}w)v - (v_{x}u + v_{y}v + v_{z}w)w \\ (u_{x}u + u_{y}v + u_{z}w)w - (w_{x}u + w_{y}v + w_{z}w)u \\ (v_{x}u + v_{y}v + v_{z}w)u - (u_{x}u + u_{y}v + u_{z}w)v \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$
(4.38)

The elements of the obtained 3D vectors are summed up within the simulation implementation. If the sum a + b + c is very small (less than a small threshold), the area is encoded with color (Figures 4.24, 4.25). PVO and Q criterion are compared in Figure 4.26, PVO and λ_2 criterion in Figure 4.27.

4.4.2.5 Coplanar Vectors Operator

Coplanar vectors operator (Section 4.2.2.5) is used for vortex extraction from 3D unsteady flow fields. As seen in Section 4.2.2.5, it can be formulated via parallel vectors operator. This, in practice, means that the two vector fields to be checked for parallellity need to be defined. From Section 4.2.2.5 these two fields are:

$$\mathbf{v_1}(\mathbf{x}) = \mathbf{e}(\mathbf{x}) = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}, \qquad \mathbf{v_2}(\mathbf{x}) = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} - f_4 \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \qquad (4.39)$$

where $e(\mathbf{x})$ is the only "real eigenvector" of the Jacobian matrix, $(u, v, w)^T$ is the original vector field and $(f_1, f_2, f_3, f_4)^T$ feature flow field (Section 4.2.2.4). The cross product of the two vector fields is calculated:

$$\begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} \times \begin{pmatrix} f_1 - f_4 u \\ f_2 - f_4 v \\ f_3 - f_4 w \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}. \tag{4.40}$$

The obtained components of 3D vectors are again summed up. The area where the sum a + b + c is small is color coded and represents a vortex region (Figure 4.28). CVO and PVO are compared in Figure 4.29.

4.4.3 Results

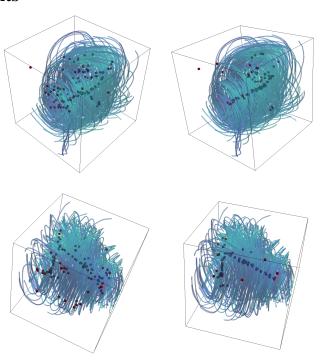


Figure 4.19: Images show a comparison between the PVO extraction from a flow data produced within the real-time simulation and outside of it. PVO eigenvector method vortex extraction from a stationary 3D vector field is preformed. The vector field is saved from a paused real-time simulation. Upper and lower row show two views of the same vector field. The extracted vortex points are denoted as spheres. The data for the extraction was prepared in two ways (left and right column). Before starting the extraction the values of the second vector field $J \cdot \mathbf{v}$ have to be calculated: Left column: values of the second vector field encoded by a separate script, Right column: values of the second vector field encoded directly in the real-time simulation. Results are noisy as expected when using the eigenvector method. Both data result in detecting the same main center structure.

Figures showing the result of vortex extraction methods will be shown here. First, a comparison between a real-time and stationary extraction will be made (Figure 4.19). The shown 3D vector field is saved from a paused real-time simulation. The fluid was circularly stirred by the user to produce a helical like flow field. Two views of the same vector field are shown (upper and lower row) in order to get a better sense of the result. Results of a PVO eigenvector method extraction from the vector field are shown. The needed flow data was prepared in two ways. For applying the PVO, two vector fields are needed, the second one being

calculated from the original vector field. Left column is obtained by encoding the values of the second vector field (Jacobian matrix times the original vector field $J \cdot \mathbf{v}$) in a separate VTK ([48]) program. The values of the second vector field in the right column are obtained directly from the real-time simulation. Extractions yield noisy result (points denoted as spheres), but the same center structure is detected, confirming that the real-time simulation is dealing with the correct numbers.

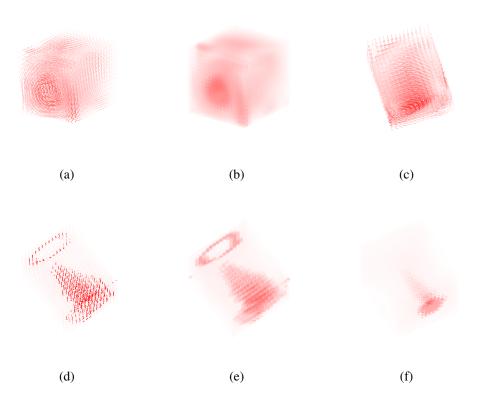


Figure 4.20: Real-time simulation with higher values of vorticity encoded red. (a), (b), (c) show an arrow and volume plots with the continuous color coding. (d), (e), (f) show an arrow and volume plots with the thresholded color coding. Values higher than a threshold have a full color value. Highlighted areas coincide with the expected position of the vortex.

Figure 4.20 shows the values of vorticity encoded into the simulation. Arrow plots and volume plots are shown. Higher values of vorticity are encoded red, either as continuous values (upper row) or a threshold is set that maps all values above it into red color (lower row). The indicated area coincides with the

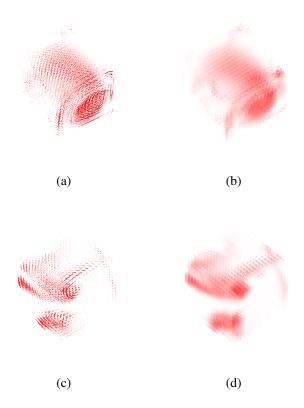


Figure 4.21: Real-time simulation with Q criterion encoded red. (a), (b) show the Q criterion of a helical like flow with nicely visible center vortex region. (c), (d) show the Q criterion of flow produced by diagonal movement across the first slice. Vortices rolling on the sides of the volume can be nicely distinguished from the rest of the flow.

expected position of the vortex.

Figure 4.21 shows the Q criterion of the simulation encoded red. The helical like flow was again tested (upper row), together with the flow obtained by diagonal movements across the first slice (lower row). Such flow has, as nicely seen in the figure, vortices rolling along the sides of the volume.

Figure 4.22 shows the λ_2 criterion of the simulation encoded blue. The helical like flow (upper and middle row) and the diagonal movements flow (lower row) are tested. Vortices are indicated at expected positions. Vortices rolling along the sides of the volume in the "diagonal" flow are clearly visible.

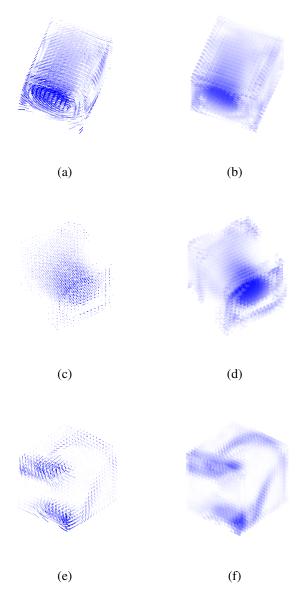


Figure 4.22: Real-time simulation with λ_2 criterion encoded blue. (a), (b) show the λ_2 criterion of a strong helical like flow, shortly after the force input. (c), (d) show the λ_2 criterion of a already dissipated helical like flow i.e. after some time has passed. As with the Q criterion detection, the values of the criterion are boosted by a threshold that has to be adjusted depending on whether one wants to clearly see the strong or weak structures. (e), (f) show the λ_2 criterion of diagonal-movement-initiated flow. Vortices rolling on the sides of the volume are nicely visible.

Figure 4.23 shows the comparison of the Q and λ_2 criterion of the simulation. λ_2 criterion is color coded as cyan, Q criterion as magenta. The area where both indicate a vortex region appears as dark blue/purple. Vortex regions are overlapping nicely.

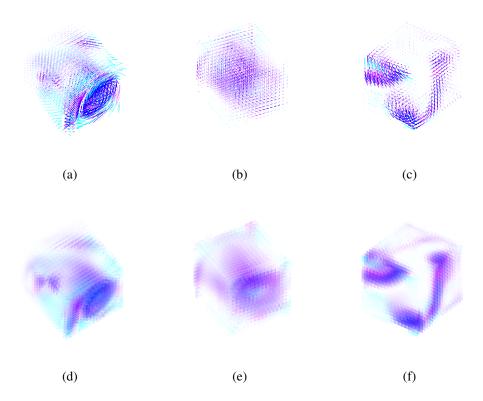


Figure 4.23: Real-time simulation comparing the λ_2 and Q criterion. λ_2 criterion is color coded as cyan, Q criterion as magenta, and the area where both indicate a vortex region appears as dark blue/purple. (a), (d) shows the comparison in a strong helical like flow, shortly after the force input. (b), (e) shows the comparison of a weak helical like flow i.e. after some time has passed. (c), (f) shows the comparison of the flow (initiated with diagonal movement across the first slice) with vortices rolling on the sides of the volume. Vortex regions overlap nicely when the boost threshold is set carefully.

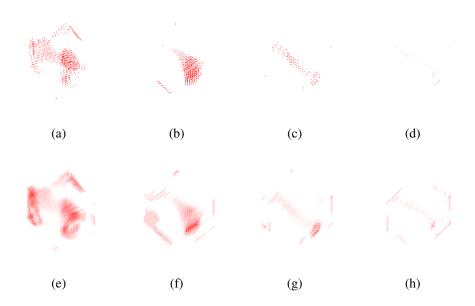


Figure 4.24: Real-time simulation where values of the parallel vectors operator are encoded red. Images show development of the vortex structure in time in a helical like flow field. (a),(e) show the initial screw like vortex structure, (b),(f) show the clear vortex center later in time, (c),(g) show the bounce back vortex core forming in the middle of the volume, (d),(h) show the final stabilization of the core. Highlighted areas coincide with the expected position of the vortex core.



Figure 4.25: Real-time simulation where values of the parallel vectors operator are encoded red. Images show the vortex structures in a flow field obtained by diagonally inputing force into the first slide of the volume.

Figures 4.24, 4.25 show the real-time simulation with eigenvector method via parallel vectors operator encoded red. Since locations of zero vectors are searched (where cross product is zero), a scalar value representing the magnitude of the resulting vectors is used. Places where this scalar is small are color coded red. Highlighted areas coincide with the expected position of the vortex cores. The results were also verified by pausing the simulation and using the algorithm

for stationary flows (Figure 4.19).

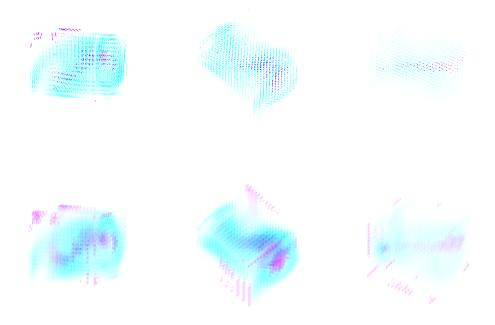


Figure 4.26: Comparison of the PVO (magenta) and the Q criterion (cyan). Core extraction via PVO and the vortex region detection via Q criterion coincide nicely. First column shows the strong helical flow, middle column intermediate flow, and the right column shows the already weak flow.

Figure 4.26 shows the comparison of the Q criterion with the eigenvector method via PVO. Figure 4.27 the comparison of the λ_2 criterion with the eigenvector method via PVO. PVO is color coded as magenta, Q or λ_2 criterion as cyan. The area where both indicate a vortex region appears as dark blue/purple. Vortex core detection using PVO and the vortex region detection using Q or λ_2 criterion both detect expected regions.

Figure 4.28 shows the real-time simulation where vortex values obtained by coplanar vectors operator are color coded blue. Coplanar vectors operator is implemented via parallel vectors operator. Two vector fields are again checked for parallelity. When swirling motion is present, Jacobian matrix has two conjugated complex eigenvalues and one real eigenvalue. First input vector field is the vector field of real eigenvectors. The second input vector field is derived from the feature

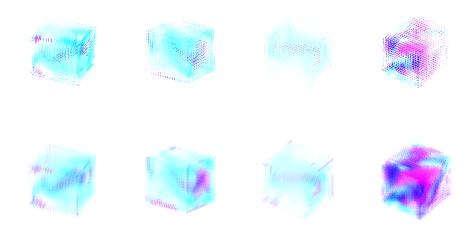


Figure 4.27: Comparison of the PVO (magenta) and the λ_2 criterion (cyan). Core extraction via PVO and the vortex region detection via λ_2 criterion coincide nicely. Right pair of images shows the comparison in a "diagonal force input" flow set.

flow field and the original vector field (see 4.4.2.5). Expected vortex structures are indicated at wanted locations.

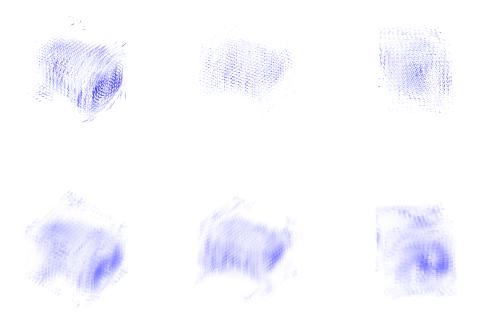


Figure 4.28: Real-time simulation with values of the coplanar vectors operator encoded blue. Coplanar operator indicates expected vortex structures.

Figure 4.29 shows the comparison of the PVO and CVO. Values of the CVO are encoded cyan, values of the PVO magenta, values where both methods detect vortex areas are dark blue/purple.

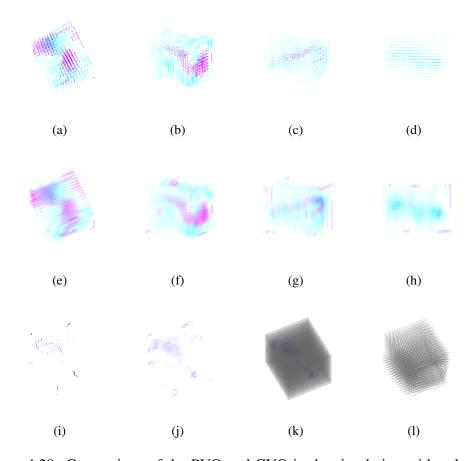


Figure 4.29: Comparison of the PVO and CVO in the simulation with values of the coplanar vectors operator encoded cyan and the values of the parallel vectors operator magenta. Dark blue/purple values indicate locations where both methods detect vortex areas. Figures (a), (b), (e), (f) show the comparison in a strong helical like flow, shortly after the force input. At this point, while the PVO indicates an initial screw-shaped region, the CVO indicates a straight vortex region in a center of the helical-like field which can be observed in dark blue/purple areas. After a short time vortex structures coincide ((c), (d), (g), (h)). Lower row shows places in a strong flow where both PVO and CVO indicate a vortex (color coded as blue).

Chapter 5

Gesture Classification by Detecting Vortices in Ensemble Flow

5.1 Introduction

This chapter presents an addition to gesture classification methods through vortex core extraction of an ensemble range flow. An approach to improve the range flow data obtained by the Kinect device and introduction of ensemble range flow are also presented here. An ensemble range flow of an image set is an accumulated 3D flow in which the structure tensor is averaged throughout the entire data set. It gives an idea of the overall movement in the image set. Detecting vortices in such accumulative flows can give an additional insight into the gesture data.

Range or scene flow is an extension of optic flow to three dimensions. As an input for the range flow estimation algorithm, standard color image pair is required, but also the depth image pair. Depth images are encoding the distance of objects in an observed scene using grayscale values. Depth images are obtained by utilizing Microsoft Kinect and contain many artifacts. Section 5.2 gives an introduction theory behind the optic and range flows. Section 5.3 shows how a range flow algorithm can be improved on. Magnitude of the derivatives of the input depth images is thresholded and the energy functional is expressed as a combined local-global approach.

Introduction 92

Section 5.4 introduces an ensemble range flow. Ensemble range flow is obtained by averaging the structure tensor throughout the entire data set consisting of number of images. The resulting flow field gives an overall motion through the data set.

Detecting vortices within the ensemble range flow of gesture data is presented in Section 5.5. Using the Microsoft Kinect camera, five people recorded nine gestures each. Every gesture data contains 60 frames of color and depth images. Ensemble range flow is calculated from each of the gesture data. Gesture area is determined and vortex cores extracted within this area. The position and the number of obtained critical points can serve as an additional indicator for classifying the gestures.

This approach is not meant to serve as a standalone gesture detection technique, but rather intended as an improvement of gesture classification methods that operate on data of reduced dimensions. An approach that reduces the dimension of the data is e.g. presented in [18]. They segment the static hand gesture images and project the segmentations to x and y axis, thus reducing the data dimension from 2D to 1D. Such 1D data is then classified. An idea for similar technique is presented in Section 5.5.1. Here, a magnitude of the flow was projected to x and y axis, so creating a 1D plot which can be used for classification.

For the purpose of gesture classification only the number of obtained vortices is considered. If processed flow field with emphasized vortices is wanted, diffusion techniques from Section 3.4.1 can be used to create it (Section 5.5.2).

A lot of research is currently being done in utilizing modern depth sensors. Particle based approach for estimating range flow from Kinect data is presented in [10]. Invalid depth areas were not treated separately and are causing incorrect flow estimation. Real-time range flow estimation can be achieved by using GPU implementation [34]. In [20] a 3D mesh is also used to estimate the range flow. Papers like [16] show state of the art usage of the Kinect sensor for real-time 3D reconstruction of the environment which can also be applied to gesture recognition. Small Leap Motion device ([19]) instantly detects gestures preformed within its reach. Alternative sensors, such as accelerometers (sensors capable of de-

tecting linear accelerations), gyroscopes (measuring angular rates around one or more axes) or geomagnetic sensors (measuring Earth's magnetic field along multiple axes) shall undoubtedly be used in future to capture complete movements in a three-dimensional space and much more. Vortex extraction for gesture classification presented in this chapter is not intended to compete with such approaches, but rather to show the possibility of introducing vortex extraction into gesture classification.

Computing range flow between two frames of the gesture data produces a 3D flow in a plane (2.5D). Computing flow between each of the two images in a gesture and stashing them next to each other gives a volume of 3D flows. Ensemble flow is mapping this "volume" into a single 3D flow. Techniques that operate within a volume of 3D flows would have an advantage of not having to use any accumulative process (like ensemble flow) and not having to determine the start and the end of a gesture. Fluid simulation in Chapter 4 is a 3D flow volume. Future work includes immersing the gesture detection into the real-time fluid simulation and taking advantage of the real-time vortex detection techniques.

5.2 Range and Optic Flow

Optic Flow In order to retrieve a vector field $(u(x, y, t), v(x, y, t))^T$ describing a motion in a 2D image sequence, following energy functional has to be minimized:

$$E(u,v) = \int \left(\frac{1}{2} \left((f_x u + f_y v + f_t)^2 + \alpha \mathcal{V}(\nabla u, \nabla v) \right) \right) dx dy, \tag{5.1}$$

where f(x, y, t) is the input image sequence, f_x, f_y, f_t corresponding partial derivatives and \mathcal{V} the chosen penalizer. The data term is the color constancy over time assumption, i.e. the optic flow constraint:

$$f_x u + f_y v + f_t = 0. (5.2)$$

Figure 5.1 shows the input image pair and color coded output images which represent the magnitude of the obtained vector field (color coding depicted in upper row). Mathematical formulation leading to a discrete explicit scheme is given in Appendix A.10.

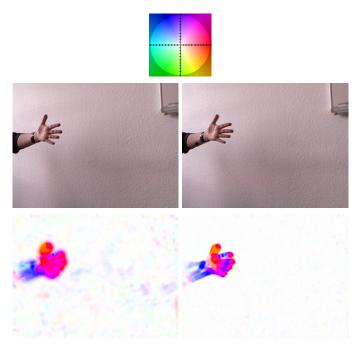


Figure 5.1: Upper row: Color coding for the uv optic flow. Middle row: two consecutive image frames of a scene that shows a human hand moving right, up and back. Lower row: color coded magnitude of the optic flow. Depending on the chosen regularization results differ slightly. Left: Charbonnier regularization, Right: anisotropic regularization.

Range Flow Range or scene flow is an extension of optic flow and describes the 3D motion in a scene ([50], [44]). Estimation of range flow requires, in addition to the standard pair of images, also information about depth (Figure 5.2). Range flow is important for various applications i.e. segmentation, gesture recognition and similar ([43]).

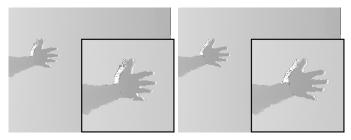


Figure 5.2: Depth channels (obtained by Kinect) are aligned to color channels (Figure 5.1(b), 5.1(c)) using backward warping. Invalid areas (white) are visible in depth images.

The aim is to estimate the range flow:

$$\mathbf{h}(x,y,t) = \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \\ w(x,y,t) \end{pmatrix}, \tag{5.3}$$

which can be done by minimizing the following energy functional:

$$E(u, v, w) = \int F(x, y, u, v, w, u_x, u_y, v_x, v_y, w_x, w_y) dxdy$$

$$= \int \left(\frac{1}{2} \left((f_x u + f_y v + f_t)^2 + \beta (g_x u + g_y v + w + g_t)^2 \right) + \alpha \mathcal{V}(\nabla u, \nabla v, \nabla w) \right) dxdy,$$
(5.4)

where f(x,y,t), g(x,y,t) are given color and depth image sequences, $f_x, f_y, f_t, g_x, g_y, g_t$ are corresponding partial derivatives and \mathcal{V} is penalizer/regularizer functional. Mathematical formulation leading to a discrete explicit scheme is given in Appendix A.11.

5.3 Correction of Range Flow Computation - Combined Local-Global Range Flow

Data recorded with Kinect, consisting of color and depth channels, requires alignment of the two channels (a.k.a. calibration) ([9]) prior to estimating the flow ([50], [44]). The recorded data exhibits invalid areas and unstable edges (Figure 5.2). As mentioned in Section 2.3.1, invalid areas are a consequence of occlusions which occur because of the shift between the source of active illumination and the infrared camera, caused by the *structured light depth estimation* approach [1], [51].

The output of the algorithm is the assessed 3D flow. Figure 5.3(a) shows the magnitude of the optic uv flow, Figure 5.3(b) shows the magnitude of the third depth w flow component. Color coding for the depth component of the range flow is interpolated between blue for backward and orange for forward movement.

Existence of invalid areas poses a problem for accurate range flow estimation. Figure 5.3 shows the result of the range flow algorithm with the invalid areas included. Visible are the noisy hand edge parts originating from the invalid areas.

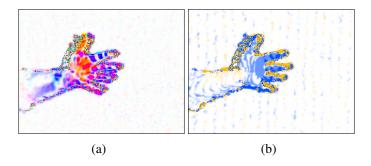


Figure 5.3: (a) Magnitude of the optic uv flow, with the invalid areas included (b) Magnitude of the w component of the range flow, with the invalid areas included i.e. with the calibrated unprocessed input images.

By excluding such invalid areas (Figure 5.4) from the range flow estimation, the results of the estimation are improved (Figure 5.5). The invalid areas are excluded by thresholding of the derivative magnitudes.

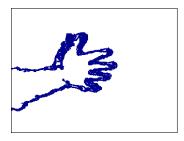


Figure 5.4: Thresholded derivative magnitudes of the depth channel indicated as blue pixels. These pixels are used for removing the invalid derivatives in the depth channel.

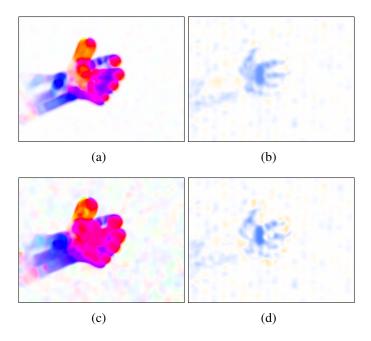


Figure 5.5: Combined local-global range flow with diffusivity \mathcal{D}_3 where invalid depth areas are removed prior to estimation. (a) Magnitude of the optic uv flow, with the invalid areas excluded, 10 iterations (b) Magnitude of the w component of the range flow, with the invalid areas excluded, 10 iterations (c) Magnitude of the optic uv flow, with the invalid areas excluded, 100 iterations (d) Magnitude of the w component of the range flow, with the invalid areas excluded, 100 iterations.

Following the optic flow approach of [2], range flow estimator is formulated as a combined local-global model:

$$E(u, v, w) = \int \left(K_{\rho} * \left((f_x u + f_y v + f_t)^2 \right) + K_{\phi} * \left(\beta (g_x u + g_y v + w + g_t)^2 \right) + \alpha \mathcal{V}(\nabla u, \nabla v, \nabla w) \right) dx dy,$$

$$(5.5)$$

where K_{ρ} , K_{ϕ} are Gaussian smoothing kernels, with parameters $\rho, \phi > 0$.

Regularizers are, as before:

• homogeneous regularizer [12]:

$$\mathcal{V}_1(\nabla u, \nabla v, \nabla w) = \Psi(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2)$$

= $|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2$, (5.6)

• nonlinear isotropic regularizer [42] e.g. Charbonnier regularizer:

$$\mathcal{V}_{2}(\nabla u, \nabla v, \nabla w) = \Psi(|\nabla u|^{2} + |\nabla v|^{2} + |\nabla w|^{2})$$

$$= 2\lambda^{2}\sqrt{1 + \frac{|\nabla u|^{2} + |\nabla v|^{2} + |\nabla w|^{2}}{\lambda^{2}}} - 2\lambda^{2},$$
(5.7)

• nonlinear anisotropic regularizer [54]:

$$\mathcal{V}_3(\nabla u, \nabla v, \nabla w) = \operatorname{tr} \Psi(\nabla u \nabla u^T + \nabla v \nabla v^T + \nabla w \nabla w^T), \tag{5.8}$$

where matching diffusivities are:

$$\mathcal{D}_1(\nabla u, \nabla v, \nabla w) = \Psi'(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2)$$

$$= 1,$$
(5.9)

$$\mathcal{D}_{2}(\nabla u, \nabla v, \nabla w) = \Psi'(|\nabla u|^{2} + |\nabla v|^{2} + |\nabla w|^{2})$$

$$= \frac{1}{1 + \frac{|\nabla u|^{2} + |\nabla v|^{2} + |\nabla w|^{2}}{\lambda^{2}}},$$
(5.10)

$$\mathcal{D}_3(\nabla u, \nabla v, \nabla w) = \Psi'(\nabla u \nabla u^T + \nabla v \nabla v^T + \nabla w \nabla w^T). \tag{5.11}$$

The corresponding diffusion-reaction system is then:

$$u_{t} = \operatorname{div}\left(\mathcal{D}\left(\nabla u, \nabla v, \nabla w\right) \nabla u\right)$$

$$-\frac{1}{\alpha} \left(K_{\rho} * (f_{x})^{2} u + K_{\rho} * (f_{x}f_{y})v + K_{\rho} * (f_{x}f_{t})\right)$$

$$+\beta \left(K_{\phi} * (g_{x})^{2} u + K_{\phi} * (g_{x}g_{y})v + K_{\phi} * (g_{x})w + K_{\phi} * (g_{x}g_{t})\right)\right),$$

$$v_{t} = \operatorname{div}\left(\mathcal{D}\left(\nabla u, \nabla v, \nabla w\right) \nabla v\right)$$

$$-\frac{1}{\alpha} \left(K_{\rho} * (f_{y}f_{x})u + K_{\rho} * (f_{y})^{2}v + K_{\rho} * (f_{y}f_{t})\right)$$

$$+\beta \left(K_{\phi} * (g_{y}g_{x})u + K_{\phi} * (g_{y})^{2}v + K_{\phi} * (g_{y})w + K_{\phi} * (g_{y}g_{t})\right)\right),$$

$$w_{t} = \operatorname{div}\left(\mathcal{D}\left(\nabla u, \nabla v, \nabla w\right) \nabla w\right)$$

$$-\frac{1}{\alpha} \left(\beta \left(K_{\phi} * g_{x}u + K_{\phi} * g_{y}v + w + K_{\phi} * g_{t}\right)\right).$$
(5.12)

The equation system (5.12) differs from system (A.42) in convolution of the derivatives with the Gaussian kernels in the reaction term. The local smoothing i.e. flow constancy assumption within a neighborhood of a size ρ and ϕ introduces robustness to noise.

5.3.1 Results

Figure 5.5 shows the result of the combined local-global range flow with excluded invalid edges. The flow estimation exhibits no artifacts.

Figure 5.6 shows a second test scene depicting an office. It is passive scene with the camera moving towards the table. Lower row shows the result of the proposed method. Larger number of iterations produce a fill-in effect as a consequence of the regularization term.

Figure 5.7 shows an arrow plot of the estimated depth component of the range flow. In order to show that the estimation is correctly located, the arrow plot is overlayed with magnitude and color channel of the input images.

In both datasets, noise produced by the invalid edges (Figure 5.3, Figure 5.6 middle row) is eliminated (Figure 5.5, Figure 5.6 lower row). Compared to the results of the standard method a more regular non-distorted result closer to the expected movement is observed.

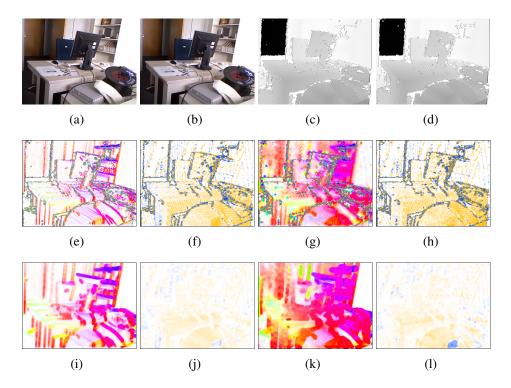


Figure 5.6: Upper row: Input pairs of images. (a), (b) color channels, (c), (d) depth channels. Middle row: Unmodified range flow with original depth data as input. Influence of noise is visible at the edges. Range flow with diffusivity \mathcal{D}_2 . (e), (f) magnitude of the uv-flow, depth w-flow, 10 iterations, (g), (h) magnitude of the uv-flow, depth w-flow, 300 iterations. Lower row: Combined local-global range flow with adjusted depth image derivatives, where invalid depth areas are removed prior to estimation. Range flow with diffusivity \mathcal{D}_3 . $\rho = 6$, $\phi = 6$. (i), (j) magnitude of the uv-flow, depth w-flow, 10 iterations, (k), (l) magnitude of the uv-flow, depth w-flow, 300 iterations.

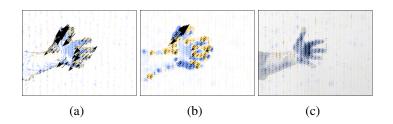


Figure 5.7: Arrow plot of the depth w-flow with arrows oriented at (1,1) instead of "into" the paper. Images are overlayed with magnitude and color channel (c) of the flow. Range flow with diffusivity \mathcal{D}_3 . 10 iterations. (a) Standard range flow method. The influence of invalid depth areas is corrupting the estimation of the depth w-flow. (b) Combined local-global range flow with invalid areas. (c) Combined local-global range flow without invalid areas is giving a correct estimation of the movement.

5.4 Ensemble Range Flow

Analysis and study of the fluid flow often requires the information about the average motion of the flow in time. When applying the techniques such as particle image velocimetry [35] or optic flow it is common to average the vector fields after recording the vector data. An ensemble optic flow approach proposed in [45] is based on averaging the structure tensor of the large data set before the flow estimation. The approach gives better average flow estimates and is more robust against outliers.

Here, an ensemble (combined local-global) range flow is proposed for estimation of the 3D flow. By additionally averaging the structure tensor throughout the entire data set an ensemble range flow is formulated as:

$$E(u, v, w) = \int \left(K_{\eta} * \left(K_{\rho} * \left((f_{x}u + f_{y}v + f_{t})^{2} \right) + K_{\phi} * \left(\beta (g_{x}u + g_{y}v + w + g_{t})^{2} \right) \right) + \alpha \mathcal{V}(\nabla u, \nabla v, \nabla w) \right) dx dy,$$

$$(5.13)$$

where K_{η} is Gaussian smoothing kernel over all frames. In addition to averaging the gradients of two consecutive frames prior to flow estimation as proposed in a combined local-global range flow, the gradients are also averaged over the entire dataset. Figure 5.10 shows an ensemble range flow of the infinity gesture images set.

Appendix B applies the ensemble range flow (5.13) to sets of fluid flow images. Such sets do not contain the depth data, so the depth information is estimated by using one of the "depth from monocular images" algorithms.

Vortex core extraction of a gesture ensemble range flow will be used here to assist the gesture classification.

5.5 Vortex Detection of Ensemble Range Flow for Gesture Classification

Vortex detection of gesture ensemble range flow is proposed as a method for improvement of gesture classification. Five persons recorded nine chosen gesture videos using the Microsoft Kinect device. Vortices are extracted from the ensemble range flow of the gestures. Number of vortices serves as an additional aid for gesture classification.

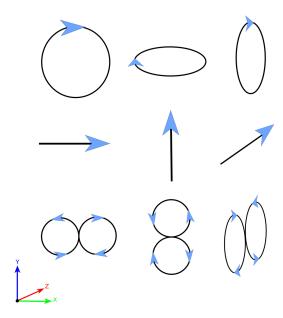


Figure 5.8: Upper row: circle in xy plane, circle in xz plane, circle in yz plane, Middle row: sweep left, sweep up, sweep "in" camera, Lower row: horizontal infinity loop in xy plane, vertical infinity loop in xy plane, horizontal infinity loop in yz plane.

Gestures Gestures are recorded as short videos using Kinect. Each gesture contains 60 to 90 frames of 640×480 rgb images (color channels) and the equal number of depth images (depth channels). Nine gestures are recorded by 5 subjects, making a total of 45 test gestures. Gestures which can be classified via vortex extraction are chosen as test gestures.

Chosen gestures are (Figure 5.8):

- circle in xy plane, circle in xz plane, circle in yz plane,
- horizontal sweep in xy plane (sweep left), vertical sweep in xy plane (sweep up), horizontal sweep in yz plane (sweep "in" camera)
- horizontal infinity loop in xy plane, vertical infinity loop in xy plane, horizontal infinity loop in yz plane

Gesture Range Flow The optic/range flow between two consecutive frames reveals very limited information about the gesture performed. Figure 5.9 shows the optic flow from three points in time of an infinity gesture.

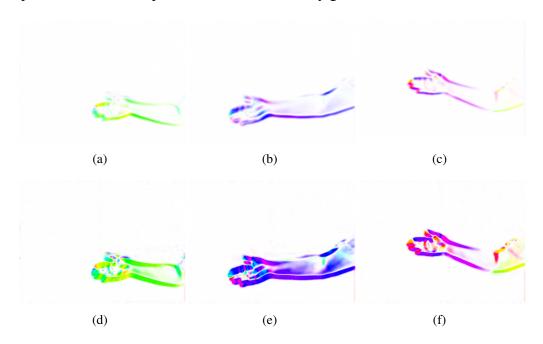


Figure 5.9: Magnitude of the optic flow for frames 19, 32, 43 of one of the infinity gestures. Minimization with the Charbonnier regularization (5.7). Upper row: 1 iteration, lower row: 10 iterations.

Gesture Ensemble Range Flow An ensemble flow (5.13) of the entire scene gives an insight into the flow of the whole gesture. Derivatives are summed up through the entire scene and the structure tensor is averaged to produce the ensemble flow (Figure 5.10). Depending on the setup of the minimization parameters, the regularization is set to weaker or stronger, the latter producing vector fields suitable for detection of critical points (Figure 5.10(b), 5.10(d)). Figure 5.11 shows ensemble uv-flows and depth w-flows of the gestures from Figure 5.8. "3D" gestures i.e the gestures with movement in z direction (right column of Figure 5.8 plus xz circle) have more intensive depth flow component.

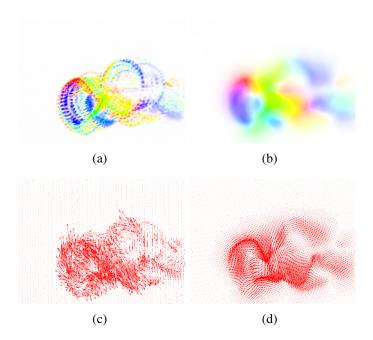


Figure 5.10: Ensemble flow of the infinity gesture with weaker (left column) and stronger regularization (right column).

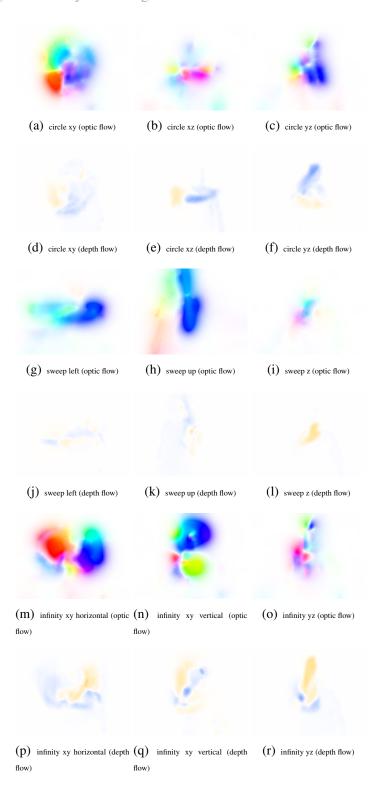


Figure 5.11: Ensemble uv-flows (color coded by image 5.1(a)) and depth w-flows (color interpolated between blue for backward and orange for forward) of the gestures from Figure 5.8. The depth flow is more intensive in "3D" gestures (ones where the movement in z direction is present) although it is always visible due to the imperfection of the user's interpretation.

Vortex Detection of an Ensemble Range Flow Figure 5.12 shows the process of detecting critical points in gesture ensemble range flows. First, an area with the gesture is selected. An area with higher flow magnitude is thresholded, then "expanded" using simple Gaussian blurring or morphological opening. These operations result in a mask that denotes the area of an image where the main gesture movement lies. Critical points are detected by determining the lines in the flow where flow components change sign. Critical points are at places (within the gesture mask) where the resulting lines of the two flow components cross. The nature of the critical point is then determined. The swirl exists if the critical point is a focus or a center i.e. if there are two conjugated complex eigenvalues of the Jacobian matrix in that point. Such critical points are 2D vortices. Two vortex cores are extracted from the infinity gesture, as expected.

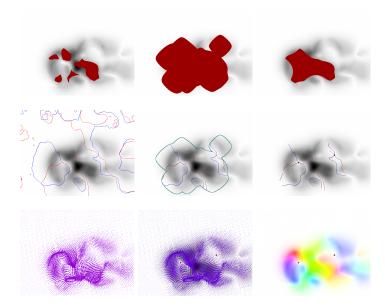


Figure 5.12: Detecting vortices in infinity gesture ensemble range flow. Magnitude of the flow is denoted in gray. Upper row: Left image shows thresholded high magnitude of the flow denoted red. Middle image shows the threshold area expanded by Gaussian blurring. Right image shows the threshold area expanded by morphological opening. Blurring is chosen as the default method. Middle row: Left image shows lines in the flow where flow components change sign. Middle image shows the same lines within the determined gesture area (cyan edge). Right image shows the detected critical points, where the resulting lines of the two flow components cross, denoted as red points. Both points are of the focus type, so none of them is excluded. Lower row shows the arrow and the magnitude plots of the original flow together with the vortex centers.

5.5.1 Results

Figure 5.13 shows the vortex detection of ensemble flows of the gestures from Figure 5.8. As expected, circular gestures have one vortex center detected, sweep gestures none. Horizontal infinity gesture has, as wanted, two vortices. However, the detection of the vertical infinity gesture gives only one vortex, and detection of the yz infinity gesture gives three vortices.

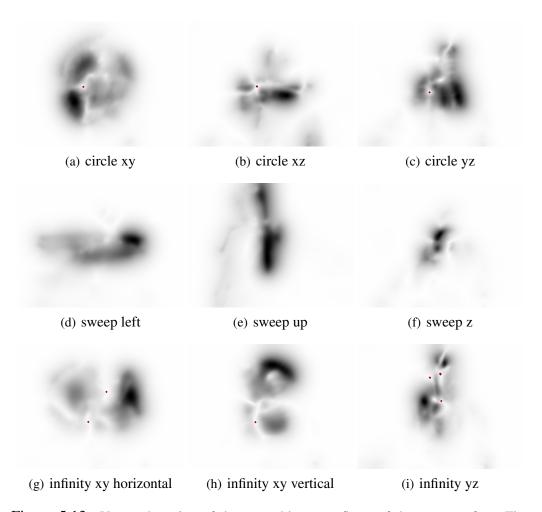


Figure 5.13: Vortex detection of the ensemble range flows of the gestures from Figure 5.8. Upper row shows circular gestures with one vortex center detected. Middle row shows sweep gestures with no vortex centers detected. Lower row shows infinity gestures. Horizontal gesture has, as expected, two vortices. One vortex is detected in vertical gesture (h). Three vortices are detected in yz gesture (i). This issue is addressed in upcoming text.

By filtering out the very steep foci (foci that are near sink or near source) from the data, two vortices are obtained in yz infinity gesture (Figure 5.14). What is desired in a gesture flow are non-steep vortices. Sinks or sources with a slight swirl (steep foci) can be filtered out. Steepness of a focus can be determined by looking at the magnitude of the imaginary part of eigenvalues of Jacobian matrix. Smaller magnitude results in a steeper focus. By thresholding this magnitude, only non-steep vortices are kept.



Figure 5.14: Steep foci critical points (foci that are near sink or near source) are filtered out from a yz infinity gesture giving only two remaining critical points as opposed to three points detected before (Figure 5.13 (i)). Steep foci are the ones with small magnitudes of imaginary parts of eigenvalues of the Jacobian matrix.

Only one critical point is detected in vertical infinity gesture flow from Figure 5.11 (n). If different vertical infinity gesture is considered (Figure 5.15), three vortices are obtained. After filtering out the steep (near sink/source) foci two vortices are obtained.

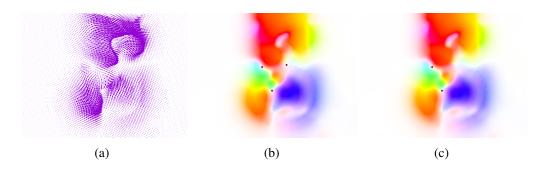


Figure 5.15: Alternative vertical infinity gesture considered. (b) Vortex detection results in three vortices. (c) Filtering out the steep vortices results in two points.

The detection of vortices in the ensemble flow depends on many parameters. Selecting the suitable gesture region and selecting the appropriate critical points within it, could be repeatedly preformed and optimized. In the future, it is planned to make the vortex detection more robust.

Although it could be used as a standalone method, vortex extraction of an ensemble range flow is designed to be used in combination with gesture classification methods. Methods that reduce the dimensionality of the data (e.g. [18]) can produce similar data for different gestures. Gesture classification based on summed magnitudes of the flow is proposed for future work. The idea is presented in the text below. Non-perfect user performance of the gesture is leading to certain flow magnitudes producing similar graphs. Detection of vortices should resolve the uncertain cases.

Gesture Classification Following gesture classification method is proposed. Magnitude of the gesture ensemble flow (including u,v and w) is summed up through every x and y coordinate (Figure 5.16). A histogram-like plot of the gesture is drawn, so decreasing a number of parameters to work with. Similar approach was taken in [18]. The graph depicting x, then y summed magnitude values has 640 + 480 = 1120 x entries. The graph can be further simplified i.e. its dimension can be reduced if PCA is used.

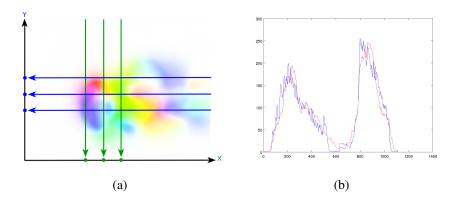


Figure 5.16: Summing up of the magnitude of the flow through every x and y coordinate produces a histogram-like graph. The x magnitudes are followed by the y magnitudes resulting in a two-peaked graph (blue). The red graph is a PCA reconstruction of the same graph with reduced number of basis function (from 1120 to 2).

Such 2D graphs can be used to classify the gestures using e.g. K-nearest neighbors (KNN). The problem when having a small number of data is the following. It is sometimes not clear which gesture is looked at i.e. classification is incorrect (Figure 5.17). If the users circle gesture is elliptical and has similar shape as an infinity gesture, resulting graphs can be difficult to distinguish from one another. This uncertainties can be resolved by detecting the critical points of the ensemble range flow of the gesture.

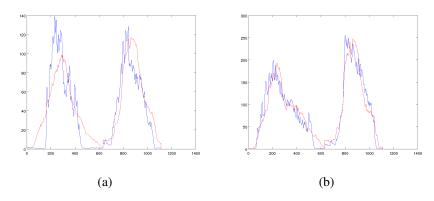


Figure 5.17: Graph for the circle gesture (a) and for the infinity gesture (b) in xy plane together with the red PCA reconstruction. If user performs the circle gesture as an ellipse similar to the infinity gesture, the graphs will also be similar and difficult to distinguish from one another.

5.5.2 Vortex Preserving Diffusion of the Ensemble Range Flow

For the purpose of gesture classification, only the number of vortices within the flow is required. If one wants to obtain an ensemble flow field of the gesture with emphasized vortices, vortex preserving diffusion techniques can be used.

If diffusion presented in Section 3.4.1 is run on the gesture data, the flows with emphasized swirling areas are obtained (Figure 5.18 rows 3, 4 from above). It would, however, make more sense to keep only the area surrounding the vortices. After detection of vortices, circular area around them is kept while the rest of the flow is blurred (Figure 5.18 lower half, rows 5, 6, 7, 8). This is also a nonlinear isotropic diffusion process with binary diffusivity set to 1 outside and 0 inside the vortex area. Different sizes of the vortex area can be used.

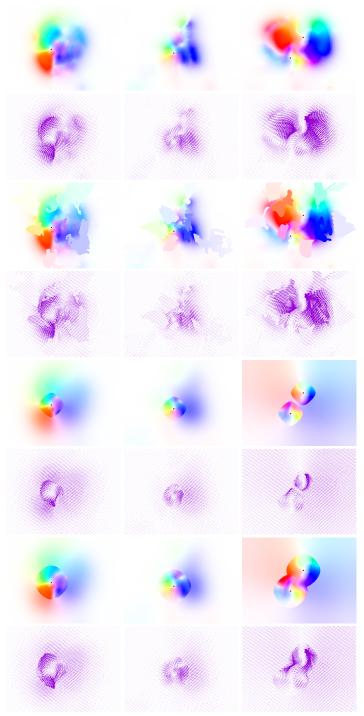


Figure 5.18: Nonlinear isotropic diffusion process on a gesture ensemble range flow with binary diffusivities steering the diffusion. Column 1: circle in xy plane, column 2: circle in yz plane, column 3: horizontal infinity loop in xy plane. Rows 1,2 (upper): magnitude and arrow plots of the three gestures. Rows 3,4: nonlinear isotropic diffusion process with diffusivity (3.27) for 5000 iterations. Rows 5,6: circular area surrounding the vortices is kept, the rest of the flow is blurred. This is also a nonlinear isotropic diffusion process with binary diffusivity set to 1 outside and 0 inside the vortex area. Rows 7,8: bigger circular area surrounding the vortices is chosen.

Chapter 6

Conclusion and Future Work

Detecting and locating vortices is crucial for understanding of the flow. Different methods for enhancing and denoting vortical structures are presented in this thesis.

Real-time vortex detection offers instant insight into vortical structures of the simulated flow data. Extraction methods are implemented within the fluid simulation. Vortices are located and denoted by color coding of the parts of the flow volume/arrow plots. Advantage of the presented real-time methods is immediate notion of the shape and location of the vortex structures. Vortex detection is no longer performed as a post processing step and vortex cores or regions are instantly denoted.

Diffusional and variational methods have the goal of obtaining vector fields in which the vortical structures are preserved and emphasized. The resulting vector fields offer an alternative insight into the structure of the flow. Diffusion and diffusion-reaction methods which produce vector fields with emphasized vortex regions are introduced.

Ensemble range flow shows the overall movement in a scene. Ensemble range flow of gestures is estimated and used to improve gesture classification methods. Non-perfect user performance of the gesture is leading to uncertainties within gesture classification. Detection of vortices within the ensemble flow helps resolving such cases.

6.1 Future Work

Diffusional and Variational Methods Diffusion and diffusion-reaction methods require setting of many parameters. Determining optimal parameters for vortex preservation is planned in future. Using advanced anisotropic regularization and implementing fast numerical non-iterative schemes shall be used to speed up the algorithms up to the real-time performance.

Real-Time Vortex Detection Optimization of the GPU implementation of the real-time fluid simulation could be achieved by using OpenGL CUDA interoperability, as well as by implementing faster numerical solvers for the simulation.

The applicability of the real-time vortex extraction methods could be increased by allowing not only simulated, but also real world data to be processed.

An alternative real-time vortex extraction method could be designed by replacing the diffusion within the real-time fluid simulation with diffusion and diffusion-reaction approaches designed in Chapter 3.

Fluid interaction with a moving obstacle could give an insight into vortices forming around an object. Figure 6.1 shows a butterfly flapping its wings and interacting with the fluid within a real-time simulation.



Figure 6.1: Object-fluid interaction.

Gesture Ensemble Range Flow Detection of vortices in the ensemble flow depends on many parameters. Calculation of the ensemble range flow itself, already has many parameters. Additionally, a gesture region has to be selected and then the suitable critical points within it. Automatic detection of optimal parameters and more robust vortex detection is planned.

Developing a gesture classification method based on reduction of data dimensions by using summed magnitudes of the ensemble range flow is planned.

Immersing the gesture detection into the real-time fluid simulation and taking advantage of the real-time vortex detection techniques is planned. Range flow between two frames gives a 3D flow in a plane (2.5D). Stashing such flows together gives a volume of 3D flows. Ensemble flow is mapping this "volume" into a single 3D flow. Operating within a volume of 3D flows would remove the need for an accumulative process, such as ensemble flow, and would allow real-time performance.

Appendices

Appendix A

Mathematical Formulations Leading to a Discrete Explicit Scheme

A.1 Finite Difference Derivative Approximations

Some of the most frequently used approximations are given here. Approximation of the first derivative with central differences:

$$\partial_x u \approx \frac{u_{i+1,j} - u_{i-1,j}}{2h_1},$$
 (A.1)

$$\partial_y u \approx \frac{u_{i,j+1} - u_{i,j-1}}{2h_2},\tag{A.2}$$

$$\partial_t u \approx \frac{u_{i,j}^{k+1} - u_{i,j}^k}{\tau},$$
 (A.3)

where $h1, h2, \tau$ are pixel sizes in x, y and time direction. Usually the pixel sizes h_1 and h_2 are taken as equal.

Approximation of the second partial derivatives:

$$\partial_{x}(b \,\partial_{x}u) \approx \frac{1}{h_{1}} \left(\frac{b_{i+1,j} + b_{i,j}}{2} \frac{u_{i+1,j} - u_{i,j}}{h_{1}} - \frac{b_{i,j} + b_{i-1,j}}{2} \frac{u_{i,j} - u_{i-1,j}}{h_{1}} \right),$$

$$\partial_{y}(b \,\partial_{y}u) \approx \frac{1}{h_{2}} \left(\frac{b_{i,j+1} + b_{i,j}}{2} \frac{u_{i,j+1} - u_{i,j}}{h_{2}} - \frac{b_{i,j} + b_{i,j-1}}{2} \frac{u_{i,j} - u_{i,j-1}}{h_{2}} \right).$$
(A.4)

Approximation of the second partial derivatives with constant b = 1:

$$\partial_{xx} u \approx \frac{u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k}{h_1^2},$$

$$\partial_{yy} u \approx \frac{u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k}{h_2^2}.$$
(A.5)

Approximation of the mixed terms of second partial derivatives:

$$\partial_x(b \,\partial_y u) \approx \frac{1}{2h_1} \left(b_{i+1,j} \frac{u_{i+1,j+1} - u_{i+1,j-1}}{2h_2} - b_{i-1,j} \frac{u_{i-1,j+1} - u_{i-1,j-1}}{2h_2} \right),$$

$$\partial_y(b \,\partial_x u) \approx \frac{1}{2h_2} \left(b_{i,j+1} \frac{u_{i+1,j+1} - u_{i-1,j+1}}{2h_1} - b_{i,j-1} \frac{u_{i+1,j-1} - u_{i-1,j-1}}{2h_1} \right).$$
(A.6)

Approximation of the mixed terms with constant b = 1:

$$\partial_{xy}u \approx \frac{1}{2h_1} \left(\frac{u_{i+1,j+1} - u_{i+1,j-1}}{2h_2} - \frac{u_{i-1,j+1} - u_{i-1,j-1}}{2h_2} \right) \approx$$

$$\partial_{yx}u \approx \frac{1}{2h_2} \left(\frac{u_{i+1,j+1} - u_{i-1,j+1}}{2h_1} - \frac{u_{i+1,j-1} - u_{i-1,j-1}}{2h_1} \right).$$
(A.7)

If there is a quadratic error term in the grid size h, then the approximation has a consistency order 2. Approximations must have a consistency order of at least 1. If not, they are inconsistent and inappropriate.

A.2 Boundary Conditions

In practice the domain Ω is a rectangle with boundary $\partial\Omega$. Boundary conditions imposed are usually:

- Dirichlet boundary conditions i.e. setting the velocity to zero at boundaries: u=0 on $\partial\Omega$,
- Neumann boundary conditions: $\frac{\partial u}{\partial n} = 0$ on $\partial \Omega$, where n is the outer normal vector on $\partial \Omega$.

A.3 Linear Isotropic Diffusion

By discretizing the diffusion equation, unknown $\boldsymbol{u}_{i,j}^{k+1}$ can be explicitly computed:

$$\partial_{t}u = \Delta u$$

$$\partial_{t}u = \partial_{xx}u + \partial_{yy}u$$

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\tau} = \frac{u_{i+1,j}^{k} - 2u_{i,j}^{k} + u_{i-1,j}^{k}}{h_{1}^{2}} + \frac{u_{i,j+1}^{k} - 2u_{i,j}^{k} + u_{i,j-1}^{k}}{h_{2}^{2}}$$

$$u_{i,j}^{k+1} = \left(1 - 2\frac{\tau}{h_{1}^{2}} - 2\frac{\tau}{h_{2}^{2}}\right)u_{i,j}^{k}$$

$$+ \frac{\tau}{h_{1}^{2}}u_{i+1,j}^{k} + \frac{\tau}{h_{1}^{2}}u_{i-1,j}^{k} + \frac{\tau}{h_{2}^{2}}u_{i,j+1}^{k} + \frac{\tau}{h_{2}^{2}}u_{i,j-1}^{k}$$
(A.8)

This explicit scheme can be expressed as a 3×3 computation stencil on a 2D grid:

0	$rac{ au}{h_2^2}$	0
$\frac{\tau}{h_1^2}$	$1 - 2\frac{\tau}{h_1^2} - 2\frac{\tau}{h_2^2}$	$\frac{\tau}{h_1^2}$
0	$rac{ au}{h_2^2}$	0

or, if pixel sizes are taken to be one:

0	au	0
τ	$1-4\tau$	τ
0	au	0

The scheme is stable for $\tau \leq \frac{1}{4}$. For larger values the scheme changes its nature away from a smoothing scheme.

A.4 Backward Linear Isotropic Diffusion

Linear isotropic diffusion can be written as update of the current pixel with the weighted neighbor influence:

$$u_{i,j}^{k+1} = (1 - 4\tau) u_{i,j}^{k} + \tau \left(u_{i+1,j}^{k} + u_{i-1,j}^{k} + u_{i,j+1}^{k} + u_{i,j-1}^{k} \right), u_{i,j}^{k+1} = u_{i,j}^{k} + \tau \left(u_{i+1,j}^{k} + u_{i-1,j}^{k} + u_{i,j+1}^{k} + u_{i,j-1}^{k} - 4u_{i,j}^{k} \right).$$
(A.9)

Backward diffusion is then the difference of the current pixel and the weighted neighbor influence:

$$u_{i,j}^{k} = u_{i,j}^{k+1} - \tau \left(u_{i+1,j}^{k+1} + u_{i-1,j}^{k+1} + u_{i,j+1}^{k+1} + u_{i,j-1}^{k+1} - 4u_{i,j}^{k+1} \right). \tag{A.10}$$

 $u_{i,j}^{k+1}$ is then expressed as:

$$u_{i,j}^{k+1} = \frac{u_{i,j}^k + \tau \left(u_{i+1,j}^{k+1} + u_{i-1,j}^{k+1} + u_{i,j+1}^{k+1} + u_{i,j-1}^{k+1} \right)}{(1+4\tau)}.$$
 (A.11)

This is a sparse linear system which can be solved by e.g. using Gauss-Seidel relaxation.

A.5 Nonlinear Isotropic Diffusion

By discretizing the diffusion equation, unknown $u_{i,j}^{k+1}$ can be explicitly computed:

$$\begin{split} \partial_{t}u &= \operatorname{div}\left(g(|\nabla u|^{2})\nabla u\right) \\ \partial_{t}u &= \partial_{x}\left(g(|\nabla u|^{2})\partial_{x}u\right) + \partial_{y}\left(g(|\nabla u|^{2})\partial_{y}u\right) \\ \frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\tau} &= \frac{1}{h_{1}}\left(\frac{g_{i+1,j} + g_{i,j}}{2} \frac{u_{i+1,j} - u_{i,j}}{h_{1}} - \frac{g_{i,j} + g_{i-1,j}}{2} \frac{u_{i,j} - u_{i-1,j}}{h_{1}}\right) \\ &+ \frac{1}{h_{2}}\left(\frac{g_{i,j+1} + g_{i,j}}{2} \frac{u_{i,j+1} - u_{i,j}}{h_{2}} - \frac{g_{i,j} + g_{i,j-1}}{2} \frac{u_{i,j} - u_{i,j-1}}{h_{2}}\right) \\ u_{i,j}^{k+1} &= \left(1 - \frac{\tau}{h_{1}^{2}} \left(\frac{g_{i+1,j} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i-1,j}}{2}\right) - \frac{\tau}{h_{2}^{2}} \left(\frac{g_{i,j+1} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i,j-1}}{2}\right)\right) u_{i,j}^{k} \\ &+ \left(\frac{g_{i+1,j} + g_{i,j}}{2} \frac{\tau}{h_{1}^{2}}\right) u_{i+1,j}^{k} + \left(\frac{g_{i,j} + g_{i-1,j}}{2} \frac{\tau}{h_{1}^{2}}\right) u_{i-1,j}^{k} \\ &+ \left(\frac{g_{i,j+1} + g_{i,j}}{2} \frac{\tau}{h_{2}^{2}}\right) u_{i,j+1}^{k} + \left(\frac{g_{i,j} + g_{i,j-1}}{2} \frac{\tau}{h_{2}^{2}}\right) u_{i,j-1}^{k}, \end{split}$$

where $g_{i,j}$ is an approximation of the diffusivity $g(|\nabla u|^2)$ in pixel (i,j). The scheme is stable for $\tau < \frac{h^2}{4}$ (in 2D).

The explicit scheme can be expressed as a 3×3 computation stencil on a 2D grid:

0	$\frac{g_{i,j+1}+g_{i,j}}{2}\frac{\tau}{h_2^2}$	0
$\frac{g_{i,j}+g_{i-1,j}}{2}\frac{\tau}{h_1^2}$	$1 - \frac{\tau}{h_1^2} \left(\frac{g_{i+1,j} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i-1,j}}{2} \right) - \frac{\tau}{h_2^2} \left(\frac{g_{i,j+1} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i,j-1}}{2} \right)$	$\frac{g_{i+1,j}+g_{i,j}}{2}\frac{\tau}{h_1^2}$
0	$rac{g_{i,j}+g_{i,j-1}}{2}rac{ au}{h_2^2}$	0

A.6 Backward Nonlinear Isotropic Diffusion

Nonlinear isotropic diffusion can be written as update of the current pixel with the weighted neighbor influence:

$$u_{i,j}^{k+1} = \left(1 - \tau \left(\frac{g_{i+1,j} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i-1,j}}{2} + \frac{g_{i,j+1} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i,j-1}}{2}\right)\right) u_{i,j}^{k}$$

$$+ \tau \left(\left(\frac{g_{i+1,j} + g_{i,j}}{2}\right) u_{i+1,j}^{k} + \left(\frac{g_{i,j} + g_{i-1,j}}{2}\right) u_{i-1,j}^{k}\right)$$

$$+ \left(\frac{g_{i,j+1} + g_{i,j}}{2}\right) u_{i,j+1}^{k} + \left(\frac{g_{i,j} + g_{i,j-1}}{2}\right) u_{i,j-1}^{k}\right),$$

$$u_{i,j}^{k+1} = u_{i,j}^{k} + \tau \left(\left(\frac{g_{i+1,j} + g_{i,j}}{2}\right) u_{i+1,j}^{k} + \left(\frac{g_{i,j} + g_{i-1,j}}{2}\right) u_{i-1,j}^{k}\right)$$

$$+ \left(\frac{g_{i,j+1} + g_{i,j}}{2}\right) u_{i,j+1}^{k} + \left(\frac{g_{i,j} + g_{i,j-1}}{2}\right) u_{i,j-1}^{k}$$

$$- \left(\frac{g_{i+1,j} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i-1,j}}{2} + \frac{g_{i,j+1} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i,j-1}}{2}\right) u_{i,j}^{k}\right).$$
(A.13)

Backward diffusion is then the difference of the current pixel and the weighted neighbor influence:

$$u_{i,j}^{k} = u_{i,j}^{k+1} - \tau \left(\left(\frac{g_{i+1,j} + g_{i,j}}{2} \right) u_{i+1,j}^{k+1} + \left(\frac{g_{i,j} + g_{i-1,j}}{2} \right) u_{i-1,j}^{k+1} \right.$$

$$\left. + \left(\frac{g_{i,j+1} + g_{i,j}}{2} \right) u_{i,j+1}^{k+1} + \left(\frac{g_{i,j} + g_{i,j-1}}{2} \right) u_{i,j-1}^{k+1}$$

$$\left. - \left(\frac{g_{i+1,j} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i-1,j}}{2} + \frac{g_{i,j+1} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i,j-1}}{2} \right) u_{i,j}^{k+1} \right).$$
(A.14)

 $u_{i,j}^{k+1}$ is then expressed as:

$$u_{i,j}^{k+1} = \frac{u_{i,j}^{k} + \tau \left((\frac{g_{i+1,j} + g_{i,j}}{2}) u_{i+1,j}^{k+1} + (\frac{g_{i,j} + g_{i-1,j}}{2}) u_{i-1,j}^{k+1} + (\frac{g_{i,j+1} + g_{i,j}}{2}) u_{i,j+1}^{k+1} + (\frac{g_{i,j} + g_{i,j-1}}{2}) u_{i,j-1}^{k+1} \right)}{\left(1 + (\frac{g_{i+1,j} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i-1,j}}{2} + \frac{g_{i,j+1} + g_{i,j}}{2} + \frac{g_{i,j} + g_{i,j-1}}{2}) \tau \right)}, \\ u_{i,j}^{k+1} = \frac{u_{i,j}^{k} + \tau \left((\frac{g_{i+1,j} + g_{i,j}}{2}) u_{i+1,j}^{k+1} + (\frac{g_{i,j} + g_{i-1,j}}{2}) u_{i-1,j}^{k+1} + (\frac{g_{i,j+1} + g_{i,j}}{2}) u_{i,j+1}^{k+1} + (\frac{g_{i,j} + g_{i,j-1}}{2}) u_{i,j-1}^{k+1} \right)}{\left(1 + (4g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}) \frac{\tau}{2} \right)}.$$
(A.15)

This is a sparse linear system which can be solved by e.g. using Gauss-Seidel relaxation.

A.7 Nonlinear Anisotropic Diffusion

By discretizing the diffusion equation, unknown $u_{i,j}^{k+1}$ can be explicitly computed:

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\tau} = \operatorname{div}\left(\begin{pmatrix} a & b \\ b & c \end{pmatrix} \nabla u\right) \\
= \partial_{x}(a\partial_{x}u + b\partial_{y}u) + \partial_{y}(b\partial_{x}u + c\partial_{y}u) \\
= \frac{1}{h_{1}}\left(\frac{a_{i+1,j} + a_{i,j}}{2} \frac{u_{i+1,j} - u_{i,j}}{h_{1}} - \frac{a_{i,j} + a_{i-1,j}}{2} \frac{u_{i,j} - u_{i-1,j}}{h_{1}}\right) \\
+ \frac{1}{2h_{1}}\left(b_{i+1,j} \frac{u_{i+1,j+1} - u_{i+1,j-1}}{2h_{2}} - b_{i-1,j} \frac{u_{i-1,j+1} - u_{i-1,j-1}}{2h_{2}}\right) \\
+ \frac{1}{2h_{2}}\left(b_{i,j+1} \frac{u_{i+1,j+1} - u_{i-1,j+1}}{2h_{1}} - b_{i,j-1} \frac{u_{i+1,j-1} - u_{i-1,j-1}}{2h_{1}}\right) \\
+ \frac{1}{h_{2}}\left(\frac{c_{i,j+1} + c_{i,j}}{2} \frac{u_{i,j+1} - u_{i,j}}{h_{2}} - \frac{c_{i,j} + c_{i,j-1}}{2} \frac{u_{i,j} - u_{i,j-1}}{h_{2}}\right)$$

The explicit scheme can be expressed as a 3×3 computation stencil on a 2D grid:

$\frac{-b_{i-1,j} - b_{i,j+1}}{4h_1h_2}$	$\frac{c_{i,j+1}{+}c_{i,j}}{2h_2^2}$	$\begin{array}{ c c }\hline b_{i+1,j}+b_{i,j+1}\\ 4h_1h_2\end{array}$
$\frac{a_{i-1,j} + a_{i,j}}{2h_1^2}$	$-\frac{a_{i-1,j}+2a_{i,j}+a_{i+1,j}}{2h_1^2} - \frac{c_{i,j-1}+2c_{i,j}+c_{i,j+1}}{2h_2^2}$	$\frac{a_{i+1,j} + a_{i,j}}{2h_1^2}$
$\frac{b_{i-1,j}+b_{i,j-1}}{4h_1h_2}$	$\frac{c_{i,j-1} + c_{i,j}}{2h_2^2}$	$\frac{-b_{i+1,j}-b_{i,j-1}}{4h_1h_2}$

In order for the minimum-maximum principle to be respected a better non-

negative stencil ([41])

$ \begin{array}{c} \frac{ b_{i-1,j+1} - b_{i-1,j+1}}{4h_1h_2} \\ + \frac{ b_{i,j} - b_{i,j}}{4h_1h_2} \end{array}$	$-\frac{\frac{c_{i,j+1}+c_{i,j}}{2h_2^2}}{-\frac{ b_{i,j+1} + b_{i,j} }{2h_1h_2}}$	$+\frac{ b_{i+1,j+1} + b_{i+1,j+1}}{4h_1h_2} + \frac{ b_{i,j} + b_{i,j}}{4h_1h_2}$
$-\frac{a_{i-1,j} + a_{i,j}}{2h_1^2} - \frac{ b_{i-1,j} + b_{i,j} }{2h_1h_2}$	$-\frac{a_{i-1,j}+2a_{i,j}+a_{i+1,j}}{2h_1^2}\\ -\frac{ b_{i-1,j+1} -b_{i-1,j+1}+ b_{i+1,j+1} +b_{i+1,j+1}}{4h_1h_2}\\ -\frac{ b_{i-1,j-1} +b_{i-1,j-1}+ b_{i+1,j-1} -b_{i+1,j-1}}{4h_1h_2}\\ +\frac{ b_{i-1,j} + b_{i+1,j} + b_{i,j-1} + b_{i,j+1} +4 b_{i,j} }{2h_1h_2}\\ -\frac{c_{i,j-1}+2c_{i,j}+c_{i,j+1}}{2h_2^2}$	$-\frac{\frac{a_{i+1,j}+a_{i,j}}{2h_1^2}}{\frac{ b_{i+1,j} + b_{i,j} }{2h_1h_2}}$
$\frac{ b_{i-1,j-1} + b_{i-1,j-1}}{4h_1h_2}$	$rac{c_{i,j-1}+c_{i,j}}{2h_2^2}$	$ \frac{ b_{i+1,j-1} - b_{i+1,j-1}}{4h_1h_2} $
$+\frac{ b_{i,j} +b_{i,j}}{4h_1h_2}$	$-rac{ b_{i,j-1} ilde{1}+ b_{i,j} }{2h_1h_2}$	$+\frac{ b_{i,j} -b_{i,j}}{4h_1h_2}$

Discretization of the Diffusion-Reaction System A.8

By discretizing the equation (3.24), unknown $u_{i,j}^{k+1}$ can be explicitly computed (the approximation is implicit in bias/reaction term):

$$\partial_t u = \operatorname{div}\left(\Psi'(|\nabla u|^2)\nabla u\right) - \frac{u-f}{\alpha}$$
 (A.17)

$$\partial_{t}u = \operatorname{div}\left(\Psi'(|\nabla u|^{2})\nabla u\right) - \frac{u - f}{\alpha}$$

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\tau} = \operatorname{div}\left(\Psi'(|\nabla u^{k}|^{2})\nabla u^{k}\right) - \frac{u^{k+1} - f}{\alpha}$$

$$u_{i,j}^{k+1} = \frac{u_{i,j}^{k} + \tau\left(\operatorname{div}\left(\Psi'(|\nabla u^{k}|^{2})\nabla u^{k}\right) + \frac{f}{\alpha}\right)}{\left(1 + \frac{\tau}{\alpha}\right)}$$
(A.18)

$$u_{i,j}^{k+1} = \frac{u_{i,j}^k + \tau \left(\operatorname{div}\left(\Psi'(|\nabla u^k|^2)\nabla u^k\right) + \frac{J}{\alpha}\right)}{\left(1 + \frac{\tau}{\alpha}\right)} \tag{A.19}$$

Depending on the form of the Ψ' function, discretization proceeds as previously. E.g. for homogeneous regularizer and penalizer $\Psi(|\nabla u|^2) = |\nabla u|^2$ and matching diffusivity $\Psi'(|\nabla u|^2) = 1$, discretization proceeds as follows:

$$u_{i,j}^{k+1} = \frac{u_{i,j}^k + \tau \left(\operatorname{div}\left(\nabla u^k\right) + \frac{f}{\alpha}\right)}{\left(1 + \frac{\tau}{\alpha}\right)} \tag{A.20}$$

$$u_{i,j}^{k+1} = \frac{u_{i,j}^k + \tau \left(u_{xx}^k + u_{yy}^k + \frac{f}{\alpha} \right)}{\left(1 + \frac{\tau}{\alpha} \right)} \tag{A.21}$$

$$u_{i,j}^{k+1} = \frac{u_{i,j}^k + \tau \left(\frac{u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k}{h_1^2} + \frac{u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k}{h_2^2} + \frac{f}{\alpha}\right)}{\left(1 + \frac{\tau}{\alpha}\right)}$$

$$u_{i,j}^{k+1} = \frac{\tau f}{\alpha + \tau} + \frac{u_{i,j}^k + \tau \left(\frac{u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k}{h_1^2} + \frac{u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k}{h_2^2}\right)}{\left(1 + \frac{\tau}{\alpha}\right)}$$
(A.22)

$$u_{i,j}^{k+1} = \frac{\tau f}{\alpha + \tau} + \frac{u_{i,j}^k + \tau \left(\frac{u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k}{h_1^2} + \frac{u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k}{h_2^2}\right)}{(1 + \frac{\tau}{\alpha})}$$
(A.23)

$$u_{i,j}^{k+1} = \frac{\tau f}{\alpha + \tau} + \frac{\left(1 - 2\frac{\tau}{h_1^2} - 2\frac{\tau}{h_2^2}\right) u_{i,j}^k + \frac{\tau}{h_1^2} u_{i+1,j}^k + \frac{\tau}{h_1^2} u_{i-1,j}^k + \frac{\tau}{h_2^2} u_{i,j+1}^k + \frac{\tau}{h_2^2} u_{i,j-1}^k}{\left(1 + \frac{\tau}{\alpha}\right)}$$

$$u_{i,j}^{k+1} = \frac{\tau f}{h_1^2} + \frac{\text{stencil } u^k}{h_1^2}, \tag{A.24}$$

with same 3×3 computation stencil as before for linear isotropic diffusion (Section 3.2):

0	$\frac{ au}{h_2^2}$	0
$\frac{\tau}{h_1^2}$	$1 - 2\frac{\tau}{h_1^2} - 2\frac{\tau}{h_2^2}$	$\frac{\tau}{h_1^2}$
0	$rac{ au}{h_2^2}$	0

The process is a blending between a diffusion and an original image.

A.9 Discriminant Steered Energy Functional Requesting Similarity to the Original Vector Field

Euler-Lagrange equations corresponding to the energy functional (3.31) are as follows:

$$0 = F_u - \partial_x F_{u_x} - \partial_y F_{u_y},$$

$$0 = F_v - \partial_x F_{v_x} - \partial_y F_{v_y}.$$
(A.25)

Using the notation:

$$F = \left(H\left((u - u_{orig})^2 + (v - v_{orig})^2\right) + \alpha \Psi\left(|\nabla u|^2 + |\nabla v|^2\right)\right),$$

$$F_u = 2H(u - u_{orig}),$$

$$F_v = 2H(v - v_{orig}),$$

$$F_{u_x} = 2\alpha u_x \Psi'\left(|\nabla u|^2 + |\nabla v|^2\right),$$

$$F_{u_y} = 2\alpha u_y \Psi'\left(|\nabla u|^2 + |\nabla v|^2\right),$$

$$F_{v_x} = 2\alpha v_x \Psi'\left(|\nabla u|^2 + |\nabla v|^2\right),$$

$$F_{v_x} = 2\alpha v_x \Psi'\left(|\nabla u|^2 + |\nabla v|^2\right),$$

$$F_{v_y} = 2\alpha v_y \Psi'\left(|\nabla u|^2 + |\nabla v|^2\right),$$

following equations are obtained:

$$0 = 2H(u - u_{orig}) - 2\alpha \partial_x \left(\Psi' \left(|\nabla u|^2 + |\nabla v|^2 \right) u_x \right) - 2\alpha \partial_y \left(\Psi' \left(|\nabla u|^2 + |\nabla v|^2 \right) u_y \right),$$

$$0 = 2H(v - v_{orig}) - 2\alpha \partial_x \left(\Psi' \left(|\nabla u|^2 + |\nabla v|^2 \right) v_x \right) - 2\alpha \partial_y \left(\Psi' \left(|\nabla u|^2 + |\nabla v|^2 \right) v_y \right),$$
(A.27)

which can be rewritten as:

$$0 = \frac{H}{\alpha} (u_{orig} - u) + \operatorname{div} \left(\Psi'(|\nabla u|^2 + |\nabla v|^2) \nabla u \right),$$

$$0 = \frac{H}{\alpha} (v_{orig} - v) + \operatorname{div} \left(\Psi'(|\nabla u|^2 + |\nabla v|^2) \nabla v \right).$$
(A.28)

By forming a diffusion-reaction system and discretizing the diffusion equations, unknowns $u_{i,j}^{k+1}$, $v_{i,j}^{k+1}$ can be explicitly computed (the approximation is implicit in bias/reaction term):

$$\partial_t u = \frac{H}{\alpha} (u_{orig} - u) + \operatorname{div} \left(\Psi'(|\nabla u|^2 + |\nabla v|^2) \nabla u \right),$$

$$\partial_t v = \frac{H}{\alpha} (v_{orig} - v) + \operatorname{div} \left(\Psi'(|\nabla u|^2 + |\nabla v|^2) \nabla v \right),$$
(A.29)

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\tau} = \frac{H}{\alpha} (u_{orig} - u_{i,j}^{k+1}) + \operatorname{div} \left(\Psi'(|\nabla u^{k}|^{2} + |\nabla v^{k}|^{2}) \nabla u^{k} \right),
\frac{v_{i,j}^{k+1} - v_{i,j}^{k}}{\tau} = \frac{H}{\alpha} (v_{orig} - v_{i,j}^{k+1}) + \operatorname{div} \left(\Psi'(|\nabla u^{k}|^{2} + |\nabla v^{k}|^{2}) \nabla v^{k} \right),$$
(A.30)

$$u_{i,j}^{k+1} = \frac{u_{i,j}^k + \tau \left(\operatorname{div}\left(\Psi'(|\nabla u^k|^2 + |\nabla v^k|^2)\nabla u^k\right) + \frac{H}{\alpha}u_{orig}\right)}{\left(1 + \frac{H}{\alpha}\tau\right)},$$

$$v_{i,j}^{k+1} = \frac{v_{i,j}^k + \tau \left(\operatorname{div}\left(\Psi'(|\nabla u^k|^2 + |\nabla v^k|^2)\nabla v^k\right) + \frac{H}{\alpha}v_{orig}\right)}{\left(1 + \frac{H}{\alpha}\tau\right)}.$$
(A.31)

A.10 Optic Flow

A minimizer of a 2D functional:

$$E(u,v) = \int F(x,y,u,v,u_x,u_y,v_x,v_y)dxdy$$
 (A.32)

necessarily satisfies the Euler-Lagrange equations:

$$0 = F_u - \partial_x F_{u_x} - \partial_y F_{u_y},$$

$$0 = F_v - \partial_x F_{v_x} - \partial_y F_{v_y}.$$
(A.33)

After calculation following equations are obtained:

$$0 = \operatorname{div} \left(\mathcal{V}'(\nabla u, \nabla v) \nabla u \right) - \frac{1}{2\alpha} \left(f_x(f_x u + f_y v + f_t) \right),$$

$$0 = \operatorname{div} \left(\mathcal{V}'(\nabla u, \nabla v) \nabla v \right) - \frac{1}{2\alpha} \left(f_y(f_x u + f_y v + f_t) \right).$$
(A.34)

Diffusion-reaction equations are then:

$$u_{t} = div \left(\mathcal{V}'(\nabla u, \nabla v) \nabla u \right) - \frac{1}{2\alpha} \left(f_{x}(f_{x}u + f_{y}v + f_{t}) \right),$$

$$v_{t} = div \left(\mathcal{V}'(\nabla u, \nabla v) \nabla v \right) - \frac{1}{2\alpha} \left(f_{y}(f_{x}u + f_{y}v + f_{t}) \right).$$
(A.35)

Again, depending on the form of the \mathcal{V}' function, discretization proceeds as previously, e.g. for homogeneous regularizer and penalizer $\mathcal{V}(|\nabla u|^2, |\nabla v|^2) =$

 $|\nabla u|^2+|\nabla v|^2$ and matching diffusivity $\mathcal{V}'(|\nabla u|^2,|\nabla v|^2)=1$:

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\tau} = \operatorname{div}(\nabla u^{k}) - \frac{1}{2\alpha} \left(f_{x}(f_{x}u^{k} + f_{y}v^{k} + f_{t}) \right),
\frac{v_{i,j}^{k+1} - v_{i,j}^{k}}{\tau} = \operatorname{div}(\nabla v^{k}) - \frac{1}{2\alpha} \left(f_{y}(f_{x}u^{k} + f_{y}v^{k} + f_{t}) \right).$$
(A.36)

Modified explicit scheme is then:

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\tau} = u_{xx}^{k} + u_{yy}^{k} - \frac{1}{2\alpha} \left(f_{x}(f_{x}u^{k+1} + f_{y}v^{k} + f_{t}) \right),
\frac{v_{i,j}^{k+1} - v_{i,j}^{k}}{\tau} = v_{xx}^{k} + v_{yy}^{k} - \frac{1}{2\alpha} \left(f_{y}(f_{x}u^{k} + f_{y}v^{k+1} + f_{t}) \right),$$
(A.37)

which is rewritten as:

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\tau} = \left(\frac{u_{i+1,j}^{k} - 2u_{i,j}^{k} + u_{i-1,j}^{k}}{h_{1}^{2}} + \frac{u_{i,j+1}^{k} - 2u_{i,j}^{k} + u_{i,j-1}^{k}}{h_{2}^{2}}\right) - \frac{1}{2\alpha} \left(f_{x}(f_{x}u^{k+1} + f_{y}v^{k} + f_{t})\right),$$

$$\frac{v_{i,j}^{k+1} - v_{i,j}^{k}}{\tau} = \left(\frac{v_{i+1,j}^{k} - 2v_{i,j}^{k} + v_{i-1,j}^{k}}{h_{1}^{2}} + \frac{v_{i,j+1}^{k} - 2v_{i,j}^{k} + v_{i,j-1}^{k}}{h_{2}^{2}}\right) - \frac{1}{2\alpha} \left(f_{y}(f_{x}u^{k} + f_{y}v^{k+1} + f_{t})\right).$$
(A.38)

Explicitly computing $u_{i,j}^{k+1}, v_{i,j}^{k+1}$ gives the following system of coupled diffusion reaction equations:

$$u_{i,j}^{k+1} = \frac{\text{stencil } u^k - \frac{1}{2\alpha} \left(f_x(f_y v^k + f_t) \right)}{1 + \frac{1}{2\alpha} f_x f_x},$$

$$v_{i,j}^{k+1} = \frac{\text{stencil } v^k - \frac{1}{2\alpha} \left(f_y(f_x u^k + f_t) \right)}{1 + \frac{1}{2\alpha} f_y f_y}.$$
(A.39)

where stencil is:

0	$\frac{ au}{h_2^2}$	0
$\frac{ au}{h_1^2}$	$1 - 2\frac{\tau}{h_1^2} - 2\frac{\tau}{h_2^2}$	$\frac{\tau}{h_1^2}$
0	$rac{ au}{h_2^2}$	0

A.11 Range Flow

Euler-Lagrange equations are:

$$0 = F_u - \partial_x F_{u_x} - \partial_y F_{u_y},$$

$$0 = F_v - \partial_x F_{v_x} - \partial_y F_{v_y},$$

$$0 = F_w - \partial_x F_{w_x} - \partial_y F_{w_y}.$$
(A.40)

After calculation the following equations are obtained:

$$0 = \operatorname{div} \left(\mathcal{V}'(\nabla u, \nabla v, \nabla w) \nabla u \right)$$

$$- \frac{1}{\alpha} \left(f_x(f_x u + f_y v + f_t) + \beta g_x(g_x u + g_y v + w + g_t) \right),$$

$$0 = \operatorname{div} \left(\mathcal{V}'(\nabla u, \nabla v, \nabla w) \nabla v \right)$$

$$- \frac{1}{\alpha} \left(f_y(f_x u + f_y v + f_t) + \beta g_y(g_x u + g_y v + w + g_t) \right),$$

$$0 = \operatorname{div} \left(\mathcal{V}'(\nabla u, \nabla v, \nabla w) \nabla w \right)$$

$$- \frac{1}{\alpha} \left(\beta (g_x u + g_y v + w + g_t) \right).$$

$$(A.41)$$

Diffusion-reaction equations are then:

$$u_{t} = \operatorname{div} \left(\mathcal{V}'(\nabla u, \nabla v, \nabla w) \nabla u \right)$$

$$- \frac{1}{\alpha} \left(f_{x}(f_{x}u + f_{y}v + f_{t}) + \beta g_{x}(g_{x}u + g_{y}v + w + g_{t}) \right),$$

$$v_{t} = \operatorname{div} \left(\mathcal{V}'(\nabla u, \nabla v, \nabla w) \nabla v \right)$$

$$- \frac{1}{\alpha} \left(f_{y}(f_{x}u + f_{y}v + f_{t}) + \beta g_{y}(g_{x}u + g_{y}v + w + g_{t}) \right),$$

$$w_{t} = \operatorname{div} \left(\mathcal{V}'(\nabla u, \nabla v, \nabla w) \nabla w \right)$$

$$- \frac{1}{\alpha} \left(\beta (g_{x}u + g_{y}v + w + g_{t}) \right).$$
(A.42)

Modified explicit scheme can be written as:

$$u^{k+1} = \frac{\text{stencil } u^k - \frac{\tau}{\alpha} \left(f_x f_y v^k + f_x f_t + \beta (g_x g_y v^k + g_x w^k + g_x g_t) \right)}{1 + \frac{\tau}{\alpha} \left(f_x f_x + \beta g_x g_x \right)},$$

$$v^{k+1} = \frac{\text{stencil } v^k - \frac{\tau}{\alpha} \left(f_y f_x u^k + f_y f_t + \beta (g_y g_x u^k + g_y w^k + g_y g_t) \right)}{1 + \frac{\tau}{\alpha} \left(f_y f_y + \beta g_y g_y \right)}, \quad (A.43)$$

$$w^{k+1} = \frac{\text{stencil } w^k - \frac{\tau}{\alpha} \left(\beta (g_x u^k + g_y v^k + g_t) \right)}{1 + \frac{\tau}{\alpha} \beta},$$

where stencil is one of the stencils from sections on linear isotropic, non-linear isotropic, and anisotropic diffusion (Section 3.2).

Appendix B

Ensemble Range Flow of the Fluid Data With Estimated Depth

Ensemble range flow is applied to the fluid data i.e. a video of an submerged oil leek. Used video sequence contains color channels, but offers no information about the depth in the scene. Depth information is necessary in order to estimate the range flow. Classical approaches for acquiring the depth information include using multiple cameras or other special equipment (e.g. Kinect), usually not used when recording scenes such as e.g. underwater spills. This problem is solved by estimating depth information from single images as proposed in [38], [40], [39].

The depth is estimated for the entire image, but the information about depth is required only in flow areas. In order to achieve this, depth gradients are set to zero when there is no change in spacial gradients between two consecutive frames.

B.0.1 Estimating Depth From Single Monocular Images

Approach from *Learning depth from single monocular images* [38] was used to estimate depth information necessary for range flow estimation.

Depth estimation from a single monocular image was done by utilizing a supervised learning approach. Training set of images was used together with their corresponding ground-truth depth maps. Note that the training was done on outdoor scenery images. Depth of an image was predicted as a function of the image. Discriminatively-trained Markov Random Field (MRF) was used with incorporated multiscale local and global features of the image. Depths at individual points and the relation between depths at different points are modeled.

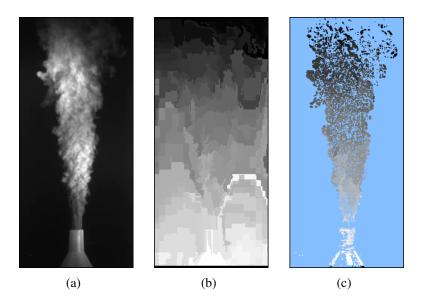


Figure B.1: (a) Frame 10 of the input image set of an opaque submerged buoyant jet. (b) Depth estimation of the frame 10 using the algorithm from [38]. (c) Depth gradients are set to zero (blue) when there is no change in spacial gradients between two consecutive frames.

Fig. B.1(b) shows the result of the algorithm. Since the information about depth is required only in flow areas, depth gradients are set to zero when there is no change in spacial gradients between two consecutive frames. Fig. B.1(c)

shows the depth image with irrelevant depth gradients set to zero. The algorithm is considering the closer objects to be darker.

Depth images were calculated by utilizing the MATLAB code ([22]) provided by the authors of [38]. Obtained depth images, together with the input video sequence, are used as an input for estimating the ensemble range flow of the data.

B.0.2 Results

The color coding used for the range flow estimation is depicted in Fig. 5.1(a). Color coding for the depth component of the range flow is interpolated between blue for backward and orange for forward movement.

500 frames of the opaque submerged buoyant jet are used for testing. The fluid is assumed to have an inverted-cone-like shape. Left column of Figure B.2 shows the result of the ensemble optic flow. Middle column of Figure B.2 shows the result of the uv-flow of the ensemble range flow. Right column of Figure B.2 shows the result of the depth w-flow of the ensemble range flow. Upper row shows only 4 iterations of the algorithm. Optic range flow (upper row, left) and uv-flow of the ensemble range flow (upper row, middle) already exhibit the predominant upward movement. w component of the ensemble range flow (upper row, right) still shows the mixed forward-backward movement. Considering the middle row of the image, uv-flow exhibits a predominant upward movement and depth w-flow predominant forward movement as expected. Inputing the reversed video sequence to the algorithm results in opposite movements (lower row).

To further improve such approaches, depth estimation algorithm from single images should be developed for and trained on underwater scenes, or scenes should be recorded with equipment capable of recording depth.

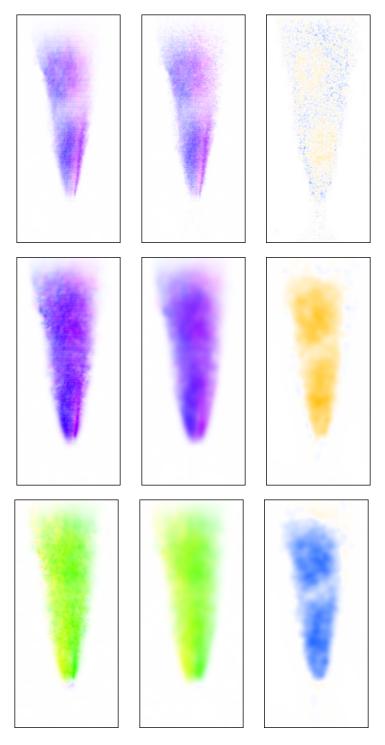


Figure B.2: 500 frames of the opaque submerged buoyant jet are used for testing. Upper row 4 iterations, middle and lower row 40 iterations. Upper and middle row are the result for the forward moving sequence, lower row is the result for the sequence played backward. Left column: Ensemble optic flow. Middle column: uv-flow from the ensemble range flow. Right column: Depth w-flow from the ensemble range flow.

Acknowledgement 135

Acknowledgement

Many people contributed in various ways to the successful completion of this thesis. I thank all of them.

I thank Jens-Malte Gottfried and his volunteers for recording gestures using the Kinect device, Frank Lenzen for advice on range flow, Oliver Stengele for helping with the implementation. I also thank my advisors for useful comments and suggestions.

I am especially grateful to my parents, Danijela and Geza, who have always supported and encouraged me.

Affidavit

Hereby I confirm that this thesis has been written only by the undersigned and without any assistance from third parties. Furthermore I confirm that no sources have been used in the preparation of this thesis other than those indicated in the thesis itself.

Dorotea Dudaš

Declaration of Consent

Herewith I agree that this thesis will be made available through the library of the Computer Science Department. This consent explicitly includes both the printed, as well as the electronic form. I confirm that the electronic and the printed version are of identical content.

Dorotea Dudaš

Affidavit 138

List of Figures

1.1	Introduction. Vortices	1
2.1	Analytical Vortex Examples. First order critical points in 2D linear	
	vector field	9
2.2	Analytical Vortex Examples. Fold bifurcation	11
2.3	Analytical Vortex Examples. 2D vector field from analytical function.	12
2.4	Analytical Vortex Examples. First order 3D critical points	13
2.5	Analytical Vortex Examples. Helical vector field	14
2.6	Analytical Vortex Examples. Circular helical vector field	15
2.7	Analytical Vortex Examples. Helical vortex with dislocated core	16
2.8	Real-Time Fluid Simulation. Advection of density through a vector	
	field	18
2.9	Real-Time Fluid Simulation. Hodge decomposition	20
2.10	Real-Time Fluid Simulation. Real-Time Fluid Simulation	20
2.11	Real-Time Fluid Simulation. Trilinear interpolation of a vector field	
	cell	21
2.12	Real-Time Fluid Simulation. CUDA grid	24
2.13	Recorded Data. Microsoft Kinect device and obtained range data	26
3.1	Diffusion. Diffusion of a 2D image	29
3.2	Vortex Preserving Diffusion. Color coded magnitude and arrow	
	plots of test vector fields	34
3.3	Vortex Preserving Diffusion. Critical points of test vector fields	35
3.4	Vortex Preserving Diffusion. Swirling areas of test vector fields	36
3.5	Vortex Preserving Diffusion. Nonlinear isotropic diffusion process	
	with diffusivity (3.27)	37

3.6	Vortex Preserving Diffusion. Resulting arrow plots	38
3.7	Vortex Preserving Diffusion. Non-swirling areas of test vector fields	
	together with swirling areas edge	38
3.8	Vortex Preserving Diffusion. Nonlinear isotropic diffusion process	
	with diffusivity (3.27) with additional blurring of the swirling	
	structure edges	39
3.9	Vortex Preserving Diffusion. Resulting arrow plots	40
3.10	Vortex Preserving Diffusion. Grayscale mapping of the discriminant	
	d of test vector fields	41
3.11	Vortex Preserving Diffusion. Nonlinear isotropic diffusion process	
	with diffusivity (3.28)	41
3.12	Vortex Preserving Diffusion. Resulting arrow plots	42
3.13	Vortex Preserving Diffusion. Critical points in the resulting flow fields.	43
3.14	Vortex Preserving Diffusion. Nonlinear isotropic diffusion process	
	with diffusivity (3.27)	44
3.15	Vortex Preserving Diffusion. Nonlinear isotropic diffusion process	
	with diffusivity (3.29)	44
3.16	Vortex Preserving Diffusion. Nonlinear isotropic diffusion process	
	with diffusivity (3.30)	45
3.17	Vortex Preserving Variational Processes. Minimization of the energy	
	functional (3.31)	47
3.18	Vortex Preserving Variational Processes. Minimization of the energy	
	functional (3.33)	48
3.19	Vortex Preserving Diffusion. Vortex preserving diffusion processes	
	based on discriminant d	50
3.20	Vortex Preserving Diffusion. Vortex preserving diffusion processes	
	based on vortex core/region detection	51
3.21	Vortex Preserving Variational Processes. Vortex preserving varia-	
	tional processes based on discriminant d	51
3.22	Vortex Preserving Variational Processes. Vortex preserving varia-	
	tional processes based on discriminant \boldsymbol{d} or vortex core detection	52
4 1	Vertex Extraction Methods Vertex region extraction	55
4.1	Vortex Extraction Methods. Vortex region extraction	\mathcal{I}

4.2	Vortex Extraction Methods. A curve of minimal bending energy	57
4.3	Vortex Extraction Methods. Interactive plane probe within a data set.	58
4.4	Vortex Extraction Methods. Minimal bending energy extraction of	
	the vortex core from a helical flow field	59
4.5	Vortex Extraction Methods. Optimal length of the vortex core can-	
	didate	59
4.6	Vortex Extraction Methods. Circular vortex core in a bent helical	
	flow field. Vortex rotating in a plane perpendicular neither to its	
	core, nor to the vorticity	60
4.7	Vortex Extraction Methods. Vortex core extraction from hurricane	
	Isabel data set, simulated cavity flow and flow behind a cylinder	60
4.8	Vortex Extraction Methods. Vortex core extraction with eigenvector	
	method	61
4.9	Vortex Extraction Methods. Extraction of the linear vortex core seg-	
	ment within a single tetrahedron using the eigenvector method	62
4.10	Vortex Extraction Methods. Reduced velocity is a projection of the	
	velocity to a plane perpendicular to the only real eigenvector	63
4.11	Stationary Flow PVO on the GPU. Dividing a cell into tetrahedrons	70
4.12	Stationary Flow PVO on the GPU. Close up view of the vortex cores	
	extracted from a helical flow field by using the PVO algorithm	72
4.13	Stationary Flow PVO on the GPU. Close up view of the vortex cores	
	extracted from a bent helical flow field by using the PVO algorithm.	73
4.14	Stationary Flow PVO on the GPU. Vortex core extraction from var-	
	ious flow fields using eigenvector method via PVO implemented	
	in OpenCL	74
4.15	Stationary Flow PVO on the GPU. Vortex cores extracted from	
	paused real-time simulated flow data using eigenvector method	
	via PVO	75
4.16	Stationary Flow PVO on the GPU. Vortex cores extracted using the	
	higher order method via PVO	76
4.17	Real-Time Flow Vortex Extraction. Interactive real-time flow simu-	
	lation	77

4.18	Real-Time Flow Vortex Extraction. Using a nonlinear isotropic dif-	
	fusion within the flow simulation	78
4.19	Real-Time Flow Vortex Extraction. Comparison of the PVO extrac-	
	tion from data produced within the real-time simulation and out-	
	side of it.	81
4.20	Real-Time Flow Vortex Extraction. Real-time simulation of vorticity	
	threshold vortex extraction	82
4.21	Real-Time Flow Vortex Extraction. Real-time simulation of Q crite-	
	rion vortex extraction	83
4.22	Real-Time Flow Vortex Extraction. Real-time simulation of λ_2 crite-	
	rion vortex extraction	84
4.23	Real-Time Flow Vortex Extraction. Real-time simulation of Q crite-	
	rion and λ_2 criterion vortex extraction	85
4.24	Real-Time Flow Vortex Extraction. Real-time simulation of the	
	eigenvector method via PVO vortex extraction	86
4.25	Real-Time Flow Vortex Extraction. Real-time simulation of the	
	eigenvector method via PVO vortex extraction (2)	86
4.26	Real-Time Flow Vortex Extraction. Real-time simulation of the	
	eigenvector method via PVO and Q criterion vortex extraction	87
4.27	Real-Time Flow Vortex Extraction. Real-time simulation of the	
	eigenvector method via PVO and λ_2 criterion vortex extraction	88
4.28	Real-Time Flow Vortex Extraction. Real-time simulation of the	
	coplanar vectors operator vortex extraction	88
4.29	Real-Time Flow Vortex Extraction. Real-time simulation of the par-	
	allel vectors operator and coplanar vectors operator vortex extrac-	
	tion	89
5.1	Range and Optic Flow. Input pair of color images for optic flow and	
5.1	resulting color coded magnitude of the estimated flow	94
5.2	Range and Optic Flow. Calibrated input pair of depth images ob-	ノヨ
J.4	tained by the Kinect device	94
5.3	Correction of Range Flow Computation. Magnitude of the resulting	ノヿ
5.5	range flow estimate with invalid areas included in input data	96
	range now estimate with invanta altas included in input data	フリ

5.4	Correction of Range Flow Computation. Thresholded derivative mag-
	nitudes of the depth channel
5.5	Correction of Range Flow Computation. Combined local-global
	range flow with invalid depth areas removed prior to estimation 97
5.6	Correction of Range Flow Computation. Combined local-global
	range flow of the office scene with invalid depth areas kept and
	removed prior to estimation
5.7	Correction of Range Flow Computation. Arrow plots of the resulting
	depth flow
5.8	Gesture Ensemble Range Flow Vortex Detection. Gestures 102
5.9	Gesture Ensemble Range Flow Vortex Detection. Optic flow between
	two gesture frames
5.10	Gesture Ensemble Range Flow Vortex Detection. Ensemble flow of
	the infinity gesture
5.11	Gesture Ensemble Range Flow Vortex Detection. Ensemble flows of
	the gestures
5.12	Gesture Ensemble Range Flow Vortex Detection. Detecting vortices
	of the ensemble range flow
5.13	Gesture Ensemble Range Flow Vortex Detection. Vortex detection of
	the ensemble range flows of the gestures
5.14	Gesture Ensemble Range Flow Vortex Detection. Filtering out the
	steep foci critical points from the result
5.15	Gesture Ensemble Range Flow Vortex Detection. Filtering out the
	steep foci critical points from the result (2)
5.16	Gesture Ensemble Range Flow Vortex Detection. Summing up of the
	magnitude of the flow through every x and y coordinate produces
	a histogram-like graph
5.17	Gesture Ensemble Range Flow Vortex Detection. Graphs (and PCA
	reconstructions) for the circle and infinity gestures
5.18	Gesture Ensemble Range Flow Vortex Detection. Vortex preserving
	diffusion of the ensemble range flow

List Of Figures 144

6.1	Future Work. Fluid-object interaction. Butterfly flapping its wings
	within a real-time fluid simulation
B.1	Ensemble Range Flow of the Fluid Data With Estimated Depth. Esti-
	mating depth from single images of a submerged buoyant jet 132
B.2	Ensemble Range Flow of the Fluid Data With Estimated Depth. Result-
	ing ensemble optic flow of a opaque submerged buoyant jet 134

List of Tables

4.1	Stationary Flow PVO on the GPU. Choice of the underlaying struc-	
	ture of the cell used for calculation results in different number of	
	triangles to be processed by the algorithm	71
4.2	Stationary Flow PVO on the GPU. Vortex core extraction from a heli-	
	cal flow field via eigenvector method (PVO) using different num-	
	bers of triangles per cells	72
4.3	Stationary Flow PVO on the GPU. Vortex core extraction from a bent	
	helical flow field via eigenvector method (PVO) using different	
	numbers of triangles per cells	73

List Of Tables 146

- [1] J. Batlle, E. Mouaddib, and J. Salvi. Recent progress in coded structured light as a technique to solve the correspondence problem: a survey. *Pat. Recog.*, 31(7):963–982, 1998.
- [2] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *Int. J. of Comp. Vis.*, 61:211–231, 2005.
- [3] Guoning Chen, Konstantin Mischaikow, Robert S. Laramee, Pawel Pilarczyk, and Eugene Zhang. Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):769–785, July 2007.
- [4] Chris Tangey, stilltalkincrazy. *http://www.youtube.com/*.
- [5] J.P. Den Hartog. Strength of Materials. Dover, New York, 1949.
- [6] V. Dousset and A. Pothérat. Formation mechanism of hairpin vortices in the wake of a truncated cylinder in a duct. *J. Fluid Mech.*, 653:519–536, 2010.
- [7] D. Dudas. Vortex core extraction of 3d vector fields based on minimal bending energy. *Masters Thesis, University of Saarland, Max Planck Institute for Computer Science*, 2008.
- [8] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of 3d vector fields. *Proceedings of IEEE Visualization '91*, pages 33–39, October 1991.
- [9] J.-M. Gottfried, J. Fehr, and C. S. Garbe. Computing range flow from multi-modal Kinect data. *7th International Symposium on Visual Computing (ISVC 2011)*, 2011.
- [10] S. Hadfield and R. Bowden. Kinecting the dots: Particle based scene flow from depth sensors. In *ICCV* 2011, 2011.

[11] G. Haller. An objective definition of a vortex. *J. Fluid Mech.*, 525:1–26, 2005.

- [12] B. K. P. Horn and B. G. Schunck. Determining optical flow. *AI*, 17:185–203, 1981
- [13] B.K.P. Horn. The curve of least energy. *ACM Transactions on Mathematical Software*, 9(4):441–460, December 1983.
- [14] J. C. R. Hunt, A. Wray, and P. Moin. Eddies, stream, and convergence zones in turbulent flows. *Center for Turbulence Research Report CTR-S88*, 1988.
- [15] T. Iijima. Basic theory on normalization of pattern (in case of typical one-dimensional pattern) (in japanese). *Bulletin of the Electrotechnical Laboratory*, 26:368–388, 1962.
- [16] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 559–568, 2011.
- [17] J. Jeong and F. Hussain. On the identification of a vortex. *Journal of Fluid Mechanics*, (285):69–94, 1995.
- [18] Eva Kollorz, Jochen Penne, Joachim Hornegger, and Alexander Barke. Gesture recognition with a time of flight camera. *Int. J. Intell. Syst. Technol. Appl.*, 5(3/4):334–343, November 2008.
- [19] Leap Motion. http://leapmotion.com/.
- [20] Antoine Letouzey, Benjamin Petit, and Edmond Boyer. Surface Flow from Depth and Color Images. In *British Machine Vision Conference*, Dundee, Royaume-Uni, August 2011.
- [21] Mathematica. Wolfram Research Inc., Champaign, Illinois, 2010.
- [22] MATLAB. The MathWorks, Inc., Natic, Massachusetts, 2010.
- [23] Mercedes-Benz Museum, Stuttgart, Germany. http://www.mercedes-benz-classic.com.

- [24] Microsoft Kinect. http://www.xbox.com/en-GB/KINECT.
- [25] Mitchell County Press News. http://globegazette.com/mcpress/.
- [26] H. P. Moreton. *Minimum Curvature Variation Curves, Networks, and Surfaces for Fair Free-Form Shape Design*. PhD thesis, Uinversity of California at Berkeley, 1992.
- [27] Renata Nascimento, Joo Paixo, Hlio Lopes, and Thomas Lewiner. Topology aware vector field denoising. In *Sibgrapi 2010 (XXIII Conference on Graphics, Patterns and Images)*, pages 103–109, August 2010.
- [28] OpenCL 1.2 Reference Pages. http://www.khronos.org/registry/cl/sdk/1.2/docs/man/xhtml/.
- [29] ParaView Open Source Scientific Visualization. http://www.paraview.org/.
- [30] R. Peikert and M. Roth. A higher-order method for finding vortex core lines. *Proceedings of IEEE Visualization '98*, pages 143–150, 1998.
- [31] R. Peikert and M. Roth. The "parallel vectors" operator a vector field visualization primitive. *Proceedings of IEEE Visualization 2000*, pages 263–270, 2000.
- [32] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.
- [33] A.E. Perry and H. Hornung. Some aspects of three-dimensional separation, part 2: Vortex skeletons. *Z.Flugwiss, Weltraumforsh.* 8, *Heft* 3, pages 155–160, 1984.
- [34] C. Rabe, T. Müller, A. Wedel, and U. Franke. Dense, robust, and accurate motion field estimation from stereo image sequences in real-time. In *ECCV* 2010, pages 582–595, 2010.
- [35] M. Raffel, C. Willert, S. Wereley, and J. Kompenhans. Particle image velocimetry: A practical guide. Springer Verlag 2nd Edition, 2007.
- [36] Jan Reininghaus, Christian Lowen, and Ingrid Hotz. Fast combinatorial vector field topology. *IEEE Transactions on Visualization and Computer Graphics*, 17:1433–1443, 2011.

[37] M. Roth and R. Peikert. Flow visualization for turbomachinery design. *Proc. IEEE Visualization*, pages 381–384, 1996.

- [38] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. *Neural Information Processing Systems (NIPS)* 18, 2005.
- [39] A. Saxena, S. H. Chung, and A. Y. Ng. 3-D depth reconstruction from a single still image. *International Journal of Computer Vision (IJCV)*, 2007.
- [40] A. Saxena, M. Sun, and A. Y. Ng. Learning 3-D scene structure from a single still image. *ICCV workshop on 3D Representation for Recognition* (3dRR-07), 2007.
- [41] Hanno Scharr and Joachim Weickert. An anisotropic diffusion algorithm with optimized rotation invariance. In *In Mustererkennung 2000, Proceedings from 22. DAGM-symposium*, pages 460–467. Springer, 2000.
- [42] C. Schnörr. Segmentation of visual motion by minimizing convex non-quadratic functionals. In *12th Int. Conf. on Patt. Rec.*, Israel, 1994.
- [43] T. Schuchert, T. Aach, and H. Scharr. Range flow in varying illumination: Algorithms and comparisons. *Patt. Analysis and Machine Intelligence*, 32(9):1646–1658, 2010.
- [44] H. Spies, B. Jähne, and J. Barron. Range flow estimation. *Comp. Vis. and Image Understanding*, 85(3):209–231, 2002.
- [45] Karsten Staack and Christoph Sebastian Garbe. Ensemble Optical Flow. In *Flucome 2011, Keelung, Taiwan*, pages 1–9, August 2011.
- [46] Jos Stam. Stable fluids. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 121–128, August 1999.
- [47] D. Sujudi and R. Haimes. Identification of swirling flow in 3d vector fields. *12th AIAA CFD Conference*, pages AIAA Paper 95–1715, June 1995.
- [48] The Visualization ToolKit. http://public.kitware.com/VTK/.
- [49] H. Theisel and H.-P. Seidel. Feature flow fields. *Proc. Fifth Joint IEEE VGTC Eurographics Symp. Visualization (VisSym)*, 2003.
- [50] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *CV 1999*, pages 722 –729 vol.2, 1999.

[51] P. Vuylsteke and A. Oosterlinck. Range image acquisition with a single binary-encoded light pattern. *IEEE Trans. Pat. Anal. Mach. Int.*, 12(2):148–164, 1990.

- [52] J. Weickert. *Anisotropic Diffusion in Image Processing*. PhD thesis, Teubner, Stuttgart, 1998.
- [53] Joachim Weickert. Coherence-enhancing diffusion filtering. *Int. J. Comp. Vis.*, 31:111–127, 1999.
- [54] Joachim Weickert and Christoph Schnörr. A theoretical framework for convex regularizers in pde-based computation of image motion. *Int. J. Com. Vis.*, 45:245–264, 2001.
- [55] T. Weinkauf, J. Sahner, H. Theisel, and H.-C. Hege. Cores of swirling particle motion in unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1759–1766, 2007.
- [56] Andrew P. Witkin. Scale-space filtering. In *Proceedings of the Eighth international joint conference on Artificial intelligence Volume 2*, IJCAI'83, pages 1019–1022, 1983.