Institut für Informatik
Technische Universität München

Real-Time Approaches for Model-Based Reconstruction and Visualization of Flow Fields

Polina Kondratieva

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende:	Univ.-Prof. G. J. Klinker, Ph. D.
Prüfer der Dissertation:	1. Univ.-Prof. Dr. R. Westermann
	2. Univ.-Prof. Dr. Th. Ertl, Universität Stuttgart

Die Dissertation wurde am 30.04.2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 24.10.2008 angenommen.

To my family, my friends, and my Muse

Abstract

The understanding of flow phenomena is of great interest in different areas of natural sciences. In this thesis, the following aspects related to the quantitative and qualitative flow exploration are investigated. Firstly, a novel reconstruction method has been developed, which employs a physics-based filter operation to generate a velocity field that is consistent with the underlying flow model described by the Navier-Stokes equations. Interactive steering of the reconstruction process is achieved by exploiting programmable graphics hardware as a co-processor for numerical computations and the interactive visualization of 2D flow fields. The quality of the proposed method has been estimated via the reconstruction of velocity fields from synthetic and real-world image sequences of laminar flow.

Secondly, a new feature-based importance-driven technique for visual exploration of complex flows in 3D space, so called anchor lines, has been introduced. Anchor lines are integral curves started in regions of high importance and accompanied by the particles seeded in the close vicinity of these regions. Experimentally it has been proven that seeding anchor lines based on the finite-time Lyapunov exponent - a scalar quantity that characterizes the rate of separation of infinitesimally close particles - allows for significant reduction of the amount of conveyed information, yet emphasizing the important flow structures. The efficiency of anchor lines is demonstrated by exploring several complex flow scenarios in 3D space.

Thirdly, motivated by the importance of tensor quantities in physics and particularly in fluid mechanics, a new method for interactive and intuitive tensor field exploration based on the GPU particle tracing approach has been developed. In this case, the vector field is derived from the diffusion tensor field by solving the associated eigenvalue problem. The benefits of using this approach for the visualization of diffusion tensor fields are shown in investigation of diffusion properties of biological tissue.

Zusammenfassung

Das Verständnis von Strömungsphänomenen ist in verschiedenen Bereichen der Naturwissenschaften von besonderem Interesse. Die folgenden Aspekte der qualitativen und quantitativen Strömungsuntersuchung werden in dieser Dissertation untersucht. Erstens wurde ein neues Rekonstruktionsverfahren entwickelt. Diese Methode mittels einer physikbasierten Filteroperation ein Geschwindigkeitsfeld erzeugt, das mit dem zugrunde liegenden Modell konsistent ist, welches durch die Navier-Stokes-Gleichungen beschriebenen wird. Die interaktive Steuerung des Rekonstruktionsprozesses wird durch den Einsatz programmierbarer Graphikhardware als Coprozessor für numerische Berechnungen und die Visualisierung von 2D Strömungsfeldern erreicht. Die Qualität der vorgeschlagenen Methode wird durch die Rekonstruierung von Geschwindigkeitsfeldern aus synthetischen und experimentellen Bildsequenzen von Laminarströmungen bestimmt.

Zweitens wurde eine effiziente merkmalsbasierte und wichtigkeitsgesteuerte Technik zur visuellen Erforschung von komplexen Strömungen im dreidimensionalen Raum, so genannte Ankerlinien, vorgestellt. Ankerlinien sind Integralkurven, die in Bereichen hoher Wichtigkeit gestartet werden und die von den Partikeln begleitet werden, die in der unmittelbaren Umgebung dieser Bereiche eingestreut werden. Es wurde experimentell geprüft, dass das Einbringen der Ankerlinien ausgehend vom zeitbegrenzten Lyapunov-Exponenten - ein skalar Wert, der das Mass der Separation infinitesimal naher Partikel charakterisiert - eine signifikante Reduktion der Menge der dargestellten Informationen ermöglicht und dennoch die wichtigen Strömungsstrukturen hervorhebt. Die Effizienz von Ankerlinien wird anhand der Untersuchung einiger komplexer Strömungsszenarien im 3D demonstriert.

Drittens wurde, motiviert durch die Wichtigkeit von Tensorgrössen in der Physik, und im Besonderen in der Fluidmechanik, eine neue Methode zur interaktiven und intuitiven Untersuchung von Tensorfeldern basierend auf GPU-Partikelverfolgungsverfahren entwickelt. In diesem Fall wird das Vektorfeld vom Diffusions-Tensorfeld abgeleitet, in dem das dazugehörige Eigenwertproblem gelöst wird. Die Leistungsfähigkeit dieser Technik zur Visualisierung von Diffusions-Tensorfeldern werden durch die Untersuchung von Diffusionseigenschaften von biologischem Gewebe gezeigt.

Acknowledgements

This dissertation has a long story and many people have participated in its creation in different ways. I would like to thank them all here, in this little chapter, though it does not show all my gratitude to them.

First of all, I would like to thank my parents, who – being scientists – taught me how to bring research into everyday's life and how to not give into a problem, until you get to its core. In particular, this was my parents' idea to talk me into taking part in a contest. After I won it, they talked me into accepting the offer to continue my studies in Germany, which was the prize. Additionally, I would like to thank the other members of my family, especially my brother Andrey, my friends Julia and Nadja, my teachers, and my study colleagues from group IUS-61 for believing in me and morally supporting me, despite the thousands of miles which separated us.

This thesis would never have been even started, if Prof. Dr. Paul from the University of Saarland had not granted me the first prize in a contest and had not brought me among other students from Khabarovsk State University to Saarbrücken for further studies. That is why I am infinitely grateful to Prof. Dr. Paul for providing me this opportunity.

During my staying in Saarbrücken, I had a lot of difficulties in both private life and studies. The foreign country, the foreign language, many factors influenced it. But I had good friends, Igorj and Yury, and very kind supervisors, Prof. Dr. Kautz, Prof. Dr. Myszkowski, Prof. Dr. Havran, and Prof. Dr. Belyaev. They helped me through the problems and they indirectly made it possible for me to stay in Germany. Since I am a foreign student, everything is connected to the visa, and the visa in turn, is directly connected to the job contract. When my scholarship expired, I had a risk to lose my right to stay since I did not have a job. Fortunately, thanks to Prof. Dr. Kautz and Prof. Dr. Havran, I have become a student assistant, and thus got a possibility to finish my studies. Especially, I am grateful to Prof. Dr. Myszkowski and Prof. Dr. Belyaev, who always encouraged me. I will never forget their words, which they told me in my minutes of desperation: "Polina, fight for yourself, and everything will be fine".

After I finally got my Master of Computer Science certificate, I decided to dedicate myself to research in the area of computer graphics and visualization. I was fortunate enough to get the offers for a PhD position in two universities simultaneously. I accepted the one from TU München and I do not regret it. I am very thankful to Prof.

Dr. Westermann that he offered me this position. I am very grateful for his advice and moral support, especially in the moments of desperation, when something did not work out as I thought it would.

Furthermore, I very much appreciate the help of *all* my colleagues: Kai Bürger, Christian Dick, Roland Fraedrich, Joachim Georgii, Stefan Hertel, Martin Kraus, Jens Krüger, Thomas Schiwietz, and Jens Schneider, for fruitful discussions, and for proof-reading of this thesis. Thanks to them, during my four years of work in the Chair I learned all the benefits of a good team and close collaboration within the team. Especially, I am thankful to some of my colleagues, who have become my real friends: Christian, Jens (Schneider), Joachim, and Stefan. They helped me through the storms in my private life. I am not sure, what I would do if I had not had their support.

Besides my direct colleagues, I am grateful to Dr. Petermeier, Dr. Kowalczyk, Dr. Zima, Frank Holtmann, Prof. Dr. Denz, and Prof. Dr. Delgado, our collaboration partners within the DFG research program SPP-1147. Without their professional explanations and fruitful discussions about the specifics of the micro-flow scenarios, I would have to spend years in searching for the answers in books and papers. The financial support by the Deutsche Forschungsgemeinschaft (DFG) is highly acknowledged.

Additionally, I would like to thank two families, which showed me, what a good German family really is: the family of Jens (Schneider) and the family of Joachim. They always were very kind and cared about my well-being. It is hard to understand what that means if you have never been far from your home for many years.

Also, I would like to thank the agencies for foreign affairs in Saarbrücken and München for extending my visa. Especially, I am thankful for the last prolongation, which gave me the possibility to finish my PhD thesis and search for job, without even having a hint to find one in a short period of time. In addition, I am very grateful to my new employer, Realtime Technology AG, for hiring me.

For breaking the silence in my room, filling the air with the beautiful sound, letting metal and Russian rock flow through my veins, and lifting my spirit by this powerful performance, I am grateful to my favorite bands: Metallica, Blind Guardian, Darkseed, Megaherz, Aria, Alisa, DDT, and Nautilus Pompilius.

Finally, I would like to thank my *Muse* for giving me the inspiration, strength of spirit, and believe into myself. Without this powerful support, I am not sure that I would have done even a half of what I have done. There are no words which could express my gratitude to this person.

Thank you all!

Contents

List of Figures

List of Tables

Abbreviations

The abbreviations used in this dissertation are listed in alphabetical order in the table below. All of these abbreviation are also explained in the text at their first occurrence. The abbreviation itself is placed in the first column, and the corresponding to it full phrase is given in the second column. The figure (Fig.), table (Tab.), etc., or/and the page, where the it appears for the first time, is referenced in the third and fourth columns, respectively.

Abbreviation	**Full phrase**	**Page**	**Place**
AMA	Acoustic Microphone Array	19	Tab. 2.1
BOS	Background Oriented Schlieren	19	Tab. 2.1
CC	Cross–Correlation	26	
CCD	Charge–Coupled Device	21	
CC–NS	Cross–Correlation and Navier Stokes	61	
CG	Conjugate Gradient	52	
CMOS	Complementary Metal-Oxide-Semiconductor	22	
DPIV	Digital Particle Image Velocimetry	28	
DT	Diffusion Tensor	126	
DTI	Diffusion Tensor Imaging	126	
DT–MRI	Diffusion Tensor Magnetic Resonance Imaging	126	
FFT	Fast Fourier Transform	28	
FTLE	Finite–Time Lyapunov Exponent	106	
GPU	Graphics Processing Units	47	
HLSL	High Level Shading Language	48	
IBFV	Image–Based Flow Visualization	88	
IPCT	Image Pattern Correlation Technique	19	Tab. 2.1
IR	Image Registration	26	
ISA	Image Space Advection	88	

Abbreviation	Full phrase	Page	Place
LIC	Line Integral Convolution	88	
LSP	Laser Speckle Photography	21	
LSV	Laser Speckle Velocimetry	20	Fig. 2.4
MRI	Magnetic Resonance Imaging	126	
MRT	Multiple Render Target Technique	92	
MSE	Mean–Square Error	61	Tab. 4.1
NSE	Navier–Stokes Equations	40	
OF	Optical Flow	26	
OF–NS	Optical Flow and Navier Stokes	61	
PIV	Particle Image Velocimetry	20	Fig. 2.4
PLV	Pulsed–Light Velocimetry	19	Tab. 2.1
PSNR	Signal–to–Noise Ratio	61	Tab. 4.1
PSP	Pressure Sensitive Paint	19	Tab. 2.1
PT	Particle Tracking	26	
PTV	Particle Tracking Velocimetry	20	Fig. 2.4
RMS	Root–Mean–Square	61	Tab. 4.1
SNR	Signal–to–Noise Ratio	61	Tab. 4.1
TSP	Temperature Sensitive Paint	19	Tab. 2.1
VF	Vector Field	25	

Notation

All symbols are defined and explained in the text at their first occurrence. The appearance of all symbols is defined according to the following rules:

$\mathbf{v}$	bold symbols are used for vectors,
$\boldsymbol{f}$	bold italic symbols are used for names of continuous functions,
var	italic symbols are used for scalar variables,
$\texttt{VF}$	typewriter-style symbols are used for discrete vector fields, textures, and names of pseudo-code functions,
$\mathbb{D}$	double-contoured symbols are used for matrices and tensors,
$\mathcal{C}$	calligraphic symbols for scalar fields,
$\boldsymbol{\mathcal{F}}$	calligraphic bold symbols for continuous vector fields and flow maps,
$\tilde{\varepsilon}$	symbols with tilde above them denote averaged value,
$\hat{\mathbf{e}}$	bold symbols with hat above them denote normalized vectors,
$\mathfrak{R}$	Gothic symbols are used for spaces (e.g., 3D space is denoted as $\mathfrak{R}^3$).

For convenience, frequently used variables are listed below in a table format. The symbol itself is depicted in the first column, and its short description is given in the second column. The equation (Eq.), figure (Fig.), table (Tab.), etc., or/and the page, where the it appears for the first time, is referenced in the third and fourth columns, respectively.

Symbol	Description	Page	Place
Vector field definition			
$\boldsymbol{f}(x, y, z)$	analytical vector field in 3D	13	
$\mathbf{\Omega}$	spatial domain	13	
$\mathfrak{R}^n$	n-D space	13	
$\mathfrak{R}^3$	3D space	13	
$\mathbf{p}$, (x, y, z)	a point in 3D space	13	

Symbol	**Description**	**Page**	**Place**
Φ	vector field value	13	
$\boldsymbol{\mathcal{F}}$	flow map (general)	13	
$\mathbf{V}$	velocity vector	14	Eq. 2.1
$\mathbf{V}(\mathbf{p})$	vector field over the spatial domain	14	Eq. 2.1
t	time variable	14	Eq. 2.1
Π	temporal domain	15	Eq. 2.2
$\mathbf{V}\big(\mathbf{p}(t),\, t\big)$	vector field over the spatial and temporal domains	15	Eq. 2.2
$\boldsymbol{\phi}(\mathbf{p},\, t)$	flow map in vector calculus	16	Eq. 2.4
$\boldsymbol{\Theta}$	phase space of a dynamical system	16	
$\mathbf{M}(t)$	state of a dynamical system at time t	16	Eq. 2.5
Γ^t	evolution function of a dynamical system	16	
$\boldsymbol{\gamma}(\mathbf{M}(t),\, t)$	velocity vector of a dynamical system	16	Eq. 2.5
Flow measurements			
ρ	density of fluid flow	17	
μ	dynamic viscosity of fluid flow	17	
Re	Reynolds number	17	
p	pressure	19	
$p(t)$	dynamic pressure on the surface	19	
H	heat transfer	19	
T	temperature	19	
$\nabla\rho$	density gradient	19	
$\delta\mathbf{x}$	deformation of the model	19	
N_s	source density	20	
N_I	image density	20	
Image registration			
$\mathtt{T}$	template image	26	Eq. 2.6
$\mathtt{R}$	reference image	26	Eq. 2.6
$\boldsymbol{Dist}$	distance measure	26	Eq. 2.6
$\boldsymbol{\varphi}(\mathbf{x})$	spatial transformation	26	Eq. 2.6
$\boldsymbol{Sm}$	regularization term (smoother)	27	Eq. 2.7
$\boldsymbol{\tau}(\mathbf{d})$	distance measure with smoother	27	Eq. 2.7
$\mathbf{d}$	displacement	27	Eq. 2.7
α	amount of smoothing	27	Eq. 2.7

Symbol	Description	Page	Place
Reconstruction algorithms			
$\boldsymbol{\sigma}(x,\, y)$	signal in the 2D domain	27	Eq. 2.8
$\boldsymbol{\sigma}(x,\, y)$	signal in the 2D domain	27	Eq. 2.8
$\mathtt{CC}_{\sigma_0,\sigma_1}[x, y]$	discrete cross-correlation function	27	Eq. 2.8
(u, v)	components of 2D velocity vector	31	Eq. 2.10
E	image brightness	31	Eq. 2.10
$E_x,\ E_y$	spatial derivatives of image brightness	31	Eq. 2.10
E_t	temporal derivative of image brightness	31	Eq. 2.10
ζ_b	general minimization function	31	Eq. 2.10
ζ_c	regularization term (smoother) for optical flow	32	Eq. 2.11
ζ	minimization function with regularization term	32	Eq. 2.12
$\mathtt{VF}_\mathtt{p}$	vector field computed by predictor	42	Fig. 3.2
$\mathtt{VF}_\mathtt{c}$	vector field after the correction step	42	Fig. 3.2
$\mathtt{D(T)}$	deformed template image	42	Fig. 3.2
$\mathtt{Im}_\mathtt{def}$	deformed template image (as a set of pixels)	42	Fig. 3.2
$\mathtt{VF}$	final vector field after several iteration of predictor-corrector scheme	42	Fig. 3.2
it	iteration number	42	Fig. 3.2
∇	gradient operator	44	Eq. 3.5
∇^2 or Δ	Laplace operator	44	Eq. 3.5
$\mathbf{G}$	external force vector	44	
(g_x, g_y)	components of external force vector	44	Eq. 3.1-3.2
p	pressure	44	Eq. 3.1-3.2
Re	Reynolds number	44	Eq. 3.4
ρ	density of the fluid	44	Eq. 3.4
μ	dynamic viscosity of the fluid	44	Eq. 3.4
U_∞	characteristic velocity	44	Eq. 3.4
L	characteristic length	44	Eq. 3.4
$\delta(\mathbf{p} - \mathbf{p_0})$	Dirac delta function shifted by $\mathbf{p_0}$	45	Eq. 3.8
h	position of a stokeslet above the planar boundary	46	
$\boldsymbol{\psi}(r,\, z)$	Green's function in cylindrical polar coordinate system	46	Eq. 3.9

Symbol	Description	Page	Place
Reconstruction algorithms			
$\mathtt{UV}$	combined column of velocity vector components $(u_{00}, ..., u_{mn}, \|v_{00}, ..., v_{mn})$	49	
$\mathtt{K}^{\mathtt{h}}$	optical flow system matrix on the fine grid	50	Fig. 3.8
$\mathtt{K}^{\mathtt{H}}$	optical flow system matrix on the coarse grid	50	Fig. 3.8
$\mathtt{e}^{\mathtt{h}}$	absolute errors between the exact solution and the approximate solution on the fine grid	50 50	Fig. 3.8 Fig. 3.8
$\mathtt{e}^{\mathtt{H}}$	absolute errors between the exact solution and the approximate solution on coarse grid	50 50	Fig. 3.8 Fig. 3.8
$\mathtt{r}^{\mathtt{h}}$	residual on the fine grid	50	Fig. 3.8
$\mathtt{r}^{\mathtt{H}}$	residual on the coarse grid	50	Fig. 3.8
$\mathtt{I}_{\mathtt{h}}^{\mathtt{H}}$, $\mathtt{I}_{\mathtt{H}}^{\mathtt{h}}$	linear transfer operators	50	Fig. 3.8
$\mathtt{U}_{\mathtt{NSE}}$	vector field of dimensionless velocity used in the Navier-Stokes equations	53	
$\mathbf{U}_{\mathbf{NSE}}$	a single vector from $\mathtt{U}_{\mathtt{NSE}}$	53	Eq. 3.16
$\mathtt{disp}_{\mathtt{OF}}$	optical flow solution (displacement field)	53	
$\mathbf{disp}_{\mathbf{OF}}$	a single vector from $\mathtt{disp}_{\mathtt{OF}}$	53	Eq. 3.16
$\mathbf{F}_{\mathbf{Stokes}}$	Stokes force	53	Eq. 3.16
$\mathbf{F}_{\mathbf{Stokes}_{\mathbf{NSE}}}$	dimensionless Stokes force	53	Eq. 3.16
$\mathtt{Im}$	image as a set of pixels	55	Fig. 3.11
$\varepsilon_{it}\left(\mathtt{VF}_{\mathtt{p}}{}^{(it)}\right)$	iteration-dependent error metric	57	Eq. 3.12
Validation of the reconstruction method			
L_1	norm for nD vector: $\|\|\mathbf{x}\|\|_1 = \|x_1\| + \ldots + \|x_n\|$	60	
L_2	norm for nD vector: $\|\|\mathbf{x}\|\|_2 = \sqrt{\|x_1\|^2 + \ldots + \|x_n\|^2}$	60	
$\mathtt{V}_{\mathtt{ref}}$	ground truth velocity vector field	61	Tab. 4.1
$\varepsilon_{\mathrm{ang}}$	per-pixel map of angular error	61	Tab. 4.1
$\varepsilon_{\mathrm{len}}$	per-pixel map of length error	61	Tab. 4.1
$\tilde{\varepsilon}_{\mathrm{ang}}$	average angular error	61	Tab. 4.1
$\tilde{\varepsilon}_{\mathrm{len}}$	average length error	61	Tab. 4.1
SNR	signal-to-noise ratio	61	Tab. 4.1
$\mathrm{RMS}_{\mathrm{ref}}$	root-mean-square error for the ground truth vector field	61	Tab. 4.1

Symbol	**Description**	**Page**	**Place**
Validation of the reconstruction method			
$\mathrm{RMS}_{\mathrm{dif}}$	root-mean-square error between the ground truth vector field and the computed vector field	61	Tab. 4.1
MSE	mean-square error	61	Tab. 4.1
PSNR	peak signal-to-noise ratio	61	Tab. 4.1
Flow visualization			
$\mathbf{p}_{\text{path}}(t;\mathbf{p},t_0)$	path line	89	Eq. 5.1
$\mathbf{p}_{\text{stream}}(t;\mathbf{p},t_0)$	stream line	89	Eq. 5.1
$\mathbf{p}_{\text{streak}}(t;\mathbf{p},t_0)$	streak line	89	Eq. 5.1
$\mathbb{P}[i,j]$	particle with index $[i,j]$	92	
`VFTex3D`	3D texture to store a discrete vector field	92	
$\mathbf{p}$	particle position of position on aline	89	
`StartPosTex`	starting position texture	93	
`PosTex`	position texture	93	
λ_2	criterion for vortex detection	91	
$\mathbb{M}$	transformation matrix for the user-defined seeding probe	93	
$\texttt{tc}[x,y]$	texture coordinates assigned to the point sprite by the graphics processing unit	95	
$\mathbb{M}_{\text{dir}}$	transformation of texture coordinates $\texttt{tc}[x,y]$	95	
$\mathbb{M}_{\text{rot}}$	rotation of texture coordinates $\texttt{tc}[x,y]$	95	Eq. 5.3
$\mathbb{M}_{\text{trans}}$	translation of texture coordinates $\texttt{tc}[x,y]$	95	Eq. 5.3
$\texttt{LPosTex}[t]$	texture for currently computed positions of line vertices	96	
`FullLPosTex`	block-texture for all positions of all lines	96	
$\texttt{tc}_{\text{up}}[t]$	texture coordinates of the upper left corner of the t^{th} block within `FullLPosTex`	97	Fig. 5.5
$\texttt{tc}_{\texttt{shift}}[i,j]$	shift in texture coordinates within the block corresponding to the line with index $[i,j]$	97	Fig. 5.5
Feature-based flow visualization			
$\boldsymbol{Imp}(\mathbf{p})$	spatial importance function	100	Eq. 5.4

Symbol	Description	Page	Place
Feature-based flow visualization			
$\boldsymbol{Imp}(\mathbf{p})$	spatial importance function	100	Eq. 5.4
$\boldsymbol{att}(\mathbf{p}, \mathbf{p_f})$	spatial attenuation function	100	Eq. 5.4
$\mathbf{p_f}$	user-specified focus point	100	Eq. 5.4
`impVol`	scalar volume of importance measure	100	Eq. 5.4
$\boldsymbol{\omega}$	flow circulation (vorticity)	102	
ω	vorticity magnitude	102	
h	helicity density	102	
w_s	streamwise vorticity	102	
$\boldsymbol{\sigma}_{t_0}^{\Delta t}(\mathbf{p})$	maximum Finite–Time Lyapunov Exponent	102	
		109	Eq. 6.9
$\widetilde{\mathbf{V}}$	average velocity	103	
ε_{ang}	angular difference between the velocity $\mathbf{V}$ and the average velocity $\widetilde{\mathbf{V}}$	103	
$\mathfrak{N}_n$	region of n voxels	103	
$thresh_{\text{ang}}$	user defined angular threshold (similarity measure)	103	
`Modulate(...)`	overall modulation function	101	
Flow visualization with anchor lines			
$\boldsymbol{\phi}_{t_0}^{t_1}$	flow map	107	Eq. 6.2
$\delta\mathbf{p}(t)$	infinitesimal distortion of a particle	108	Fig. 6.1
$\mathbb{C}$	Cauchy-Green tensor (deformation tensor)	108	Eq. 6.6
$\boldsymbol{\lambda}_{\mathbf{max}}$	largest eigenvalue of $\mathbb{C}$	109	
$\boldsymbol{\sigma}_{t_0}^{\Delta t}(\mathbf{p})$	largest finite-time Lyapunov exponent	109	Eq. 6.9
$\frac{\mathrm{d}\boldsymbol{\phi}_{t_0}^{t_1}(\mathbf{p})}{\mathrm{d}\mathbf{p}}$	gradient of the flow map	113	Eq. 6.10
t_{safe}	safety threshold for integration time	113	Fig. 6.4
t_{user}	user-defined threshold for integration time	113	
Δr	amount of particle scattering around the seeding point	115	
$\boldsymbol{\mathcal{A}}$	anchor line	115	
$\mathbb{P}_{\boldsymbol{\mathcal{A}}}$	accompanying particles of the anchor line $\boldsymbol{\mathcal{A}}$	115	
$\boldsymbol{\mathcal{P}}_{\boldsymbol{\mathcal{A}}}$	trajectory of the accompanying particle $\mathbb{P}_{\boldsymbol{\mathcal{A}}}$	116	Fig. 6.6
$\sigma_{\max}$	threshold of the finite-time Lyapunov exponent used for seeding of anchor lines	115	

Symbol	Description	Page	Place
Flow visualization with anchor lines			
$maxDev$	maximum deviation of a particle along its path	116	
age	particle age	117	
$\mathbf{a}$	position on the anchor line	116	
`PPosTex`	texture for particle positions and their age	117	Fig. 6.8
`ALPosTex`	texture for anchor lines positions	117	Fig. 6.8
`ALStartPosTex`	texture for starting positions of anchor lines	117	Fig. 6.8
`PStartPosTex`	texture for starting particle positions	118	Fig. 6.8
Diffusion tensor visualization			
J	movement of a material by a diffusion (flux)	127	
$\mathcal{C}$	concentration of the material	127	
$\mathbf{J}$	net flux	127	Eq. 7.1
$\nabla\mathcal{C}$	gradient of concentration	127	Eq. 7.1
$\mathbb{D}$	diffusion tensor	128	Eq. 7.2
D_{ij}	components of diffusion tensor	128	Eq. 7.2
$\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$	eigenvectors of the diffusion tensor	128	
$\lambda_1 \geq \lambda_2 \geq \lambda_3$	eigenvalues of the diffusion tensor	128	
c_l	coefficient of linear tensor anisotropy	129	Tab. 7.1
c_p	coefficient of planar tensor anisotropy	129	Tab. 7.1
c_s	coefficient of spherical tensor anisotropy	129	Tab. 7.1
μ_1	1^{st} central moment of eigenvalues	129	
μ_2	2^{nd} central moment of eigenvalues	129	
A_3	skewness of eigenvalues	129	
J_4	tensor invariant	129	
FA	fractional anisotropy	129	
RA	relative anisotropy	129	
$\mathbb{E}_{3\times3}$	3×3 identity matrix	133	Eq. 7.8
$\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$	diffusion tensor invariants	134	Eq. 7.9-7.10
$\mathbf{\hat{e}_i}$	normalized eignevector	135	Eq. 7.18
$\mathbf{d}_{\text{def}}$	deflection direction	136	Eq. 7.21
$\mathbf{d}_{\text{in}}$	incoming direction for deflection	136	Eq. 7.21
$\mathbf{d}_{\text{out}}$	propagation direction for tensor line	137	Eq. 7.22
ω_{punct}	user-controlled parameter for deflection	137	Eq. 7.22

Symbol	**Description**	**Page**	**Place**
Diffusion tensor visualization			
c_a	overall anisotropy metric	142	
$\mathbb{M}_{\mathrm{dir}}^{\mathrm{DT}}$	transformation of texture coordinates of the point sprite for diffusion tensor field	143	Eq.7.23
`TensorField`	two textures for tensor field values	141	Eq. 7.7
`ValidBitMask`	bit-mask of valid regions within the volume	138	
`StartPosTex`	texture for starting particle position	141	Eq. 7.7
`PosTex`	texture for particle positions	141	Eq. 7.7
`DirTex`(t)	texture for particle directions of propagation	141	Eq. 7.7
`EigenvaluesTex`	texture for eigenvalues	141	Eq. 7.7
`EigenvectorTex`	texture for eigenvectors $\hat{\mathrm{e}}_1$	141	Eq. 7.7
$\sigma_{\mathrm{iso\_max}}$	threshold for anisotropy used for rendering	144	
$\sigma_{\mathrm{iso\_min}}$	threshold for anisotropy used for rendering	144	
σ_{valid}	validity threshold	145	
$validBit$	single bit in the validity bit-mask	144	
$aniso$	anisotropy metric for rendering	145	Eq. 7.25
$\boldsymbol{Opacity}(aniso)$	anisotropy-based transfer function	145	Eq. 7.25

Chapter 1

Introduction

1.1 Motivation

The investigation of flow fields is of great importance in many areas of science from fluid mechanics and aerodynamics to biology and medicine. In general, the exploration of flow fields is performed on an interdisciplinary level implying close collaboration between physicists, mathematicians, computer scientists, and engineers in a particular applied science, e.g., fluid dynamics, biology, or medicine (see Figure 1.1). Typically, physicists are setting up the experiment and are performing the measurements of the flow, taking into account the specifics of the investigated phenomena explained by the engineers. The result of such measurements is a bulk of raw data, e.g., images in a 2D or 3D spacial domain, which have to be processed and discretized by mathematicians, taking into account the features of the underlying process suggested by the engineers. Alternatively, the digital representation of the flow can be obtained as the output of numerical simulations. The task of computer scientists is to efficiently represent the digital data by means of computers and other related equipment in order to help the engineers to interpret the results and to gain insight into the phenomena of their interest.

Looking over all these complications related to the flow exploration, two reasonable questions arise: Why would anyone need to spend all these efforts to study the flow? Why not just gain insight by direct observations? There are several answers for these questions. The direct observation can only provide the qualitative description of the flow while the detailed dynamics and precise mechanisms of feature development remain concealed deep inside of the flow. Moreover, in complex flow scenarios the characteristic features of the flow are appearing, developing, and disappearing, and there is no possibility to reverse the process or stop it to study these features in more detail.

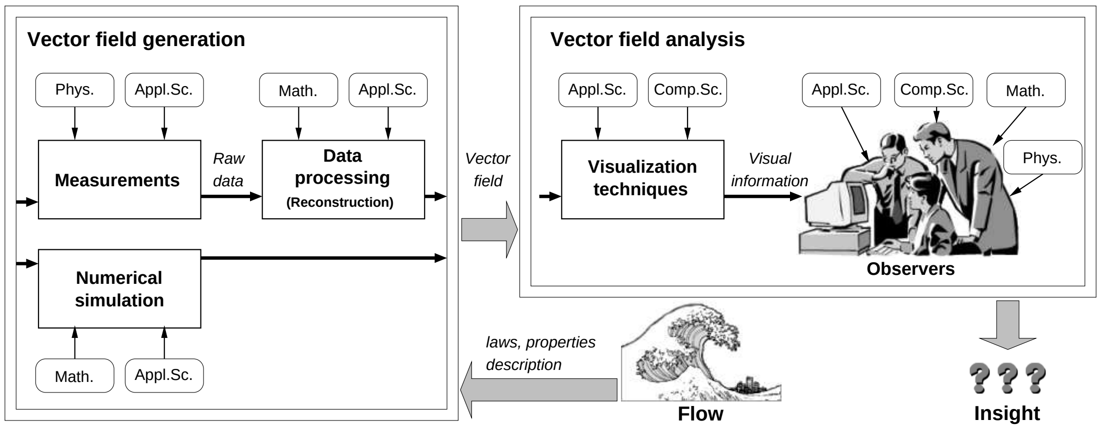


Figure 1.1: *Flow exploration.*

Thus, the flow investigation should be performed via thoroughly planned experiments, allowing for capturing of instantaneous flow states, efficient extraction of all the meaningful information from the recordings, and explanatory quantitative analysis of the results. In this way, the insight gained during the experiments can be efficiently shared and discussed among the scientists, helping them to conduct the research in a new light. Taking into account the recent advances in computational fluid dynamics, it is also of increasing importance to use the results of experimental studies for the validation of numerical flow models.

In order to understand the unknown flow phenomena, it is necessary to find out, what is happening inside of the flow domain under investigation. In other words, it is important to obtain the physical properties of the flow, such as velocity, pressure, density, viscosity, etc., and determine their dynamics. Furthermore, it is of great interest to figure out how the characteristic flow pattern is affected by the change of these quantities or the influence of external forces. Today, due to the progress in lasers, optoelectronics, video and computer techniques made in the last few decades, it has become possible to obtain the important physical quantities of the flow at high frame rates and store them as a sequence of instantaneous flow states [RWK01].

The velocity of the flow and quantities derived from it are the most important physical properties of the flow as they reveal its dynamics. Note that in general, it is not possible to measure the velocity directly. Quantitative flow exploration is not an exception from this rule. The standard method to evaluate the instantaneous velocity in each point of the flow domain is to seed the particle tracers into the flow and record the subsequent particle images at appropriate frame rates. In this way, the displacement of each particle or at least a group of particles can be defined. Knowing the timestep between the flow recordings, instantaneous velocity maps can be computed. If the particles are small enough and closely follow the flow without altering it or lagging behind it, the instantaneous velocity map extracted from the particle images corresponds to the actual flow field up to a certain degree of accuracy [Wes97].

Despite its seeming simplicity, the reconstruction of vector fields from experimental image sequences is a challenging task. While setting up the equipment for flow measurements, each flow situation should be carefully analyzed and experimental setup adjusted according to the application-specific requirements. For instance, when the measurements are performed in flows induced by living organisms, substantial efforts should be made to avoid any artificial intrusion into the natural environment of the organisms that could affect their behavior. Possible application-specific restrictions imposed on flow measurement systems involve reduction of seeding density and restrained characteristics of particle tracers, limitations on lighting conditions, restricted range of temperatures, etc. Moreover, the phenomenon under investigation per se can be characterized by very complex flow pattern covering a wide dynamic range of velocities within the flow domain. In addition taking into account general measurement imperfections and discretization errors of the imaging process, the problem of the reconstruction of vector fields from experimental image sequences becomes underconstrained. Therefore, it is nearly impossible to reconstruct the admissible flow fields from such sequences using standard methods. Recently, model-based approaches positioned themselves as an efficient means to resolve the aforementioned problems. These methods introduce additional constraints to the intrinsically ill-posed problem based on a priori information, and thus allow for the reliable reconstruction of flow fields from experimental image sequences.

Once the velocity field is reconstructed, it should be visualized in an efficient and intuitive way in order to provide the observers with meaningful information for the understanding of the flow phenomenon under investigation. The simplest method for 2D flow portrayal is an arrow plot, where each arrow is associated with the velocity vector at a given position in the spatial domain of the flow. However, this representation does not give all the insights that can be gained from the velocity field. Many sophisticated visualization algorithms described in the literature allow for the extraction of more valuable information, such as path-lines, vorticity, shear, strain, topological structure of the flow, etc., which enable the observer to see the unseen. Since all the data for 2D flow is stored in one plane, its visualization is more or less straightforward. Using the wide range of techniques proposed so far, the 2D flow information can be already efficiently presented to the observer [PvW93].

In 3D flows the situation is much more complicated: the large amount of densely packed self-obstructing information should be depicted at once, still enabling the user to intuitively untangle the complex data dependencies and clearly see the interweaving flow patterns. A number of sophisticated algorithms have been already elaborated for this purpose and offered for exploitation to engineers. However, judging by their tendency to use good old 2D arrow plots dragged through the volumes instead, the visual representation of 3D vector fields is still a great challenge for the visualization community [FG98].

Since many physical properties of fluid flow, such as rate of strain, stress, velocity gradient, etc., are described in terms of tensor data, it is highly important to enable the visualization of this data as well. It is worth mentioning that direct measurement of tensor fields is not possible in general. Tensors belong to a special category of multivariate data, incorporating valuable physical information about the underlying phenomenon. Taking into account the challenge of visualizing the vector fields which are essentially the tensor fields of first-order, visualization of general tensors, e.g., stress or strain represented by second-order tensors, is not a trivial task. The physical interpretation of the results of visualization is highly important and it strongly depends on a particular application. However, it is not obvious how to find the intuitive "mapping" between the mathematical and physical properties of the tensors. Therefore, visualization of tensor fields is another open question in the area of visual flow exploration.

Despite of the great advances in different branches of science collaborating under the headline of flow exploration, there is still a long thorny way to go pursuing the main goal of the research in this area. According to the observations made by McCormick et al. [MDB87], scientists want to literally interact with their computational or experimental data and they want to see immediately how the phenomenon under investigation is affected by the change of parameters. In other words, the interactive steering and dynamical modification of the investigation process has to be achieved. Thus, the main goal of visualization is to provide the scientific exploration with new insights, encourage the creation of new approaches, and change the way scientists do science [MDB87]. Therefore, our goal is to follow this direction and to make our contribution by developing novel approaches.

1.2 Contribution

This dissertation presents various methods dedicated to two aspects of flow exploration: the reconstruction of flow fields from experimental image sequence and the interactive visualization of flow fields. Flow exploration is a challenging task, requiring a certain knowledge in many disciplines, including mathematics, physics and, in particular, fluid mechanics, and computer graphics. Therefore, the chapters describing each method contain a lot of expressions specific to all of these areas. In order to allow the reader to understand these specific terms, I tried to briefly explain and emphasize the most relevant terms in introductory sections. The most essential formulae and short theoretical background underlying the application-specific processes are also provided. Moreover, since the central theme of the whole thesis draws on the theory of flow fields, a detailed introduction into this topic is included as well.

1.2.1 Tribute to the Particle Engine

All the problems investigated in this thesis are related to the flow. Taking into account its long history, which started from qualitative analysis performed via passive observation of moving particle tracers immersed into the flow, and nowadays enabling a comprehensive quantitative analysis of the flow phenomena under investigation, *particle tracing* has established itself as one of the most powerful and intuitive techniques for flow visualization. Moreover, there is a vast number of applications, where moving particles (discrete or connected) allow to reveal the features within the investigated data set and give insight into the underlying process. A number of research papers published by our Chair [Cha] as well as two international contests, which we have won, embracing different scientific areas from flow visualization and medicine to geomechanics and tectonics, prove the value of the particle tracing paradigm.

As a consequence of collaborative research conducted in our Chair, particle tracing has been efficiently implemented on programmable graphics hardware, which allows for interactive streaming and rendering of millions of particles and enables the virtual exploration of high resolution data fields [KKKW05]. Additionally, our *particle engine* has been extended by a linear algebra framework and a Navier-Stokes simulation module [KW03], which allows to interactively modify the flow field and immediately visualize the effects of imposed changes.

All of the methods presented in this thesis are implemented as modules within the particle engine framework and they extend it into a *comprehensive tool for interactive flow exploration*, which includes numerical simulation, model-based reconstruction, and feature-based flow visualization. Since many co-workers of our Chair as well as students have contributed to the particle engine framework, I cannot use “I” when I describe the developed methods in this thesis. Moreover, all my publications include the co-authors, which contributed in different ways to the work associated with these papers. Therefore, through the rest of this thesis I will use the academic *we* in order to acknowledge the collaboration with my colleagues and partners of interdisciplinary projects.

1.2.2 Contribution

The main results of the four-years work described in this dissertation are the following novel algorithms:

- a model-based approach for the reconstruction of flow fields from image pairs,
- anchor lines: a feature-based importance-driven technique for flow visualization,
- visualization techniques for diffusion tensor fields based on the particle tracing paradigm.

All of these algorithms incorporate knowledge from different areas of science. Therefore, the value of this dissertation cannot be judged solely by the number of algorithms. Theoretically sound modeling, careful adjustment of these algorithms, and their efficient implementation, is what makes the most value and was most time consuming. In the following few paragraphs, the history behind each algorithm is briefly summarized, which in my opinion describes the research process itself, and thus, the contribution of this dissertation.

A Model-based Reconstruction Algorithm

The development of the reconstruction algorithm has been the most theoretically involved and challenging problem investigated in this thesis. The consecutive development of this method is also depicted by the number of publications dedicated to it [KGW06, PDK$^+$06, KGW$^+$08]. First of all, we have had to make a motivated choice on the direction of development of the reconstruction algorithm, based on an analysis of the drawbacks of already existing algorithms. The results of our investigations on this issue are summarized in Section 2.3. As the result of this research, we have decided to follow the most promising direction related to the class of *model-based* reconstruction algorithms.

Since we have chosen the model-based direction, a lot of literature has been investigated in order to find the appropriate models and physically-based scaling factors to make the reconstruction of vector fields reliable and consistent with the actual model of the underlying flow phenomenon. It is worth mentioning that initially the purpose of this reconstruction algorithm was to improve the quality of flow fields reconstructed from image sequences recorded in experiments with microorganisms. Therefore, related work about the characteristic pattern of the underlying flow, as well as assumptions about the numerical flow model have been thoroughly investigated. The result of this research on the border between microbiology and fluid mechanics is presented in a compressed form in Section 4.3.

As the final step, we have thoroughly validated the proposed reconstruction method and compared its performance to other algorithms. Validation is a crucial issue for algorithms in general and it is especially critical for algorithms dealing with real-world experimental data. We have tested our reconstruction algorithm quantitatively using synthetic data sets for which the ground truth vector field is known, and semi-qualitatively for experimental image sequences recorded in two different experiments with living microorganisms. The results of validation are presented in Chapter 4.

Anchor Lines

We have started the development of the anchor lines technique from experiments on the visualization of various complex flow fields in 3D. In this way, the implementation basis for interactive visualization of unsteady 3D flows with a diversity of visualization options including volume rendering of the derived flow properties has been established [BSK$^+$07]. However, the problem of adequate visual perception was unresolved in all of our experiments. We have studied the literature related to this problem arising in many areas of science, including medicine.

Focus and context techniques have appeared to us as the most promising direction allowing to reduce the amount of information displayed at once, yet emphasizing the important structures within the data set. While in medicine it is straightforward to choose what is of importance, the decision on importance of a region in fluid flow is a non-trivial task. A number of recent research papers on flow visualization has been dedicated to topological and feature-based visualization methods. Therefore, in our first experiments we have computed various quantities from given vector fields and used these quantities as importance measures for emphasizing the flow features. Focus and context techniques based on these measures have been adopted and embedded into the particle engine. Moreover, several information reduction techniques based on similarity of flow regions, for example arrow clusters, have been implemented [BKKW08]. The aforementioned improvements within the particle engine framework are described in more detail in Section 5.3.

While experimenting with the visualization of complex flow phenomena, we have noticed that sometimes, when the seeding probe for particles is sufficiently small, the trajectories traced by the particles can evolve into a complex but clearly visible flow pattern. It was also observed that seeding the particles in other regions yields a bundle of trajectories of similar shape. Inspired by these experimental results, we have studied the theory behind this phenomenon and found out that the quantity, which describes the degree of separation between the particle trajectories, is the finite-time Lyapunov exponent. Thus, employing the finite-time Lyapunov exponent as importance measure, we have discovered the theoretically sound and physically meaningful method for detection of heterogeneous regions in the flow and we called it *anchor lines*. The theoretical aspects as well as implementation issues related to this technique are presented in Chapter 6.

Visualization Techniques for Diffusion Tensor Fields

Before we have started to work with diffusion tensors, we were looking for appealing and unusual areas of application for the particle tracing paradigm. Our attention was drawn to the diffusion process in highly anisotropic tissues such as the human brain, where molecular pathways reveal the direction of actual neural fibers. By thorough modification of the routines for classical particle/line tracing based on the physical meaning of the underlying diffusion tensors and integration of a number of new medically-based visualization modes we have created an efficient and interactive tool for intuitive visualization of diffusion tensor fields [KKW05].

It is worth mentioning that we have encountered a number of application-specific problems on implementation stage. To resolve them, we consulted the literature dedicated to diffusion tensor fields. We have discovered several theoretical issues related to tensor fields in medical applications, which have never occurred to us while we have been working solely with vector fields. This includes the non-iterative solution of the eigenvalue problem associated with tensors, the diffusion-driven line/particle propagation algorithm, the visualization of anisotropy, the medically-based termination criteria for the line/particle propagation procedure, etc. All of these issues are addressed in detail in Chapter 7.

1.2.3 Outline of This Thesis

The hierarchical structure and logical ordering of this dissertation is as follows. Since the thesis deals with two aspects of flow exploration, namely reconstruction and visualization, the text is split into two logical parts: *Model-Based Reconstruction of Vector-Fields from Image Sequences* and *Interactive Visualization of Flow Fields*. Each part contains an introductory chapter into the subject, where the terminology, the theoretical background, and related work are presented in a compressed format. The developed methods are described in detail in the following chapters. The efficiency of each algorithm is demonstrated in a number of examples including a performance analysis at the end of the corresponding chapter.

Chapter 2 provides an introduction to the subject of flow fields. It contains the mathematical and physical definition of flow fields as well as the information about the main types of flows and their features in terms of fluid mechanics. Additionally, a short description of the most popular methods for experimental flow measurement is presented, including an example of the experimental setup adjusted for experiments with microorganisms. An overview of existing reconstruction techniques from image sequences, compiled at the end of this chapter, exposes the general problems related to the reconstruction process.

Chapter 3 is dedicated completely to the model-based reconstruction method and provides theoretical as well as practical aspects of its development. The theoretical part contains an overview of the algorithm including the functional scheme and the block diagram, the choice of the basis approach for the prediction step, as well as the description of various flow models incorporated into the correction step. The practical part addresses application-specific numerical and implementation issues related to the proposed reconstruction method.

Chapter 4 compiles the results of the validation of the model-based reconstruction algorithm presented in Chapter 3. In order to give a methodological overview of the validation process, a list of error measures as well as a short description of the external software applications involved in the validation process, including the standard tools for the reconstruction of flow fields from image sequences, is provided in the beginning of this chapter. A thorough validation of efficiency and accuracy of our algorithm in comparison to standard approaches is performed using two different synthetic image sequences, each supplied with the ground truth vector field. Furthermore, a semi-qualitative analysis of the efficiency of our method is demonstrated in an application to two real-world microbiological image sequences. The results of the reconstruction for experimental image sequences have been compared to the results provided by a particle tracking algorithm applied to several particles segmented from the images. The comparison of our method to standard methods in terms of computational costs is presented at the end of this chapter.

Chapter 5 gives an introduction to the second main topic of the thesis: interactive visualization of flow fields. A short overview of the state-of-the-art techniques describes solved and unsolved problems related to flow visualization in general. The basic mathematical description of the particle tracing paradigm underlying the proposed visualization methods can be found in this chapter as well. In order to give the reader a rough idea about the technical foundation of the developed visualization techniques, we briefly summarize the most important aspects related to the efficient implementation of the core particle tracing algorithm on the GPU.

Chapter 6 provides a detailed description of the anchor lines technique. It contains the relevant information on theoretical as well as practical aspects related to the computation of the finite-time Lyapunov exponent. The description of the anchor lines technique itself includes a discussion about the problems encountered during the implementation and their practical solution. The efficiency of anchor lines in combination with other feature-based techniques is demonstrated by employing them for visualization of different flow scenarios.

Chapter 7 is focused on the visualization of diffusion tensor fields. The chapter starts from a concise theoretical background for diffusion tensors and their properties as well as an overview of the related work on visualization of diffusion tensor fields. Then we proceed by presenting the details on the implementation of our method, including a number of encountered difficulties and their elimination. Finally, the advantages of the proposed visualization technique are shown using the rendering results and the performance analysis.

The main results of this thesis as well as directions for further research are summarized in Chapter 8.

Part I

Model-Based Reconstruction of Vector-Fields from Image Sequences

Chapter 2

Experimental Exploration of Flow Fields: From Theory to Practice

The logical step before delving deep into the implementation of any algorithm is to clarify the terminology, specify the input and the output of the algorithm, and describe its desired behavior in defined terms. This chapter provides an introduction into flow fields and flow measurement techniques as well as a review of related work in the area of flow field reconstruction.

2.1 Flow Fields

Before starting with the description of reconstruction and visualization of flow fields, let us first clarify, what do the flow fields represent. Surprisingly, there is no unique definition of the *flow field*. In various areas of science and even within the same scientific domain the term "flow field" is interpreted in different ways. In this section we will show several common definitions.

In mathematics and physics, a *field* is a map that assigns to each point in a given spatial domain and/or temporal domain a certain value. Fields can consist of scalars, vectors, or tensors or any combination of these quantities [SPS99]. For example, a three-dimensional *vector field* in 3D can be represented by a global analytical function $\boldsymbol{f}(x, y, z)$ defined over a bounded spatial domain $\boldsymbol{\Omega} \subset \mathfrak{R}^3$. Alternatively, the vector field can be specified by a map $\boldsymbol{\mathcal{F}}$: $\mathfrak{R}^3 \rightarrow \mathfrak{R}^3$. A field value Φ can be found at any point (x, y, z) of the domain by direct computation of the value of the analytical function at this point: $\Phi = \boldsymbol{f}(x, y, z)$. Note that the situation is a bit more complicated in the discrete case: the field is specified only at grid nodes. Thus, in order to get a field value at any position within a grid, an *interpolation* between the values at the neighboring grid nodes is required.

2.1.1 Flow Fields in Fluid Mechanics

In fluid mechanics and other technical sciences all liquids and gases are considered as *fluids*. Fluids are substances, for which even a small shear force applied externally causes a deformation of their molecular structure. This deformation of the internal fluid structure essentially is the *flow*. In general, the fluid flow can be caused by action of various types of external forces, such as gravity, pressure differences, rotation, shear, and surface tension.

While all fluids behave similarly under the action of external forces, their *macroscopic properties* differ considerably. Among the most important and simple properties of fluids are the *density* and *viscosity*. Although there is a substantial difference between liquids and gases, both types of fluids obey the same laws of motion. In terms of field theory, a fluid is a *continuum* with respect to all relevant properties: all the properties of interest are defined *everywhere* within a given domain [FP02].

In fluid mechanics the term *flow* is sometimes related to the *moving fluid* itself with all its properties. From this point of view, which is traditionally called Eulerian approach, a *flow field* is a collection of all the relevant properties of the fluid defined everywhere over the spatial as well as temporal domain [PG92]. Thus, each entry of the flow field consists of a number of scalar (density, pressure, viscosity), vector (velocity, acceleration), and even tensor (stress, strain) values [LV84].

More often, the term *flow* denotes the *motion of fluid* or *fluid dynamics*. This approach takes its roots from the first scientific attempts to understand flow phenomena. To get a rough idea about the structure of the underlying flow the observer needs to see how, namely with which speed and in which direction, the particle tracers are advected within the flow. In this case, the flow field is the corresponding *velocity vector field* of a fluid or just *vector field* under investigation. Thus, the expression “reconstruction of a flow field” is related to the reconstruction of velocity vectors in each grid point of the fluid domain. An example of the reconstruction of a velocity vector field from a real-world flow image is presented in Figure 2.1. Note that the properties of the flow field are the quantities derived from the velocity vector field.

The velocity is the time derivative of the position. Therefore, a velocity vector field can be constructed if we compute this time derivative at every point in space. In the simplest case, the time-independent (steady) vector field over the spatial n-dimensional domain Ω can be computed as follows:

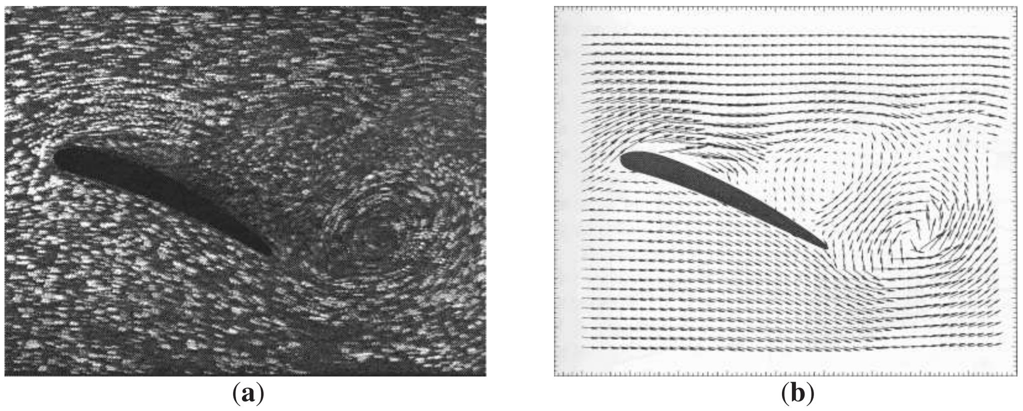


Figure 2.1: *Separated flow behind wing: (**a**) particle image; (**b**) reconstructed instantaneous vector field.*

$$\begin{aligned} \mathbf{V}(\mathbf{p}) &: \mathbf{\Omega} \rightarrow \mathfrak{R}^n, \quad \mathbf{p} \in \mathbf{\Omega} \subseteq \mathfrak{R}^n \\ \mathbf{V}(\mathbf{p}) &= \frac{\mathrm{d}\,\mathbf{p}}{\mathrm{d}\,t}, \qquad t \in \mathfrak{R}, \quad \mathbf{V}(\mathbf{p}) \in \mathfrak{R}^n. \end{aligned} \tag{2.1}$$

Thus, the velocity vectors at every point $\mathbf{p}$ in space are tangent to the *flow lines* $\mathbf{p}(t)$ visualized by the particle paths at this point (see Figure 2.2 (a)). In the flow field described by Equation 2.1 particles seeded at the same position in space will follow the same path independent of the time instant when the particles are released [Jir03].

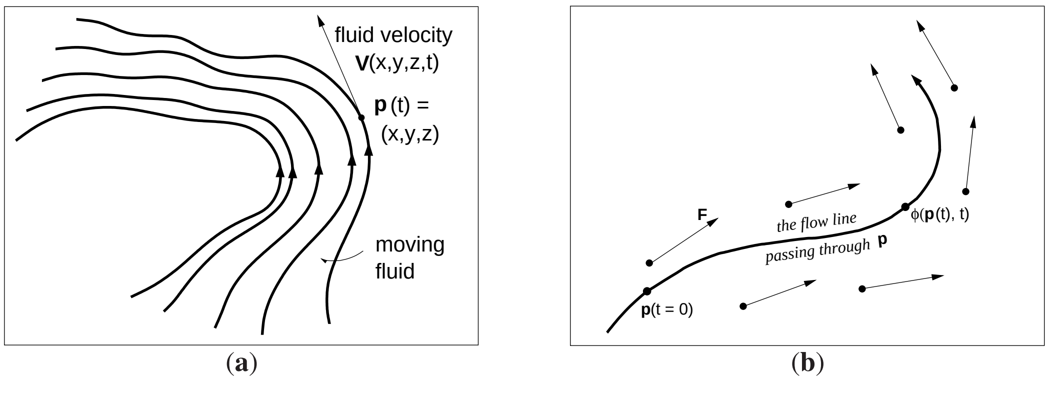


Figure 2.2: *Flow field in: (**a**) fluid mechanics; (**b**) vector calculus.*

In general, the velocity vector field itself can vary in time, causing the particles released at the same position in different moments of time to follow different trajectories. In this case, given the particle locations, the velocity vector field over the n-dimensional spatial domain $\mathbf{\Omega}$ and temporal domain $\mathbf{\Pi}$ can be computed as follows:

$$\begin{aligned}
&\mathbf{V}\big(\mathbf{p}(t),\, t\big) : \boldsymbol{\Omega} \times \boldsymbol{\Pi} \to \mathfrak{R}^n, \quad \mathbf{p}(t) \in \boldsymbol{\Omega} \subseteq \mathfrak{R}^n,\, t \in \boldsymbol{\Pi} \subseteq \mathfrak{R} \\
&\mathbf{V}\big(\mathbf{p}(t),\, t\big) = \left.\frac{\mathrm{d}\,\mathbf{p}(t^*)}{\mathrm{d}\,t^*}\right|_{t^*=t}, \quad \mathbf{V}\big(\mathbf{p}(t),\, t\big) \in \mathfrak{R}^n.
\end{aligned} \tag{2.2}$$

Note that with constant t, Equation 2.2 simplifies to Equation 2.1 for the steady flow situation.

2.1.2 Flow Field in Vector Calculus

In vector calculus there is no definition for a flow field as such. However, there is a rigorous definition of the velocity map of the fluid in 3D, which assigns a vector $\mathbf{V}(\mathbf{p}, t)$ to each point $(\mathbf{p}, t)$ in its domain $\boldsymbol{\Omega}$:

$$\mathbf{V}(\mathbf{p},\, t) :\ \mathfrak{R}^4 \to \mathfrak{R}^3, \tag{2.3}$$

The term *flow* is associated with any vector field, no matter whether it is a velocity fluid map or any other abstract vector field. Flow is defined as the mapping $\phi(\mathbf{p},\ t)$, which associates the position of the point $\mathbf{p}$ at time 0 with the position of this point after time t has elapsed. Analytically, $\phi(\mathbf{p},\ t)$ is defined as follows [MT95]:

$$\begin{aligned}
\frac{\partial}{\partial t}\,\phi(\mathbf{p},\, t) &= \mathbf{V}(\phi(\mathbf{p},\, t)), \\
\phi(\mathbf{p},\, t) &= \mathbf{p}.
\end{aligned} \tag{2.4}$$

A graphical interpretation of the flow field definition given by Equation 2.4 is depicted in Figure 2.2 (b).

2.1.3 Flow Field as Dynamical System

Flow can be also considered from a completely different point of view: *dynamical systems*. Dynamical systems represent the evolution of a collection of interdependent quantities $\mathbf{M}$ within the common system according to a specific set of rules. The domain $\boldsymbol{\Theta}$ spanned by the system variable values is called the *phase space* of the dynamical system. A set of values $\mathbf{M}(t)$ build up the specific *state* of the system at a specific time t.

The evolution function Γ^t of a dynamical system can be obtained as the solution of a differential equation of motion:

$$\frac{\partial \mathbf{M}(t)}{\partial t} = \gamma(\mathbf{M}(t),\, t). \tag{2.5}$$

Technically, $\gamma(\mathbf{M}(t),\, t)$ represents the velocity vector at position $\mathbf{M}(t)$ in the phase space at time t. Thus, $\gamma(\mathbf{M}(t),\, t)$ specified over the whole domain $\mathbf{\Theta}$ is the velocity vector field defined at each position in the phase space. In this sense, *flow* can be interpreted as a special class of dynamical system [Löf98], for which the temporal domain is restricted to the non-negative real numbers. Depending on the content of the phase space, the *flow field* can be considered either as a velocity vector field or as a complex vector field, with each entry consisting of a collection of different quantities of the dynamical flow system (cf. Section 2.1.1).

2.1.4 Fluids and Flows

Fluid flows are characterized by a number of different physical parameters. The density ρ, dynamic viscosity μ, and velocity $\mathbf{V}$ are of particular interest for fluid mechanics. Depending on these parameters, flows can be classified in different ways. Note that this division into classes is not fixed, transitions between the different types of flows commonly take place.

Creeping, laminar, and turbulent flows. Depending on the speed of the flow, *creeping*, *laminar*, and *turbulent* flows[1] can be distinguished . At low speed, characterized with the Reynolds numbers $Re \leq 1$, the flow is dominated by the viscous forces, pressure, and body forces and is called *creeping flow*. This regime takes place in the flows through porous media, flows in narrow tubes, or in experiments conducted on microscopic scales.

The flows at increased velocities, characterized by a broad range of Reynolds numbers, are called *laminar flows*. Inertial forces in these flows are strong enough to move the fluid particles further even if the action of the external force has ended. The important feature of these flows is that the fluid particles follow the smooth trajectories without intermixing with the particles on the other fluid "layers". Laminar flows are the most common observed flows in the every day life.

A further increase in velocity leads to the *turbulent flow* regime, characterized by rather random, and instable flow patterns. Frictional force in such flows can be neglected. This type of flows can be observed, for example, in wind-tunnel experiments or, in real life, in the clouds behind an aircraft [FP02].

Compressible and incompressible flows. One of the important properties of flows is incompressibility. If the density of a fluid is constant under different conditions, this fluid is considered to be *incompressible*. Gases at low velocities, as well as most liquids,

[1] There are also subsonic, sonic, and hypersonic flows, but we do not consider them in this work.

are considered to be nearly incompressible. On the other side, if the density of a fluid is affected by the change of the pressure values, this fluid is considered to be *compressible* [GDN98]. At high velocities (e.g., supersonic and hypersonic flows), most of the fluid flows become compressible.

Viscous and inviscid flows. Separation of flows in viscous and inviscid is related to viscosity, the property of fluids that generates frictional forces within the fluid. In this sense, if the frictional forces are very strong, fluids are called *viscous*, otherwise – *inviscid*. In idealized form, gases are regarded as inviscid fluids. On the other side, honey or oil are highly viscous flows [GDN98].

It is worth mentioning that fluid mechanics is also dealing with other classifications, which are based on many other physical properties of the fluid flows, such as *buoyancy* or *phase*. Considering all the phenomena affecting the fluid flows and all the types of flows is out of the scope of this thesis. Therefore, we consider only a limited number of flow types. All the flow scenarios used for the validation of our reconstruction algorithm in Part I are constrained to *creeping* and *laminar incompressible viscous one-phase* flow models. Since the visual exploration of *turbulent* flows is a great challenge for the visualization community, the efficiency of developed visualization methods is demonstrated using this type of flows in Part II.

2.2 Flow Measurement

The most ancient and intuitive way to investigate flow is to passively observe the motion of particles seeded in the flow. In this way the great artist and scientist of the XVI century Leonardo Da Vinci provided the first detailed drawings of the complex structures within the turbulent flow (see Figure 2.3 (a)). A further great step forward in the flow exploration was made several centuries later by Ludwig Prandtl, who performed a qualitative analysis of the basic features of unsteady flows by careful planning the experiments and being able to vary a number of parameter settings (see Figure 2.3 (b)) [RWK01].

The development of the laser in the mid-1960s provided the researchers in experimental fluid mechanics with non-intrusive optical measurement techniques, and thus enabled for the first time the acquisition of time-resolved velocity information at a point in a flow [Hal88]. Today, the progress made with optical and video-recording techniques and the rapid development of computer-aided systems allows for the extraction

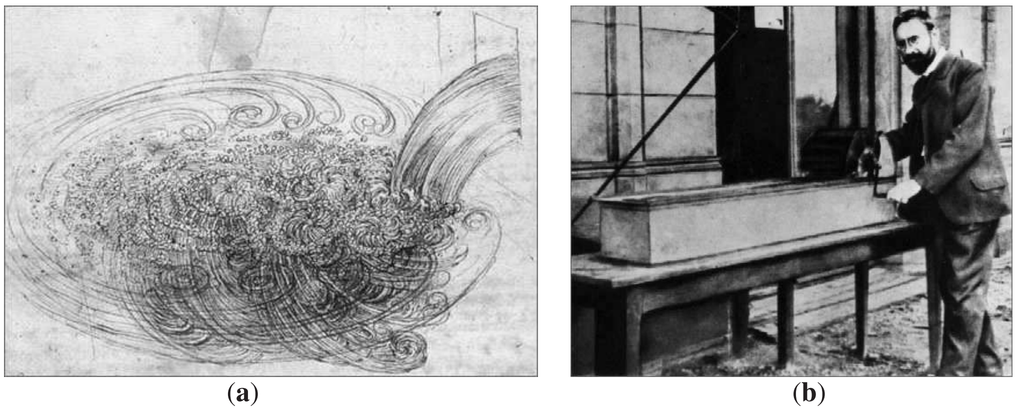


Figure 2.3: *History of flow exploration: (**a**) turbulence sketch by Leonardo Da Vinci; (**b**) Ludwig Prandtl in front of his water channel.*

Table 2.1: *State-of-the-art techniques for flow measurement.*

Abbreviation	**Full Name**	**Measured quantity**		**Reference**
PLV	Pulsed-Light Velocimetry	velocity	$\mathbf{V}$	[Adr91]
PSP	Pressure Sensitive Paint	pressure	p	[LCBS97]
TSP	Temperature Sensitive Paint	heat transfer,	H	
		temperature	T	[LCBS97]
BOS	Background Oriented Schlieren	density gradient	$\nabla\rho$	[RR01]
AMA	Acoustic Microphone Array	dynamic pressure	$p(t)$	
		on the surface		[BFH+02]
IPCT	Image Pattern Correlation Technique	deformation	$\Delta\mathbf{x}$	[Kir04]
		of the model		

of the whole field quantitative flow velocity information in a plane or even in a volume of the flow.

2.2.1 Variety of Flow Measurement Techniques

Today the most important physical properties of the flow can be measured quantitatively at high frame rates in a wide range of experiments from blood micro-channel to industrial wind-tunnel. Fluid mechanics exploits several measurement techniques to acquire the flow quantities of interest [Kom07]. The state-of-the-art techniques commonly used in the fluid mechanics laboratories are listed in Table 2.1.

Taking into account the variety of different flow features which can be explored, we will restrict ourselves to one feature, velocity, and focus on the exploration of *velocity flow fields*. Thus, Part I of this thesis is devoted to the reconstruction of velocity fields from the measurements provided by *Pulsed-Light Velocimetry* (PLV) [Adr91].

This choice is motivated by the increasing popularity of the PLV techniques, especially *Particle Image Velocimetry* (PIV), and a never-ending challenge related to the investigation of flow dynamics represented by velocity fields requiring the elaboration of efficient reconstruction and visualization techniques.

PLV is a group of particle-imaging techniques which indirectly measure the velocity flow fields by observing the motion of small distinguishable regions (markers) of the flow at two or more time steps. Solid and liquid particles present in liquids and gases are usually considered as markers. Additionally, gaseous particles can be also applied as markers in liquid flows. Other types of markers are fluorescenting molecular patches and speckle patterns formed by densely overlapping particles. The quantitative analysis of motion of image markers yields the displacement field of marker images. The velocity field is obtained using the fundamental definition of the local velocity given by Equation 2.2 [Adr91]. The large scale classification of PLV techniques based on the type of particle tracers and their seeding density is shown in Figure 2.4.

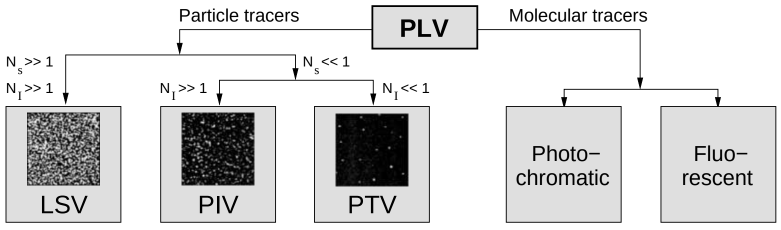


Figure 2.4: *Various forms of pulsed-laser velocimetry (PLV). The abbreviations stand for: LSV – Laser speckle velocimetry, PIV – Particle image velocimetry, PTV – Particle tracking velocimetry.*

The dimensionless numbers N_s and N_I in Figure 2.4 correspond to the *source density* and the *image density* [Adr91]. The source density corresponds to the mean number of particles in a resolution volume and can be used to define the number of the overlapping particles in the image plane. The image density is equal to the average number of particles in an interrogation area.

Interestingly, the experimental setup for all the PLV techniques based on particle markers is nearly the same, except the seeding density. Since almost all the experimental image sequences investigated in this thesis were recorded using PIV, a more detailed description of this method is provided in the next section.

2.2.2 Particle Image Velocimetry

Particle Image Velocimetry (PIV) is a well established non-intrusive PLV technique for measuring whole field instantaneous velocity maps indirectly via the displacement of flow *particle tracers* within a certain time interval Δt. The particle tracers can be naturally contained in the flow, as it is the case in microbiological PIV investigations (see Section 2.2.3), or artificially seeded into the flow before the experiment is started, as it is usually the case for most experiments not involving living beings. The typical experimental setup for PIV systems, as it is arranged, for example, for flow investigation in a wind tunnel, is presented in Figure 2.5.

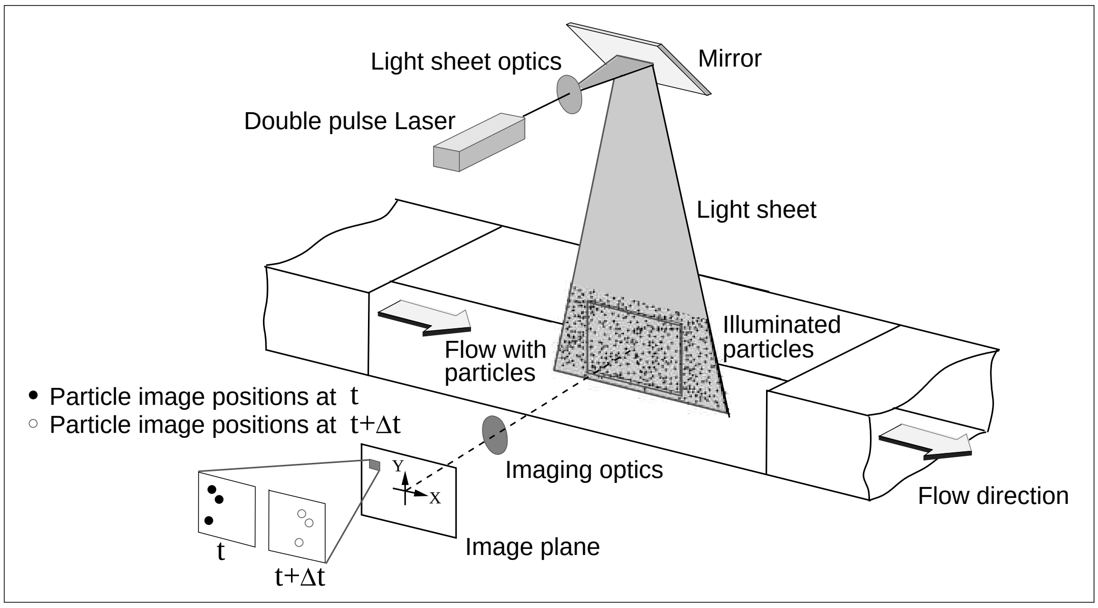


Figure 2.5: *Experimental setup for PIV systems.*

In most applications, a selected plane in the flow seeded with particles is illuminated twice by a thin laser light-sheet. The light scattered by the particle tracers towards the recording optics in the time moments t and t' is stored in a single frame or in two frames on the image plane of a special cross-correlation CCD[2] sensor. The capturing CCD camera is adjusted in such a way that its optical axis is perpendicular to the laser-sheet. The output of the CCD sensor is transferred in digital format into the memory of a computer, where the automatic analysis of the recorded flow images (reconstruction) is performed [RWK01].

Being methodologically developed from the Laser Speckle Photography (LSP) and terminologically described by [PH84] and [Adr84], the PIV technique has quickly become a state-of-the-art universal technique for quantitative measurement of flow velocity fields. There are several reasons for this increased interest in the PIV technique over

[2] CCD stands for charge-coupled device.

the last two decades:

Fast measurements. Today it is possible to perform the flow measurements online [RWK01]. The newest camera CMOS[3] sensors are able to perform full-resolution (1600×1200) measurements at 1000 fps and partial-resolution measurements at up to 60000 fps [ILA].

Simple particle seeding. In most practical applications the extreme efforts to provide the high concentration of seeding particles without alternating the underlying flow as required for LSP techniques are not justified [Adr84, PH84]. Thus, PIV with its relatively low particle density is the natural choice for many experiments. Moreover, in some applications the particles are naturally contained in the flow, thus the artificial flow seeding becomes unnecessary.

High quality wide range measurements. PIV provides the accurate high quality measurements of instantaneous velocity fields at every grid point of the flow domain. The laboratory scales of the measured flows are spanning the range from micrometers per second to several hundreds meters per second [Adr05].

Possibility to repeat the velocity evaluation. Since PIV records the full information about the flow field into the storage device, these data can be easily exchanged many times between many research groups for reconstruction and analysis. The information about the velocity fields is completely contained in the PIV recordings, thus the repetition of the experiment becomes unnecessary.

Possibility to calculate additional quantities. Ensuring that the flow fields are measured at each point of the dense grid, PIV enables the high quality acquisition of such important derived properties of the flow field as the deformation tensor and thus vorticity and rate-of-strain [Adr91].

Vast area of applications. Due to the flexibility and scalability of the PIV method, it can be applied to steady and unsteady flows of different nature. Investigations in various areas of natural sciences from aerodynamics to microbiology have proven this. Additionally, taking into account all its capabilities, PIV has become a powerful tool for experimental validation of numerical codes and models of complex flows [RWK01].

[3] CMOS stands for complementary metaloxidesemiconductor.

Besides the classification by the seeding density of the flow, PIV techniques can be also categorized by the number of illuminations and/or frames per recording. Accordingly, there exist three modes for the recording of the illuminated flow: *single*, *double*, and *multi-frame* mode. Depending on the camera settings, PIV images can be captured with *single*, *double*, and *multi-* exposure. All the possible combinations of PIV modes according to the number of frames and the exposure type are combined in Figure 2.6. The filled circles in the picture are the particle images written onto the current frame and the empty circles correspond to the particle images stored in the previous frames.

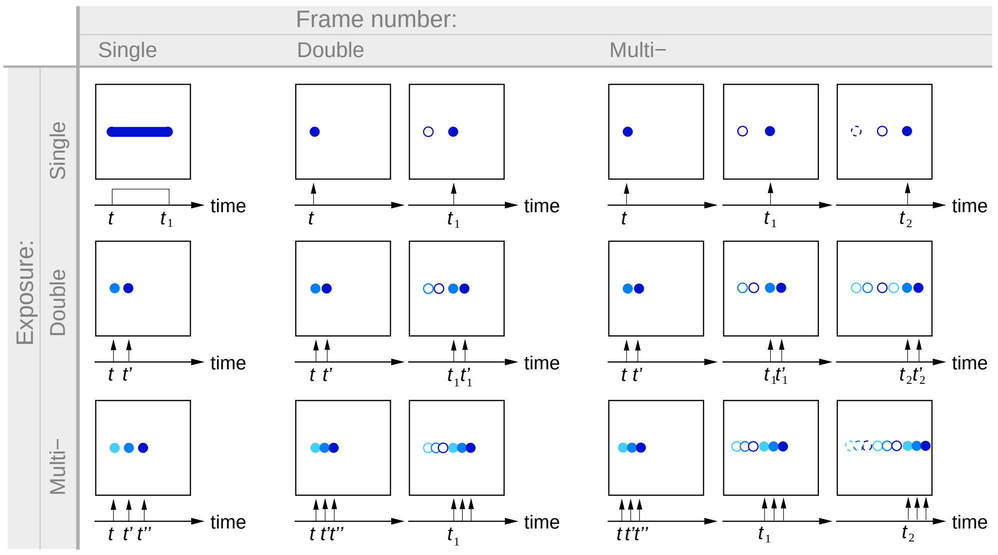


Figure 2.6: *PIV modes according to the number of illuminations and/or frames per recording.*

Finally, the PIV measurements can be distinguished by the dimensionality (D) and the number of components (C) per velocity vector. Besides the classical 2D-2C PIV recordings, there also exist 2D-3C and 3D-3C PIV modes. While 2D-3C techniques perform the slice-by-slice flow volume measurements allowing for extraction of the third component of velocity, 3D-3C methods enable the full-value three-dimensional measurements of the flow volume.

2.2.3 Application of Particle Image Velocimetry to Microbiological Experiments

Despite the fact that the experimental settings for PIV measurements in most applications are quite similar, each application should be carefully analyzed and the experimental setup should be adjusted according to the specific requirements for this application. This is especially the case when the measurements are performed for flows induced by living organisms, in particular, microorganisms. The restrictions imposed on the experimental setup are caused by the necessity to maintain the *biocompatibility*, and thus to avoid any artificial intrusion into the natural milieu of the organisms that could affect their behavior.

In biological systems, fluid motion is induced via different types of activity of the organisms such as locomotion through the fluid volume (for *motile* organisms) or cilia beat and stalk contraction (for *sessile* organisms)[4]. The activity of microorganisms is governed by the principles of minimum energy and is aiming towards the efficient feeding processes or movement into the location with more favorable environmental conditions, which in general is supposed to ensure the flourishing of the biological system. Therefore, a better understanding of natural biological processes caused by living organisms is of ever growing interest in different areas of industry from food production to water purification. PIV enables the investigation of the fluid exchange characteristics and the dynamics of the induced micro-flows, thus allowing researchers to gain insight into the mechanisms steering the biological processes and helping to adapt them for technical applications.

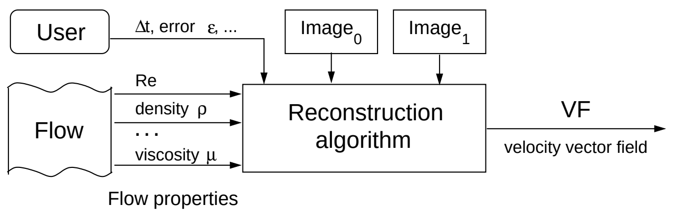
In an experimental setup for the investigation of living microorganisms the biocompatibility implies that the illumination must not be too powerful, and thus the traditional laser-sheet technique is not applicable for PIV measurements in this case. Moreover, the density and the size of seeded particles should be carefully adjusted [HOP$^+$07, PKD$^+$07]. Since microorganisms are able to efficiently detect the synthetic particle tracers and reject them as soon as they approach their "mouth", traditional particle tracers cannot be used to seed the flow. On the other hand it was observed that *biotic particle tracers*, such as yeast cells are recognized by the microorganisms as nutrients and thus almost perfectly follow the flow [HOP$^+$07].

Taking into account all these restrictions, the experimental setup of a PIV system for investigation of fluid flows induced by living microorganisms is adjusted as follows. An inverted microscope [Gol98] equipped with differential interference contrast [Mur01] is used to observe the flow. The light source included in the microscope is used for illumination. The camera is mounted directly onto the microscope objective. The biotic particle tracers are injected into the flow using a pipette. Finally, the sequences of illuminated instantaneous flow images are recorded and stored in the same way as for all PIV techniques.

[4] *Motile* organisms are able to move spontaneously and independently. *Sessile* organisms are permanently attached or fixed to some surface and are not able to move freely.

2.3 Reconstruction Techniques

After the introduction into the subject, we are now ready to describe the behavior of the target reconstruction algorithm and to define its input and output. While doing this, it is important to keep in mind Figure 1.1 since it gives the global picture of the flow exploration process. A schematic representation of the target reconstruction algorithm as a black box is depicted in Figure 2.7.



Goal: Given an input image sequence, compute the corresponding vector field (VF) as fast as possible. The reconstructed VF should match the underlying flow pattern with a high level of accuracy.

Remarks:

1. Incorporate all the available and relevant information about the experimental setup into the reconstruction process.
2. Provide interactive steering of the reconstruction process via user-friendly interface and direct visualization of the current result.

Figure 2.7: *Description of the reconstruction algorithm.*

In simple words, the reconstruction process (or in some literature *interrogation*) is the procedure which extracts the velocity vector field from a given image sequence. Since the reconstruction process is strongly dependent on the mode of PIV recordings (cf. Figure 2.6), it is important to define precisely which mode was employed to acquire the data sets used for the validation of the proposed reconstruction method in Chapter 4. All the investigated experimental images are recorded using the *multi-frame single exposure* technique. Knowing the index number of each image, the direction of the flow can be unambiguously defined. To keep the further text concise, the terms "digital images", "PIV images", "particle images" or just "image sequence" will be used instead of the precise term "multi-frame single exposure digital images".

Given a sequence of digital images, there exist several ways to extract the displacements of the particle tracers and, thus, the velocity vector field. One of the oldest methods to obtain the quantitative information about the flow from the images was known already several decades ago [FT32]. This method is closely related to PTV and is performed by *manual tracing* of particle images. It is clear that this approach is applicable when only a few clearly distinguishable particles can be observed in the images. Taking into account ever increasing length of image sequences and spatial image resolutions in combination with the increasing complexity of the flow patterns under investigation, it is reasonable to exploit high-performance computers for the reconstruction purposes.

Nowadays, there exist three main groups of algorithms dedicated to the reconstruction of velocity fields from image sequences: *cross-correlation based* methods (CC), *optical flow based* methods (OF), and computerized version of *particle tracking* (PT). Taking into account the success in computational fluid dynamics and numerical simulations in general, model-based approaches built upon either of three algorithms, CC, OF, or PT, are becoming a new powerful means for the reliable reconstruction of vector fields [BAHH92].

According to the description of the reconstruction algorithm, it should evaluate the *velocity vector field*. Since PT-based methods allow only for tracking of individual particles, the stated task cannot be performed by definition (cf. Section 2.1). On the other hand, CC and OF based techniques provide the reconstruction of the whole velocity fields. Therefore, these methods can be used as a basis for a new reconstruction algorithm. A brief description of CC- and OF-based methods with their advantages and disadvantages is presented in Sections 2.3.2-2.3.4.

2.3.1 Image Registration

Before describing in more detail each class of the reconstruction algorithms, based on either CC or OF, it is important to mention that all of them can be unified under the term *Image Registration* (IR). In computer vision, *image registration* is the process, which computes a motion field that best aligns pixels in one frame of the image sequence with those in subsequent frames [BAHH92]. More precisely, given two images, the reference image $\mathtt{R}$ and the template image $\mathtt{T}$, the task is to find a spatial transformation φ that deforms $\mathtt{T}$ in such a way as to minimize the difference between $\mathtt{R}$ and modified $\mathtt{T}$:

$$\boldsymbol{Dist}\,(\mathtt{R},\,\mathtt{T}\circ\varphi) \;=\; \min, \tag{2.6}$$

where $\boldsymbol{Dist}$ denotes an appropriate distance measure. With respect to the assumed transformation φ, image registration techniques can roughly be classified into the following categories: *parametric*, *quasi-parametric* and *non-parametric*. *Parametric* approaches impose specific restrictions on the transformation, e.g., requiring the transformation to be rigid, polynomial, affine, etc. *Quasi-parametric* methods describe a motion of a pixel as a combination of a parametric component valid for the entire interrogation region and a spatially varying local component. *Non-parametric* approaches, on the other hand, are much more flexible in the type of transformation they compute. They are typically employed for the registration of measured real-world objects for which it is not always possible to find a parametric equation that properly describes the transformation of all the heterogeneous parts of these objects. CC and OF techniques fall into the category of non-parametric approaches.

Since the non-parametric minimization problem is ill-posed at the origin, it is usually solved using a regularization term, or smoother $\boldsymbol{Sm}$:

$$\boldsymbol{\tau}\,(\mathbf{d}) \;:=\; \boldsymbol{Dist}\,(\mathtt{R},\,\mathtt{T};\,\mathbf{d}) \;+\; \alpha\,\boldsymbol{Sm}\,(\mathbf{d}) \;=\; \min, \tag{2.7}$$

Here the transformation is split into a trivial part and the displacement $\mathbf{d}$, such that $\varphi(\mathbf{x}) \;=\; \mathbf{x} + \mathbf{d}(\mathbf{x})$ and $\boldsymbol{Dist}(\mathtt{R},\,\mathtt{T};\,\mathbf{d})$ = $\boldsymbol{Dist}\,(\mathtt{R}, \mathtt{T}(. + \mathbf{d}(.)))$. The regularizing term $\boldsymbol{Sm}$ can be chosen according to the properties of the deformed materials, for example, it can be based on the elastic potential (*elastic registration*), or the Navier-Stokes equations (*fluid registration*). A comprehensive survey of methods commonly applied in image registration along with a detailed description of parametric and non-parametric motion models can be found in [Mod04].

2.3.2 Techniques Based on Cross-Correlation

Given a number of point samples in two images, the natural way to find the interdependency between these samples is to analyze them using statistical approaches that probabilistically insure the proper matching between the samples and provide the average displacement of a group of particles within a small interrogation area. In order to evaluate the velocity in the whole field, all the interrogation areas covering the whole image space should be analyzed.

Employing the theory of linear digital signal processing each interrogation area in the particle image can be considered as a signal $\boldsymbol{\sigma_0}(x,\,y)$ defined over the two-dimensional domain[5]. The counterpart of this signal $\boldsymbol{\sigma_1}(x,\,y)$ can be found in subsequent particle images. The temporal modifications of the signal are mainly caused by the unknown spatial displacement function $\mathbf{d}(x,\,y)$. Mathematically, these temporal modification can be described by the discrete convolution function:

[5] Here (x, y) denote the coordinates within the two-dimensional domain.

$$\sigma_1[x,\, y] = \left[\sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \mathbf{d}[i-x,\, j-y] \cdot \sigma_0[i,\, j] \right]. \tag{2.8}$$

One of the standard statistical approaches for finding a spatial displacement function $\mathbf{d}[x,\, y]$ is *cross-correlation analysis*. The discrete cross-correlation function $\mathtt{CC}[x,\, y]$ of the sampled regions $\sigma_0[x,\, y]$ and $\sigma_1[x,\, y]$ is given by the expected value:

$$\mathtt{CC}_{\sigma_0,\sigma_1}[x, y] = \frac{\sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \sigma_0[i,\, j] \cdot \sigma_1[i+x,\, j+y]}{\sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \sigma_0[i,\, j] \cdot \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \sigma_1[i,\, j]}. \tag{2.9}$$

The clearly distinguished peaks of this function correspond to the regions, where the samples $\sigma_0[x,\, y]$ in one image match up with their shifted counterparts $\sigma_1[x,\, y]$ in a subsequent image. The position of the highest peak corresponds to the average displacement of particles within the interrogation region. In this way, the main idea underlying the cross-correlation based techniques is to detect the highest cross-correlation peak and to find its position in every interrogation area within the image, yielding the extraction of the whole velocity vector field [WG91].

The first digital implementation of this class of algorithms, named *digital* PIV (DPIV) was independently proposed in [WG91] and [Wes93]. At the core of this implementation is the property of Fourier transform, which simplifies the time-consuming cross-correlation operation in the spatial domain to trivial multiplication in frequency domain. It was shown by Willert and Gharib [WG91] that employing the fast Fourier transform (FFT) dramatically speedups the whole process without loss of precision. The latter fact made DPIV the best choice for experimental flow investigation for many years.

Throughout all the years of DPIV development, various sources of errors causing a decrease in precision of the reconstructed vector fields were discussed and multiple improvements were suggested by different authors. All the reported accuracy problems are related to measurement imperfections, quantization, and discrete stochastic nature of cross-correlation underlying the reconstruction method [Sca02]. The following types of reconstruction errors are distinguished in the literature:

- The *in-plane loss of pairs* is related to the disparity between the interrogation window size and the actual particles displacement. As a consequence, the dynamic range of velocity is limited [Adr91].

- The *out-of-plane loss of pairs* occurs due to the three-dimensional nature of the underlying flow. In this case the particles are leaving the plane of observation, and thus, corresponding pairing cannot be established [WG91], [HDG97].
- A *velocity bias* (underestimation) is related to the limitations imposed onto the size of the interrogation window, which enforces to compromise between spatial resolution and accuracy [Wes93].
- *Peak-locking* effect, which introduces a bias towards integer values of the displacements and occurs due to poor spatial resolution of particle images, wrap-around errors of the FFT, truncated particles at the borders or interrogation areas, or an inappropriate choice of the interpolation function for peak finding [Wes93], [NLRa].
- *Peak broadening* is caused by the presence of large velocity gradients. Loss of pairs in this case occurs due to the deformation of particle image pattern.
- Limitation of spatial resolution imposed by the insufficient number of particles within the interrogation window.

A variety of heuristical methods allowing for the elimination or partial reduction of these artifacts have been suggested in the literature. In the following we will shortly summarize the proposed heuristical methods.

The various sources of the *peak-locking* phenomenon have been detected and analyzed in [PALO, LK95, Wes98, NLRa]. The techniques proposed in order to eliminate this effect can be roughly classified into two groups. The first category of algorithms suggests to extend the support and improve the interpolation scheme in the correlation peak estimator [WG91, Wes93, MSLJ00, CK05, Roe]. The second group of algorithms is based on the iterative *image-deformation technique* [HFW93], which aligns the interrogation window along the flow and allows to get better statistical estimates of the correlation peak [TD95, JJDAF95, FD00, CK05, Sca02, WG03, NOBT05]. In addition, it was proven that normalization of the cross-correlation function allows to avoid the peak-locking problem related to the wrap-around error of the FFT method and particle truncation effect on the borders of the interrogation area [RRK98].

Since all the types of errors inherent to the DPIV method are mainly related to the discrete and stochastic nature of the cross-correlation process, compensation of this feature yields the solution to several problems simultaneously. In this way, the image-deformation methods and image-shifting techniques [Wil00, LPW03, YJW04] suited for elimination of the peak-locking effect also partially resolve the problem due to the presence of velocity gradients within the interrogation region [Sca02, NOBT05]. The image deformation methods align the particle images along the flow and adjust the correlation function for local velocity changes, thus preventing the averaging of the velocity within the interrogation window and preserving the velocity gradients.

Furthermore, many efforts have been spent on the improvement of the quality of the reconstructed vector fields in terms of spatial resolution, accuracy, and dynamic range. Following this trend, several iterative prediction-correction schemes [SR00, NLR01] and techniques directly modifying the correlation criterion [LK00, WM01] have been proposed. Moreover, many authors suggested hierarchical approaches which increase the spatial resolution of the reconstructed velocity fields and simultaneously reduce the bias error [KAZ95, Har00b, RFH02, NLR$^+$b, Sca04]. Unfortunately, it was shown that according to the *one-quarter rule*[6] [Adr97], even the most sophisticated hierarchical methods do not always produce satisfactory results in the presence of low magnitude displacements [GJ03, YJW04, TMKK05]. To cure this problem, it was suggested to use the hierarchical methods in combination with improved correlation peak-finding procedures and *outlier* removal techniques in each iteration or each level of the hierarchy.

The presence of outliers or *spurious vectors*, which significantly deviate in magnitude or/and direction from its close neighbors, is a critical point for all the CC-based reconstruction methods. The spurious vectors inevitably appear due to the presence of noise in measured particle images and the stochastic nature of the cross-correlation process, which destroys the correspondence between the position of the maximum peak of the correlation function and the true displacement. The problem of outliers was thoroughly discussed in the literature and various methods for detection and correction of these spurious vectors in the post-processing stage have been suggested. Most of the outlier-removal methods are based on traditional thresholding, filtering, and improved interpolation schemes[Wes94, NLR97, RWK01, Bol99, GDC00, YJW04, WS05]. Finally, some authors are arguing for incorporation of the outlier detection procedure directly into the cross-correlation process, avoiding the post-processing step [Har00a, RNI00].

Despite the fact that many heuristical approaches already have been proposed in order to improve the quality of the vector fields reconstructed by means of CC-based techniques, there is still an open discussion regarding several aspects: biased velocity estimation caused by the in-plane/out-of-plane loss of pairs and the finite extent

[6] *One-quarter rule*: to retain a large fraction of pairs of particle images within the interrogation area the maximum displacement should be less than one-quarter of the size of this area.

of the interrogation window, inefficiency of peak searching algorithms, and a necessity to compromise between the accuracy and spatial resolution. A more detailed overview of various CC-based techniques with an extensive discussion about its advantages and disadvantages in application to different flow situations can be found in [RWK01, Sca02, Adr05, NOBT05].

It is worth mentioning that an *iterative coarse-to-fine multigrid algorithm with window shift and deformation* [Sca02] combined with the *outlier correction* on each step is considered to be the state-of-the-art CC-based method for high quality reconstruction of vector fields from PIV images [NOBT05].

2.3.3 Techniques Based on Optical Flow

Originally, optical flow (OF) was developed for detecting motion of large objects in real world scenes [HS81]. Inspired by the promising results demonstrated by the CC-based techniques with image deformation [HFW93, TD95] and supported by the ability of OF to return dense motion fields independent of the interrogation window size, several researchers [QPK98, MP99, RKSN05] proposed to adapt the OF method for fluid flow exploration. It was shown that the conceptual problems inherent to the CC-based DPIV techniques, namely, the trade-off between spatial resolution and reliability of the reconstructed vector fields as well as insufficient capability to resolve the strong velocity gradients encountered within the interrogation area, can be overcome by using OF[7], and thus OF-based methods can be considered as a powerful alternative for investigation of particle image sequences.

Mathematically, the concept of optical flow is based on a continuous variational formulation for global estimation[8] of a vector field over the whole image. More precisely, the definition of optical flow is as follows [HS81]. Given two images, the optical flow (OF) estimates the apparent motion of brightness patterns in those images. The main idea is to minimize a global cost function representing the rate of change of image brightness E:

$$\zeta_b = E_x u + E_y v + E_t, \tag{2.10}$$

where u and v are the components of the OF vector, i.e., the unknown velocity components; E_x and E_y are the spatial derivatives of brightness, and E_t is the derivative of brightness with respect to time. Since the original problem is ill-posed, an additional

[7] Not all of the many OF implementations [BFB94] are well suited for DPIV applications [QPK98].

[8] There also exist methods relying upon the local spatial and spatiotemporal brightness constancy assumption of OF [LK81, BG88]. However, the OF scheme by [HS81] is considered to be a better choice for DPIV applications [HHN$^+$07].

constraint or *regularizer* (cf. Equation 2.7) assuming smooth movements between adjacent cells is introduced:

$$\zeta_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2. \tag{2.11}$$

Accompanied with such a regularizer the minimization problem is posed as follows:

$$\zeta^2 = \iint \left(\alpha^2 \zeta_c^2 \;+\; \zeta_b^2\right)\, dx\; dy, \tag{2.12}$$

where α is used to weight the smoothing term in such a way as to compensate for the noise in the image brightness measurements in real-life experiments. By using the calculus of variations, Equation 2.12 can be transformed into a system of two elliptic partial differential equations:

$$\begin{aligned} \alpha\,\Delta u \;-\; E_x\,(E_x u \;+\; E_y v \;+ E_t) &\;=\; 0, \\ \alpha\,\Delta v \;-\; E_y\,(E_x u \;+\; E_y v \;+ E_t) &\;=\; 0. \end{aligned} \tag{2.13}$$

The partial differential equations are finally transformed into a set of finite difference equations using any suitable discretization scheme. The resulting sparse linear system can then be solved for the velocity (u, v) using standard iterative methods.

As it was shown by many authors [RKSN05, CHA$^+$06, NOBT05], the OF-based method offers a range of advantages over their CC-based counterparts: high resolution (velocity vectors are given in each pixel of the original image) and high accuracy of the reconstructed velocity fields; robustness to noise and thus increased value in application to real-world data sets; complete elimination of outliers, and in the case of multigrid implementation, high-speed performance [MP98, RKSN05]. Moreover, it was reported that in contrast to CC-based techniques, OF-based approaches are also able to provide reliable results for highly non-rigid passive scalar images (see e.g., [JCT01]), where no separated particles can be distinguished [RS07].

However, similar to all reconstruction methods, the standard OF approaches are prone to numerical errors and have some weaknesses, especially when applied to experimental PIV images. Some of these disadvantages, such as the sensitivity to the changes in brightness pattern and the reduction of accuracy of the reconstruction results in presence of large particle displacements, are common to most of the standard reconstruction algorithms, regardless of the underlying algorithm, CC and OF.

Many heuristical approaches have been proposed in order to eliminate the reported artifacts. Due to its global nature, the OF method does not undergo the problems linked to the discrete and local nature of the CC-based methods such as in-plane loss of pairing[9], or averaging the velocity vectors over the interrogation window. Taking into account that the OF techniques originally have been devised to perform the analysis of quasi-rigid objects, most of the improvements aimed to make the OF approaches more suitable for fluid flow applications. In this way, it has been proven that the natural choice for an additional smoothing constraint for the OF problem (cf. Equation 2.11) is a function which incorporates all the relevant information about the flow into the reconstruction process. Such information-embedding techniques belong to the class of so called *physics-based* or more general *model-based* approaches and are considered in more detail in Section 2.3.4.

2.3.4 Hybrid and Model-Based Techniques

Looking through the vast body of published literature on the subject, one can be astonished how many authors were independently coming up with similar ideas such that only implementation differences or the style of description of the methods allow to distinguish them. It is interesting to see how the authors from different communities, image processing and experimental fluid mechanics, were trying to resolve the same problem and were looking at it from different points of view. The results were given for the same image sequences, however, the way how the methods were described were different. If the image processing researchers (mathematicians) are rather focused on mathematical proofs of their methods and their papers are densely seeded with lengthy formulae, experimental fluid mechanics researchers (practical physicists) are mainly interested in physical value of the methods in terms of reliability and physical insight to be gained from the results of the reconstruction.

The terminology used in the literature for the description of the methods themselves and their validation were also different and, surprisingly, the authors from one community were referencing only the papers from their community, almost ignoring the counterparts developed at the same time in the other community. Lately, however, the mutually beneficial tendency of exchanging and merging the methods between the two worlds can be clearly observed. Perhaps the advent of the Internet as a powerful tool for researchers played a crucial role in establishing this tendency. Nowadays, typing a few keywords in the web search machines allows to find in a few seconds dozens of references, regardless of the date of publication, community, and country these publications were issued from.

[9] Note that due to the presence of particles leaving the image domain, on the borders of the images the problem is still unsolved.

Taking into account the number of different approaches, we are not attempting to review all of them here. However, we will roughly classify the most successful algorithms and briefly outline them in this section. Since our objective is to develop a new model-based reconstruction method, it is important to know, what has been done already and which direction should be followed in order to create a more efficient algorithm. It is worth mentioning that among the existing model-based approaches, there exist implementations for both basis algorithms: CC and OF.

Class 1: Coarse-to-Fine OF-Based Approaches with Physics-Based Constraints

At the core of efficient implementation of this class of algorithms is the universal framework for hierarchical model-based motion estimation, proposed by Bergen et al. [BAHH92]. This framework draws on a *coarse-to-fine refinement strategy* combining a *global model* that constrains the overall structure of the estimated motion and a *local model* that is used in the estimation process. Being defined in general terms, hierarchical framework has an advantage of compatibility with different motion representations from OF to parametric transformation, and various areas of application, from robotics to vehicle navigation, where the motion observation can be of interest. The ability to accurately recover large-scale motion is achieved due to an efficient computation of large displacements on a coarser level of the hierarchy. In addition, this framework provides a significant speedup.

A rough idea of the hierarchical approaches is to compute the large scale motion on a low resolution image and then use the higher resolution information to improve the accuracy of the estimated displacement field by incrementally estimating the small displacements. The basic stages of the algorithm are:

Hierarchy construction: A multi-resolution hierarchy of images is built from the original images. This can be implemented, by using, for example, the Laplacian pyramid [BA83].

Motion estimation: Given a current state estimate[10], the incremental displacements can be computed by minimizing a suitable error measure, which is built upon the model-based constraints and assumption of local brightness constancy within the image.

Image warping: The flow field, computed on the previous stage, is used to warp (deform) the image towards the current state. The warped image is used for the error computation in further iterations.

[10] An iteration within one level or a level itself can be considered as a state.

Coarse-to-fine refinement: The current motion estimates are propagated from one level/iteration to the next level/iteration, where they are used as initial estimates. The propagation from one level of the hierarchy to the next one is performed via a pyramid expansion operation. The transitional flow field between the levels is computed using a combination of global and local parameters.

The multi-resolution hierarchy of images is computed only once and is used on the subsequent stages of the algorithm. Motion estimation, image warping, and coarse-to-fine refinement procedures are iteratively repeated starting from the coarsest level up to the finest level of the hierarchy. Depending on the implementation of the motion estimation algorithm, the displacement field on each level is computed using any suitable iterative technique, e.g., the Gauss-Seidel relaxation method.

The algorithms, belonging to class 1 are built upon the above sketched hierarchical framework. The minimization problem in this case is stated in terms of OF devised by Horn and Schunck [HS81] and encompasses the continuity equation into the regularization term [Sut94, GP96, CMP02, CHA+06, HHN+07] or simply uses a modified continuity equation instead of the brightness constancy constraint [WALL97, NIN+03]. Furthermore, some authors include the residuals of the Navier-Stokes equations as additional physics-based constraints into the minimization function [NIN+03]. Several approaches also take into account the physical boundaries of the flow, such as obstacles within the flow domain, and incorporate them into the minimization process [WALL97, NIN+03].

Class 2: Alternative Approaches for Solution of Joint Minimization Problems

Similarly to class 1 algorithms, the reconstruction problem is stated as an inverse problem with embedded physics-based constraints. Then this problem is wrapped into a more general joint problem described in terms of systems theory and solved using the method from a specific branch of this theory.

In this light, Okuno et al. [OYN00] considered the whole reconstruction process as a *dynamical system* (cf. the definition of the flow field in Section 2.1.3) functioning according to the continuity equation and the Navier-Stokes equations. From this point of view, the reconstruction problem is transformed into a parametrization problem of a non-linear dynamical system, where the estimated image is constantly modified according to the laws governing this system and compared to the observed image. The iterative process is continued until the difference between the observed and the estimated images computed via cross-correlation is sufficiently small. The corresponding optimization problem is solved using methods from evolutionary programming.

Another, non-iterative approach was suggested by Kimura et al. [KSK$^+$02]. The reconstruction process is considered as an artificial neural network, or *adaptive system*, obeying the continuity equation. The artificial neural network is added as an additional constraint into the minimization problem, which is finally solved using a gradient-based method such as OF [HS81]. Note that in contrast to the approach suggested by Okuno et al. [OYN00], this method does not perform an iterative image modification.

It is interesting to point out that both evolutionary programming and artificial neural networks are rather general-purpose approaches, and thus exploiting them complicates the analysis of stability and reliability of these algorithms. Recently, Ruhnau and Schnörr [RS07] proposed the *Optical Stokes Flow*, a variational approach using the time-independent Stokes equation as prior knowledge. This approach considers the motion characteristics of a viscous fluid in the optical flow estimation in that it computes external and body forces acting on the fluid in such a way as to yield a velocity field equal to the optical flow field. By integrating this restriction into the coupled optical flow system of equations a constrained minimization problem is derived. The latter problem is solved using the iterative coarse-to-fine approach [BAHH92, RKSN05].

Class 3: CC-Based Approaches with Iterative Window Shift and/or Deformation

The approaches of this class are tailored to cure one of the main drawbacks of the conventional CC-based approaches, their inability to recognize the strong velocity gradients within the interrogation area and, as a consequence, reduction of accuracy of the estimated vector field in this area. As it was observed by Huang et al. [HFW93], the image pattern within the interrogation area tends to continuously change its position and shape in the spatiotemporal domain. The authors suggest to reproduce the corresponding deformation within the correlated image pair and thus to increase the correlation between the two interrogation windows. The unknown displacement distribution is estimated by means of a Taylor series. The choice of the order of the series expansion leads to a compromise between the accuracy and the computational cost of the reconstruction [Sca02]. According to the choice of the truncation order, all the algorithms of class 3 can be further categorized as follows:

Zero-order: a uniform window deformation is associated with this subclass of algorithms. There are two variations of these methods: techniques with simple window extent [KA90, Adr91, FS97, Wes00, GLS01] and techniques with a discrete [Wes97, WM00] or fractional window offset [Sca02].

First-order: linear image deformation using the current estimated displacements over the interrogation area is performed on each stage of this group of algorithms [HFW93, JJDAF95, SR99].

Higher-order: in this case a non-linear image transformation is defined on the basis of the governing flow equations [TD95, FD00, RSY01, NOBT05].

Most of the algorithms of this class are implemented using the coarse-to-fine strategy [BAHH92, SR99]. Some of them suggest to use a more robust technique, e.g., phase-correlation, for a large scale motion estimation at the coarsest level of the hierarchy [TMKK05]. In addition, an appropriate choice of image interpolation, displacement prediction-correction and correlation peak-fit schemes is of crucial importance [Sca02]. Moreover, despite of the complexity of class 3 algorithms, the outliers can still be present in the reconstructed vector field and on intermediate stages. Thus, it is strongly recommended to use this class of algorithms in combination with the outlier removal techniques applied after each iteration [Sca02].

Class 4: Advanced Outlier Removal Techniques for CC-Based Methods

The presence of outliers in the reconstructed vector field is inherent to all of the CC-based approaches, no matter how sophisticated they are (cf. class 3). Thus detection of these spurious vectors and their efficient removal is a crucial issue for CC-based techniques. Besides the various median-based [Wes94, RWK01, WS05] and interpolation-based [YJW04] outlier-handling techniques, various advanced methods relying upon the prior knowledge about the underlying flow have been proposed in the literature.

Several authors suggest to build a new cost function based on the governing flow equations, initialize the optimization framework with original results of the CC-based algorithm and solve the corresponding problem by finding the displacement field which conforms to the specified flow equations [ONI97, KYKI98].

Taking into account the randomized nature of outliers, many authors suggested to use an artificial neural network to solve the corresponding physically constrained optimization problem [GP97, Lab01, LJL03, SBB04]. Recently, in context of microfluidics, Petermeier et al. [PKD$^+$07] suggested to embed the underlying flow equations into the so called functional nodes of the artificial neural network. Additionally, these authors have proven that in application to micro-flows, the simplified flow equations based on the Taylor hypothesis (Taylor 1938) can be efficiently incorporated into the reconstruction process.

Chapter 3

Model-Based Flow Reconstruction

Despite the fact that the experimental arrangements for the PIV measurements are quite similar in most applications, the experimental setup must be carefully adjusted according to the application-specific requirements. It is especially important when the measurements are performed in flows induced by living organisms. In this case, any artificial intrusion into the natural environment of the organisms, which may affect their behavior, must be avoided. Therefore, taking into account these application-driven restrictions imposed on the experimental setup in combination with general measurement imperfections and discretization error of the imaging process, the problem of the reconstruction of vector fields from experimental image sequences becomes underconstrained. Therefore, it is nearly impossible to reconstruct admissible flow fields from such sequences using standard methods. The recently proposed *model-based approaches* have demonstrated promising results in resolving the aforementioned problems. These methods incorporate *a priori* information about flow as additional constraints into intrinsically ill-posed problem, and thus, allow for reliable reconstruction of flow fields from experimental image sequences.

The reconstruction method we have developed was inspired by non-parametric image registration techniques used in medical imaging to find the correspondence between images of the same anatomical structure taken under different conditions (e.g., different relative camera-patient position, different methods of acquisition, etc.). The reconstruction results demonstrated by Modersitzki [Mod04] for the case of fluid registration were especially valuable for us to make a choice of direction for algorithm development. The argumentation for this reconstruction method is technically sound and, furthermore, it is based on the laws of physics, which the investigated image sequences have to obey. Like true experimentalists, we decided to implement the method and check whether its application to PIV image sequences makes sense at all, before delving into the abyss of theoretical investigations. The preliminary results were quite promising, though many conceptual and implementation issues needed further elaboration. Therefore, the direction for further research has been approved.

This chapter is completely devoted to the developed model-based reconstruction approach. The theoretical aspects involved in this method as well as practical details and implementation issues are thoroughly described therein.

3.1 Method: Compilation and Assembly

In terms of image registration (IR), the goal of the model-based reconstruction algorithm is the computation of a non-parametric transformation describing the motion of particle tracers seeded in the flow given as a sequence of instantaneous flow images. We exploit a priori knowledge about the physical model of the flow to make the transformation consistent with the laws of physics. In the tradition of advanced class 3 algorithms described in terms of control theory (cf. class 2 in Section 2.3.4), the proposed method can be seen as an *automated feedback control system*, which performs the model-based global filter operation that corrects a predicted flow field appropriately. A coarse functional scheme of such a control system for flow reconstruction is presented in Figure 3.1 (cf. a black-box version of this scheme in Figure 2.7).

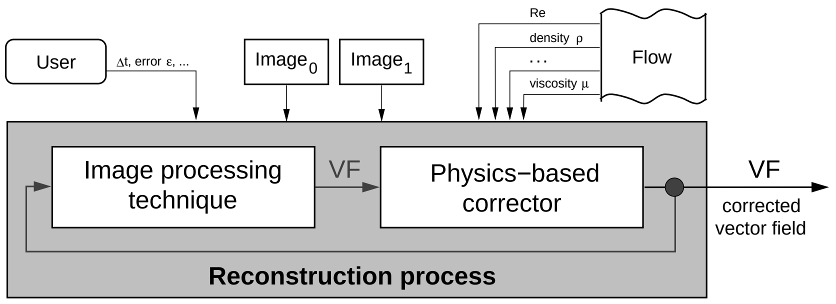


Figure 3.1: *Coarse functional scheme of the model-based reconstruction algorithm.*

Thus, our reconstruction algorithm represents an interaction between the three main components: input information, including the parameters of experimental setup and user requests; image processing module, which estimates the difference between the images at the current stage of the algorithm; and physics-based correction module, which corrects the vector field in each iteration.

In order to support the reconstruction for a wider spectrum of flows, which in general are obeying the Navier-Stokes equations (NSE), a complete NSE-solver is included into the physics-based correction module. It is worth mentioning that some flows, such as *creeping flow*, can be described using the simplified version of NSE, the *Stokes equations*. Since the choice of the model is a crucial point for the model-based algorithms, more details about the chosen flow models for the investigated test cases in Chapter 4 are provided in Section 3.1.3.

3.1.1 Algorithm Overview

The developed algorithm inherits many ideas from class 2 and class 3 algorithms (cf. Section 2.3.4). In addition, the most valuable suggestions for each of these classes were taken into account. Briefly summarizing, the suggested method extends related model-based approaches in the following ways:

- Rather than penalizing optical flow (OF) with a model-based regularizer, a flow field is first predicted by classical OF [HS81] and then corrected by applying a global filter operation.
- The model-based filter implements a numerical solver for the NSE, and it is thus suitable for the class of viscous, incompressible fluids. It simulates the effect of external forces applied to the fluid, with the direction of these forces being given by the predicted velocity field.
- In contrast to other approaches, the NSE-solver does not perform a simulation of the fluid motion over time, but it corrects the instantaneous flow field in one particular time step. The process is iteratively performed until the average of the predicted displacements becomes smaller than a user-defined threshold. More details on this issue are given in Section 3.2.6.
- Boundaries and boundary conditions are considered in the model-based correction step. In this way it is possible to simulate the interaction between the fluid and the obstacles in the flow (e.g., microorganisms in a test case investigated in Section 4.3).
- All stages of the iterative process have been implemented on programmable graphics hardware. For the first time ever it is thus possible to interactively control specific flow parameters as well as experimental settings within the reconstruction process.

A block diagram of the algorithm is presented in Figure 3.2. In order to better juxtapose the functional scheme of the algorithm and its block diagram, the functional modules of the algorithm are shaded with grey rectangles.

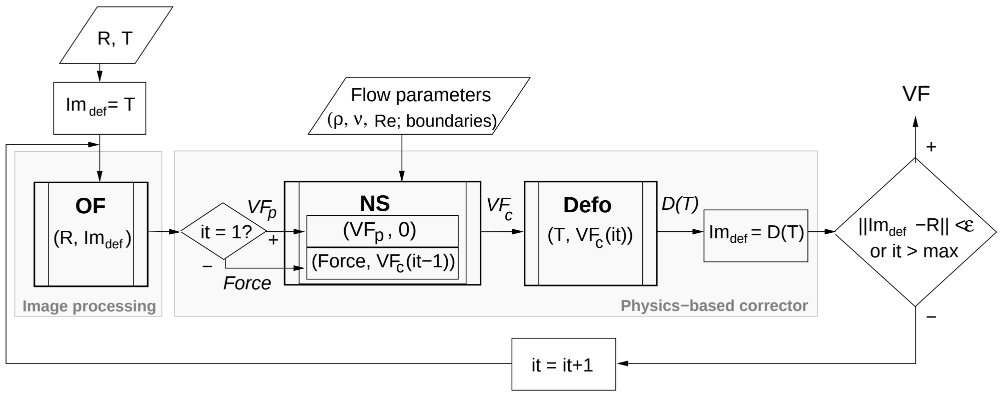


Figure 3.2: *Block diagram of the proposed predictor-corrector scheme.*

First, the displacement field $\mathtt{VF_p}$ is predicted using OF. Next, the NSE-solver is applied to compute the corrected field. In the context of PIV, the displacement simulates the movement of particles from their initial positions to the destinations in the reference image R. Thus, the resulting vector field $\mathtt{VF_c}$ is used to displace pixel values in the template image T. This process is iteratively repeated on the deformed template D(T) and the reference image R. The iteration stops when image T finally matches image R, or at least when both images are close to each other with respect to any suitable error measure. An illustration of three iterations of the reconstruction process is presented in Figure 3.3.

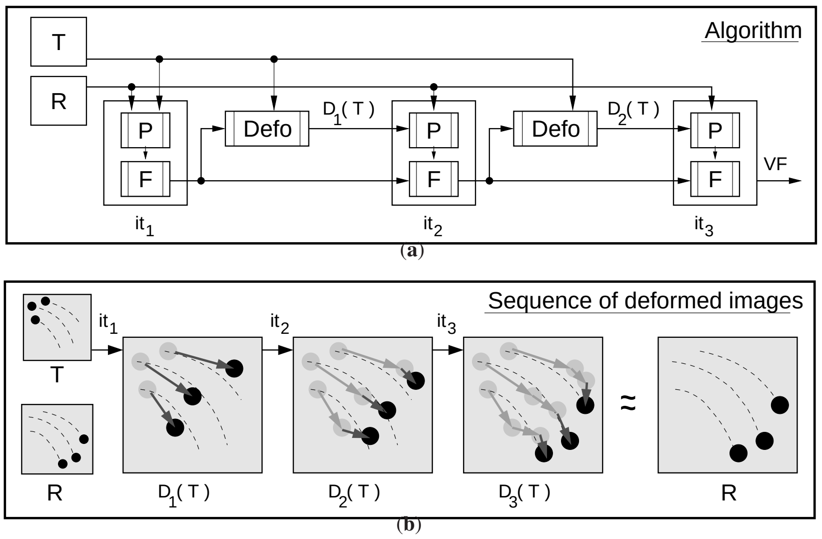


Figure 3.3: *Illustration of the reconstruction process: (**a**) Functional scheme:* Module P *reads images* T *and* R *and computes (predicts) the displacement field.* Module F *filters the predicted displacement field.* Module Defo *deforms* T *towards* R *using the filtered displacement field. (**b**) Iterative deformation of image* T *towards image* R.

The result of each iteration is the predicted displacement field $\mathtt{VF_p}$, the corrected field $\mathtt{VF_c}$, and the deformed image $\mathtt{D(T)}$. The output after all iterations is the corrected flow field $\mathtt{VF}$, which complies with the assumed physical model of the flow. In the following we will describe the different parts of the proposed algorithm in more detail.

3.1.2 Choice of Flow Predictor

In order to make a choice of image processing technique to use on prediction step of our method, we took into account the following observations:

- In contrast to cross-correlation (CC) based techniques, there is no limit (up to the image resolution) on the spatial resolution of the estimated motion vector field. Thus, in each iteration we obtain a dense initial velocity field that can directly be used to compute the external force field in the NSE simulation at the same scale as the simulation grid, i.e., the pixel grid.

- The OF approach gives a smooth motion vector field even if the particle seeding density is low, which is the case in some applications, like biocompatible microscopic PIV systems. This, in turn, yields a smooth external force field, which is important to avoid numerical instabilities in the NSE simulation.

- In contrast to OF-based approaches, the CC-based methods assume that the velocity gradient of the flow within an interrogation window is negligible. This greatly affects the variation of the derived external force field and it results in slower convergence of the iterative predictor-corrector approach.

Guided by this strong argumentation on behalf of OF-based approaches, the OF technique was chosen as a basis for the predictor scheme of the proposed model-based algorithm.

3.1.3 Flow Models for Correction Step

The choice of the flow model for physics-based approaches plays a crucial role. Moreover, the implementation of the corrector module is directly based on a particular flow model. Therefore, in this section we provide a theoretical description of the general flow model, represented by the NSE and two flow models employed for the investigation of specific flow situations in Chapter 4.

General Flow Model: Navier-Stokes Equations

In general, laminar flows of viscous, incompressible fluids are described by the Navier-Stokes equations – the fundamental differential equations describing the dynamics of fluid flow [GDN98]. The dimensionless form of these equations in the 2D case is given by:

$$\frac{\partial u}{\partial t} = \frac{1}{Re}\nabla^2 u - \nabla \bullet (u\mathbf{V}) + g_x - \frac{\partial p}{\partial x}, \quad \text{momentum equation, } x \tag{3.1}$$

$$\frac{\partial v}{\partial t} = \frac{1}{Re}\nabla^2 v - \nabla \bullet (v\mathbf{V}) + g_y - \frac{\partial p}{\partial y}, \quad \text{momentum equation, } y \tag{3.2}$$

$$\operatorname{div}\mathbf{V} = \nabla \bullet \mathbf{V} = 0, \quad \text{continuity equation} \tag{3.3}$$

where u and v are the two components of the velocity vector $\mathbf{V} = (u,\, v)^{\mathrm{T}}$; t is the time variable; x, y are the components of the coordinate system assigned to the flow domain; g_x, g_y are the components of external force $\mathbf{G} = (g_x,\, g_y)^{\mathrm{T}}$ to be considered at every grid point; p is the pressure. The dimensionless quantity Re is called the Reynolds number and can be computed as follows:

$$Re \;=\; \rho U_\infty L/\mu, \tag{3.4}$$

where ρ is the density of the fluid, U_∞ and L are the characteristic velocity and length[1], and μ is the dynamic viscosity of the fluid.

The operators, gradient ∇ and Laplace ∇^2 (or Δ), are defined as follows:

$$\begin{aligned} \nabla u &= (\partial u/\partial x,\, \partial u/\partial y)^{\mathrm{T}}, \\ \nabla^2 u &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}. \end{aligned} \tag{3.5}$$

Thus, taking into account Equation 3.5 and unpacking the dot-product operator[2] in Equations 3.1-3.3 leads to:

$$\begin{aligned} \nabla \bullet (u\mathbf{V}) &= u\nabla \bullet \mathbf{V} + \mathbf{V} \bullet \nabla u = u\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + \mathbf{V} \bullet \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{bmatrix} = \\ &= u\frac{\partial u}{\partial x} + u\frac{\partial v}{\partial y} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \\ &= 2u\frac{\partial u}{\partial x} + \left(u\frac{\partial v}{\partial y} + v\frac{\partial u}{\partial y}\right) = \frac{\partial (u^2)}{\partial x} + \frac{\partial (uv)}{\partial y}, \\ \nabla \bullet (v\mathbf{V}) &= \frac{\partial (v^2)}{\partial y} + \frac{\partial (uv)}{\partial x}. \end{aligned} \tag{3.6}$$

[1] E.g., in a wind-tunnel, U_∞ and L could be the upstream velocity and the length of an obstacle in the flow.
[2] $\nabla \bullet \mathbf{V} = \partial u/\partial x + \partial v/\partial y$

As in the case of any partial differential equations, the treatment of boundary regions is of great importance for the solution of the NSE. The *boundary regions* are the regions on the border of the fluid domain or on the border of fluid-structure interface, where a particular *boundary conditions* should be specified. In general, there are five different types of boundary conditions [GDN98]:

- *No-slip:* No fluid penetrates the boundary and the fluid is at rest there. Example: all the solid bodies immersed into the fluid.
- *Free-slip:* No fluid penetrates the boundary. However, there are no frictional loses at the boundary. Example: the plane/line of symmetry for a symmetric flow situation.
- *In(out)-flow:* The velocities along the boundary are given. Example: the inlet region in a channel.
- *Periodic:* Occur for problems that are periodic in one coordinate direction.

For all the cases studied in Chapter 4 the *no-slip* boundary conditions are imposed.

Creeping Flow Model: Stokes Equations

According to the definition from Section 2.1.4, *creeping flow* (or *Stokes flow*) is a low speed flow, characterized by a low Reynolds number ($Re \leq 1$) and dominated by the viscous forces, pressure, and body forces [FP02]. Thus, the inertial terms in the NSE can be neglected resulting in a simplified version of momentum equations:

$$\begin{aligned} \nabla p &= \mu \Delta \mathbf{V} + \mathbf{G}, \\ \nabla \bullet \mathbf{V} &= 0. \end{aligned} \tag{3.7}$$

Note that Equation 3.7 represents a *steady* or *time-independent* flow model.

Stokes Flow Model with a Single Stokeslet

In particular, when the force field $\mathcal{G}$ is given by a single point force $\mathbf{G}$ or *stokeslet*, Equation 3.7 can be rewritten as follows [BO96]:

$$\begin{aligned} \nabla p &= \mu \Delta \mathbf{V} + \mathbf{G}\delta(\mathbf{p} - \mathbf{p_0}), \\ \nabla \bullet \mathbf{V} &= 0. \end{aligned} \tag{3.8}$$

where $\delta(\mathbf{p} - \mathbf{p_0})$ is the Dirac delta function and $\mathbf{p_0}$ is a position of a point force application.

The action of a single stokeslet results in the development of one or several viscous eddies, and thus initiates an interesting flow situation [LB81]. For example, when the stokeslet is originated at height h above a planar no-slip boundary and is oriented orthogonal to this boundary, the resulting stream line pattern corresponds to a single toroidal eddy (in 3D) around the stokeslet. As depicted in Figure 3.4, the axial cross-section of this toroidal eddy represents two axisymmetric vortices.

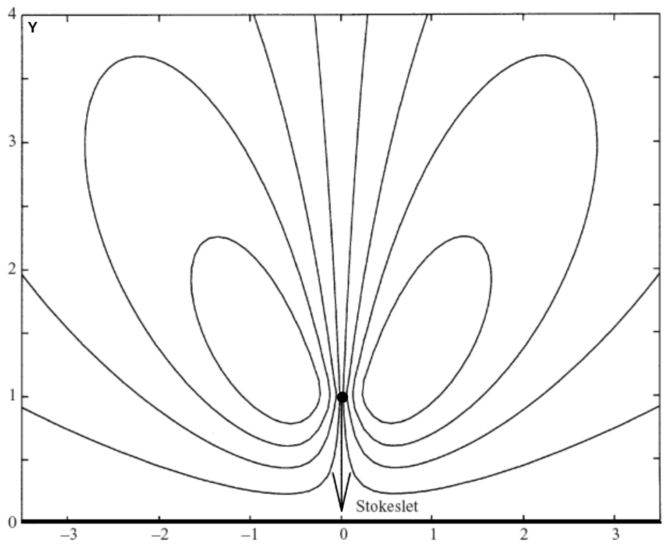


Figure 3.4: *Stream lines corresponding to a stokeslet at* $(0,\ 1)$ *above a plane boundary* ($y\ =\ 0$).

An analytic solution of Equation 3.8 is described by the Green's function and the corresponding stream function $\psi(r,\ z)$ is given by:

$$u_z = \frac{1}{r}\frac{\partial\psi}{\partial r}, \quad u_r \ = \ -\frac{1}{r}\frac{\partial\psi}{\partial z},$$
$$\psi(r,z) = \frac{\mathbf{G}}{8\pi\mu}\left[\frac{r^2}{(r^2+(z-h)^2)^{\frac{1}{2}}} - \frac{r^2}{(r^2+(z+h)^2)^{\frac{1}{2}}} - \frac{2hr^2z}{(r^2+(z+h)^2)^{\frac{3}{2}}}\right], \quad (3.9)$$

where r and z are the radial and axial coordinates in a cylindrical polar coordinate system [AB78].

3.2 Algorithm Implementation

This section provides a detailed description of each part of the proposed model-based method. In particular, the implementation issues related to mapping the employed numerical algorithms onto graphics hardware are discussed.

3.2.1 Motivation for Performing Computations on Graphics Hardware

Over the last few years, graphics processing units (GPUs) have evolved towards highly parallel and fully programmable computing architectures, which are key to high performance and cost-effective super-computing. On current GPUs parallel processing units can be accessed via high-level programming languages. In addition to computational functionality these units provide an efficient interface to local graphics memory. This allows for high-performance read-write access to data that are stored on the GPU, and in particular it provides an efficient means for the communication of intermediate results in iterative algorithms.

In recent years, one direction of research has led towards the implementation of general techniques of numerical computing on such processors. These developments have shown that the GPU can significantly outperform CPU implementations for both compute- and memory-intensive applications. A comprehensive overview and collection of references in this particular area can be found in [OLG$^+$07].

Since both the prediction and the model-based correction steps of the proposed reconstruction method involve complex numerical algorithms, it is important that these methods are realized as efficiently as possible (cf. Figure 2.7). Therefore, we implement all the stages of the reconstruction process on programmable GPUs.

For the solution of linear systems of equations, underlying the involved numerical methods, we employ a library for linear algebra on GPUs [KW03]. This library provides classes for vectors and matrices as well as standard operations on and between them, with all of these operations performed at full 32-bit floating-point precision. In particular, this framework allows for the efficient representation of sparse matrices on the GPU, including matrix-matrix and matrix-vector operations. Matrices as well as vectors are stored as 2D memory objects (textures) in a compact format on the GPU. We use this framework for the numerical solution of the NSE, and we describe a novel implementation of a multigrid approach for solving the OF equations in Section 3.2.3.

3.2.2 Flow Prediction

The GPU-implementation of the OF algorithm takes advantage of the special structure of the OF system matrix. By using a standard finite difference discretization the OF differential equation, Equation 2.13, is transformed into the following set of linear equations:

$$E_x^2 u_{i,j} + E_x E_y v_{i,j} - \alpha(-4u_{i,j} + u_{i-1,j} + u_{i,j-1} + u_{i+1,j} + u_{i,j+1}) = -E_x E_t \quad (3.10)$$

$$E_x E_y u_{i,j} + E_y^2 v_{i,j} - \alpha(-4v_{i,j} + v_{i-1,j} + v_{i,j-1} + v_{i+1,j} + v_{i,j+1}) = -E_y E_t. \quad (3.11)$$

The partial spatiotemporal derivatives E_x, E_y, and E_t at pixel $[i, j]$ are approximated using central differences as follows [HS81]:

$$\begin{aligned} E_x &\approx \tfrac{1}{4}\left(E_{i+1,j,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i,j+1,k}\right. \\ &\quad \left. + E_{i+1,j,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i,j+1,k+1}\right), \\ E_y &\approx \tfrac{1}{4}\left(E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k}\right. \\ &\quad \left. + E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}\right), \\ E_t &\approx \tfrac{1}{4}\left(E_{i,j,k+1} - E_{i,j,k} + E_{i+1,j,k+1} - E_{i+1,j,k}\right. \\ &\quad \left. + E_{i,j+1,k+1} - E_{i,j+1,k} + E_{i+1,j+1,k+1} - E_{i+1,j+1,k}\right), \end{aligned} \tag{3.12}$$

where i, j, and k are the indices along the axes of the spatiotemporal cuboid build from the image pairs: i, j are the spatial axes within the image plane and k is the temporal axis (see Figure 3.5 (a)).

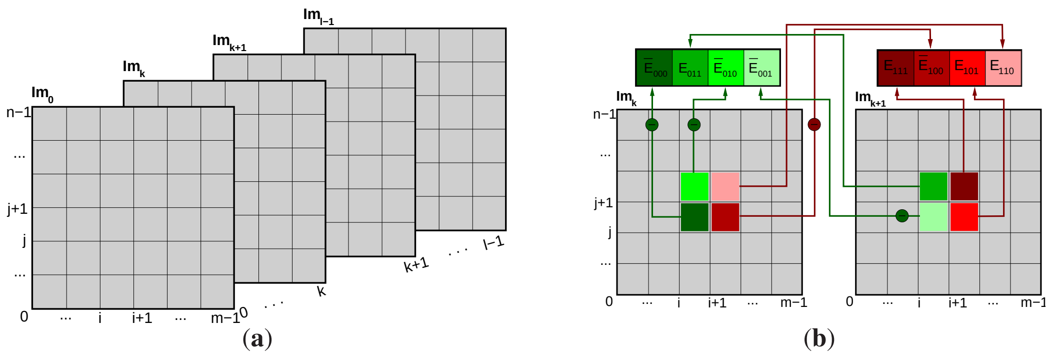


Figure 3.5: *Estimation of partial derivatives: (**a**) spatiotemporal cuboid; (**b**) filling up the auxiliary vectors* $\overline{\mathtt{E}}_{000}\mathtt{E}_{011}\overline{\mathtt{E}}_{010}\overline{\mathtt{E}}_{001}$ *and* $\mathtt{E}_{111}\overline{\mathtt{E}}_{100}\mathtt{E}_{101}\mathtt{E}_{110}$.

Taking into account that the GPU performs operations on all components of a 4D-vector simultaneously and swizzling of components is not counted as an operation [HLS], the partial derivatives for each pixel can be approximated in only four HLSL[3] instructions:

$$\begin{aligned} \mathtt{float4\ diff} &= \mathtt{E}_{111}\overline{\mathtt{E}}_{100}\mathtt{E}_{101}\mathtt{E}_{110} + \overline{\mathtt{E}}_{000}\mathtt{E}_{011}\overline{\mathtt{E}}_{010}\overline{\mathtt{E}}_{001}; \\ \mathtt{float3\ E_xE_yE_t} &= \mathtt{diff.xxx} + \mathtt{diff.wwy}; \\ \mathtt{E_xE_yE_t} &+= \mathtt{diff.zyz}; \\ \mathtt{E_xE_yE_t} &-= \mathtt{diff.yzw}; \end{aligned}$$

where $\overline{\mathtt{E}}_{000}\mathtt{E}_{011}\overline{\mathtt{E}}_{010}\overline{\mathtt{E}}_{001}$ and $\mathtt{E}_{111}\overline{\mathtt{E}}_{100}\mathtt{E}_{101}\mathtt{E}_{110}$ are 4-component vectors, filled up with the intensity values at position $[i, j]$ in the input image textures as depicted in Figure 3.5 (b).

[3] High Level Shading Language, a programming language for GPUs.

Combining and rearranging Equations 3.10-3.11 for all pixels in the image yields a system of linear equations. Since the number of unknown velocity vectors is equal to $m \times n$, the number of pixels in the image, and each velocity vector has two components, the corresponding system matrix is a sparse band-diagonal matrix of size $2mn \times 2mn$. An example of such a band-diagonal matrix for an image of size 3×3 is depicted in Figure 3.6. We denote a combined column of velocity vector components $(u_{00}, ..., u_{mn}, |v_{00}, ..., v_{mn})$ as UV.

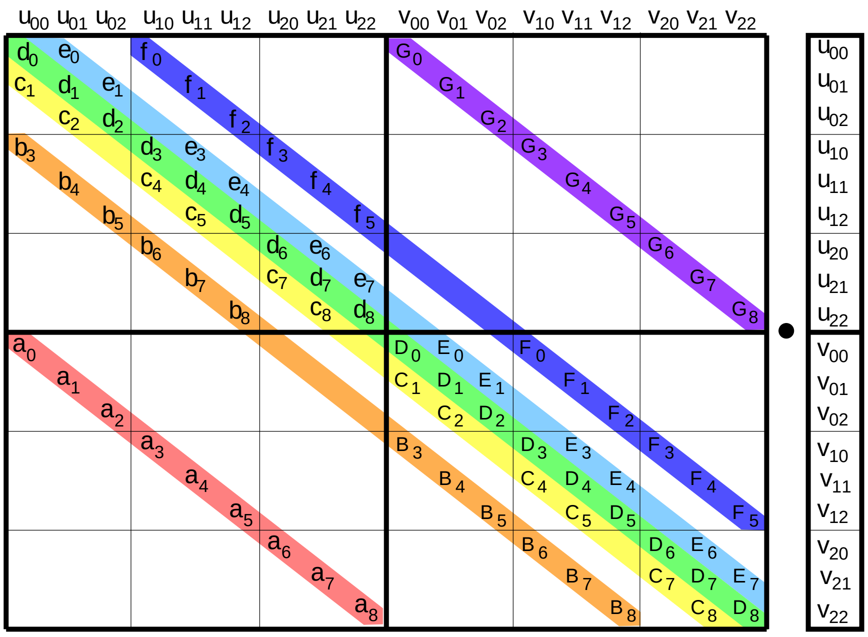
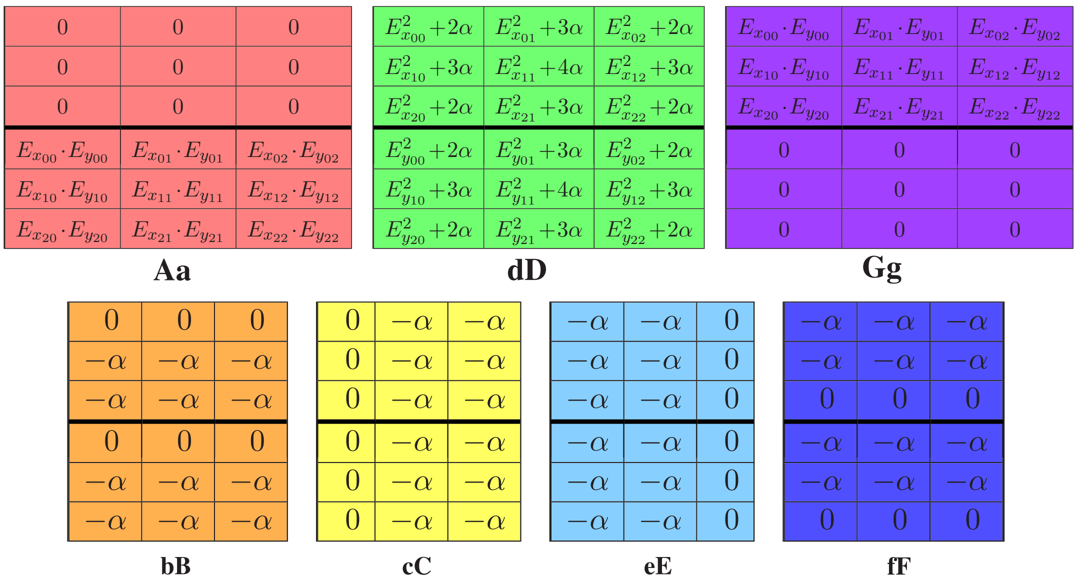

u00 u01 u02 u10 u11 u12 u20 u21 u22 v00 v01 v02 v10 v11 v12 v20 v21 v22
d0 e0 f0 c1 d1 e1 f1 c2 d2 f2 b3 d3 e3 f3 b4 c4 d4 e4 f4 b5 c5 d5 f5 b6 d6 e6 b7 c7 d7 e7 b8 c8 d8
G0 G1 G2 G3 G4 G5 G6 G7 G8
a0 a1 a2 a3 a4 a5 a6 a7 a8
D0 E0 F0 C1 D1 E1 F1 C2 D2 F2 B3 D3 E3 F3 B4 C4 D4 E4 F4 B5 C5 D5 F5 B6 D6 E6 B7 C7 D7 E7 B8 C8 D8
u00 u01 u02 u10 u11 u12 u20 u21 u22 v00 v01 v02 v10 v11 v12 v20 v21 v22

Figure 3.6: *Pattern of the left-hand side of the linear system of equations for OF computation: example for* 3×3 *images.*

In the GPU-library special classes are available to handle such sparse band-diagonal matrices. As demonstrated in Figure 3.6, the OF system matrix has only seven filled bands. Thus, this matrix can be efficiently stored into seven textures of resolution $2mn$. While computing the derivatives on the borders of the image, the mirror boundary conditions are imposed. An example of *band-textures* for a 3×3 image is depicted in Figure 3.7.

After all the necessary textures including the one for the right-hand side are filled up, the corresponding system of linear equations can be solved for the unknown velocity field VF using standard iterative methods (e.g., the conjugate-gradient method) implemented within the GPU linear algebra framework. Due to the massive parallelism available on recent GPUs coupled with an efficient memory interface to local data objects, the matrix generation is efficiently performed on the GPU. This allows to avoid



Aa

0	0	0
0	0	0
0	0	0
$E_{x00}\cdot E_{y00}$	$E_{x01}\cdot E_{y01}$	$E_{x02}\cdot E_{y02}$
$E_{x10}\cdot E_{y10}$	$E_{x11}\cdot E_{y11}$	$E_{x12}\cdot E_{y12}$
$E_{x20}\cdot E_{y20}$	$E_{x21}\cdot E_{y21}$	$E_{x22}\cdot E_{y22}$

dD

$E^2_{x00}+2\alpha$	$E^2_{x01}+3\alpha$	$E^2_{x02}+2\alpha$
$E^2_{x10}+3\alpha$	$E^2_{x11}+4\alpha$	$E^2_{x12}+3\alpha$
$E^2_{x20}+2\alpha$	$E^2_{x21}+3\alpha$	$E^2_{x22}+2\alpha$
$E^2_{y00}+2\alpha$	$E^2_{y01}+3\alpha$	$E^2_{y02}+2\alpha$
$E^2_{y10}+3\alpha$	$E^2_{y11}+4\alpha$	$E^2_{y12}+3\alpha$
$E^2_{y20}+2\alpha$	$E^2_{y21}+3\alpha$	$E^2_{y22}+2\alpha$

Gg

$E_{x00}\cdot E_{y00}$	$E_{x01}\cdot E_{y01}$	$E_{x02}\cdot E_{y02}$
$E_{x10}\cdot E_{y10}$	$E_{x11}\cdot E_{y11}$	$E_{x12}\cdot E_{y12}$
$E_{x20}\cdot E_{y20}$	$E_{x21}\cdot E_{y21}$	$E_{x22}\cdot E_{y22}$
0	0	0
0	0	0
0	0	0

bB

0	0	0
$-\alpha$	$-\alpha$	$-\alpha$
$-\alpha$	$-\alpha$	$-\alpha$
0	0	0
$-\alpha$	$-\alpha$	$-\alpha$
$-\alpha$	$-\alpha$	$-\alpha$

cC

0	$-\alpha$	$-\alpha$
0	$-\alpha$	$-\alpha$
0	$-\alpha$	$-\alpha$
0	$-\alpha$	$-\alpha$
0	$-\alpha$	$-\alpha$
0	$-\alpha$	$-\alpha$

eE

$-\alpha$	$-\alpha$	0
$-\alpha$	$-\alpha$	0
$-\alpha$	$-\alpha$	0
$-\alpha$	$-\alpha$	0
$-\alpha$	$-\alpha$	0
$-\alpha$	$-\alpha$	0

fF

$-\alpha$	$-\alpha$	$-\alpha$
$-\alpha$	$-\alpha$	$-\alpha$
0	0	0
$-\alpha$	$-\alpha$	$-\alpha$
$-\alpha$	$-\alpha$	$-\alpha$
0	0	0

Figure 3.7: *Example of band-textures filling for* 3×3 *images.*

the costly transfer operations between the CPU and the GPU [KW03].

3.2.3 Multigrid Optical Flow

Since OF-based techniques tend to be rather slow due to the numerical complexity of the employed solvers, considerable effort has been put into the development of advanced numerical techniques like multigrid schemes in a number of research projects [MP98, KR03, RKSN05]. In addition to providing significant speedup, multigrid techniques allow for reliable reconstruction of large displacements [BAHH92, RKSN05]. In our paper [KGW$^+$08] underlying Part I, for the first time ever to our best knowledge, an efficient implementation of multigrid OF on GPUs was demonstrated.

The essential step in the multigrid method is a coarse grid correction. The equation to be solved on the finer grid is $\mathtt{K}^\mathtt{h}\,\mathtt{UV}^\mathtt{h} = \mathtt{f}^\mathtt{h}$. On a coarser grid of size H, the defect equation $\mathtt{K}^\mathtt{H}\,\mathtt{e}^\mathtt{H} = \mathtt{r}^\mathtt{H}$ is used instead of the fine grid defect equation $\mathtt{K}^\mathtt{h}\,\mathtt{e}^\mathtt{h} = \mathtt{r}^\mathtt{h}$. Here, $\mathtt{K}^\mathtt{h}$ and $\mathtt{K}^\mathtt{H}$ are the system matrices on the fine and the coarse grid, respectively. Variables $\mathtt{e}^\mathtt{h}$ and $\mathtt{e}^\mathtt{H}$ are the absolute errors between the exact solution and the approximate solution on either grid, and $\mathtt{r}^\mathtt{h}$ and $\mathtt{r}^\mathtt{H}$ are the residuals on these grids. To establish a relation between $\mathtt{e}^\mathtt{h}$ and $\mathtt{e}^\mathtt{H}$, and between $\mathtt{r}^\mathtt{h}$ and $\mathtt{r}^\mathtt{H}$, linear transfer operators $\mathtt{I}^\mathtt{H}_\mathtt{h}$ and $\mathtt{I}^\mathtt{h}_\mathtt{H}$ are employed. For a good introduction to the theory and applications of multigrid approaches let us refer to [Hac94, BHM00].

Given these linear operators as well as an initial approximation $\mathtt{UV}^\mathtt{h} = \overline{\mathtt{UV}}^\mathtt{h}$ of the

OF values for two images on a fine grid, a new approximation of $\mathtt{UV}^h$ can be computed as it is shown in Figure 3.8.

```
TwoGridCorrection($K^h$, $UV^h$, $f^h$) {
    1. $r^h = f^h - K^h UV^h$        // Compute residual
    2. $r^H = I_h^H r^h$             // Restrict residual
    3. $K^H e^H = r^H$               // Exact solution on coarse grid
    4. $e^h = I_H^h e^H$             // Transfer correction
    5. $UV^h = UV^h + e^h$           // Correction
}
```

Figure 3.8: *Two-grid correction scheme.*

This two-grid correction can be extended to a full multigrid V-cycle by applying the two-grid correction recursively to step 3: TwoGridCorrection(K^H, e^H, r^H). It is straightforward to implement such a V-cycle using matrix-vector operations as they are provided by the employed GPU-library. For the transfer operators we use 2D texture interpolation, which takes advantage of the regular grid structure underlying the OF computations.

In order to exploit the multigrid approach to its full potential, the Galerkin property[4] has to be enforced. This implies, that once the OF matrix has been updated, the multigrid hierarchy has to be rebuilt in the following way:

$$K^H = I_h^H K^h I_H^h.$$

The matrix layout for update operation on the highest level of the multigrid hierarchy is illustrated in Figure 3.9.

Note that entries in both the restriction matrix I_h^H and its transpose I_H^h result from the five-point stencil used to locally interpolate the values given on a regular grid structure. Consequently, these entries do not have to be stored explicitly, but they are used as constants in the update of K^H. In order to support update operations we have extended the GPU-library by a special routine for the multiplication of the banded system matrix with seven non-zero diagonals and the respective procedural matrices used to enforce the Galerkin property (cf. Figure 3.9). Taking into account that the structure of the matrix on each level does not change, the implementation only has to respect the increasing number of diagonals in the matrices on ever coarser levels.

[4] The Galerkin property ensures a consistent calculation on different levels of resolution, and thus it guarantees fast convergence of the multigrid scheme.

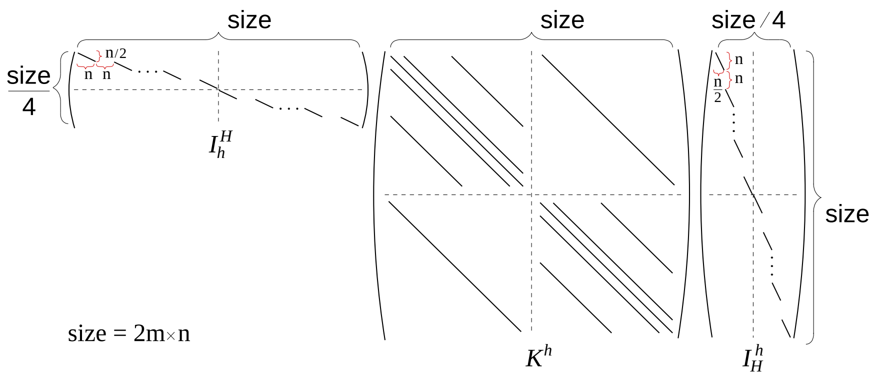


Figure 3.9: *Multigrid update operation: layout of the matrices used to enforce the Galerkin property.*

3.2.4 Vector Field Correction

To correct the predicted displacement field we employ a GPU-based simulation system [KW03]. In particular, we use this system to solve the incompressible NSE. Besides excellent performance, the system provides a variety of different parameters to model the fluid properties. To obtain the solution for the NSE (see Equations 3.1-3.3), we solve explicitly for the velocity $\mathbf{V} = (u, v)^T$ and implicitly for the pressure p. First, by ignoring the pressure term p, an intermediate velocity is computed using the time discretization of the momentum equations:

$$\begin{aligned} \bar{u}^{(it)} &:= u^{(it)} + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (u\,v)}{\partial y} + g_x \right], \\ \bar{v}^{(it)} &:= v^{(it)} + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial (u\,v)}{\partial x} - \frac{\partial (v^2)}{\partial y} + g_y \right]. \end{aligned} \tag{3.13}$$

The computation of the velocity and pressure fields is carried out on a uniform staggered 2D grid with the number of grid points equal to the size of the original image pair. On this grid, the Laplace operator $\nabla^2 u$ is discretized by means of central differences, and, as proposed by [Sta99], the advection part $\mathbf{V} \cdot \nabla \mathbf{V}$ is solved by tracing the velocity field backward in time. To make the resulting intermediate vector field free of divergence, the pressure is used as a correction term. Mass conservation of incompressible media leads to a Poisson equation for updating this pressure term:

$$\frac{\partial^2 p^{(it+1)}}{\partial x^2} + \frac{\partial^2 p^{(it+1)}}{\partial y^2} = \frac{1}{\delta t} \left(\frac{\partial \bar{u}^{(it)}}{\partial x} + \frac{\partial \bar{v}^{(it)}}{\partial y} \right). \tag{3.14}$$

Equation 3.14 is solved using a GPU-based conjugate-gradient method (CG). Finally, the computed pressure values are substituted into the following equations for velocity calculation:

$$\begin{aligned} u^{(it+1)} &= \bar{u}^{(it)} - \delta t \frac{\partial p^{(it+1)}}{\partial x}, \\ v^{(it+1)} &= \bar{v}^{(it)} - \delta t \frac{\partial p^{(it+1)}}{\partial y}. \end{aligned} \tag{3.15}$$

In each iteration it, pressure and velocity are computed starting with their values from the previous iteration. Initial conditions for the Poisson equation (3.14) in the *first iteration* can be set to zero or they can be guessed based on a priori knowledge about the experimental settings. In the current implementation, which does not perform a dynamic simulation of the fluid motion, the OF solution, i.e., the displacement field $\texttt{disp}_{\texttt{OF}}$, which is properly scaled according to the experimental setup, is used as initial condition for the velocity $\texttt{U}_{\texttt{NSE}}$ (in the *first iteration*).

The scaling takes into account the characteristic length L and velocity U_∞, the time step Δt between two subsequent image frames as recorded by the imaging system, and the pixel size pxsize in meters:

$$\mathbf{U_{NSE}} = \mathbf{VF_p}/U_\infty = (\mathbf{disp_{OF}} \cdot \mathrm{pxsize}) \,/\, (U_\infty \cdot \Delta t)\,, \tag{3.16}$$

where $\mathbf{U_{NSE}}$ is a dimensionless velocity used in the NSE; $\texttt{disp}_{\texttt{OF}}$ is the displacement field returned by the OF and measured in pixels px; pxsize is the scale-factor measured in $\mathrm{m/px}$; $\mathbf{VF_p}$ is the velocity from the OF solution $\texttt{VF}_{\texttt{p}}$ converted from px into metric units $\mathrm{m/s}$; U_∞ is the characteristic velocity measured in $\mathrm{m/s}$; and Δt is the time step between the subsequent images measured in s.

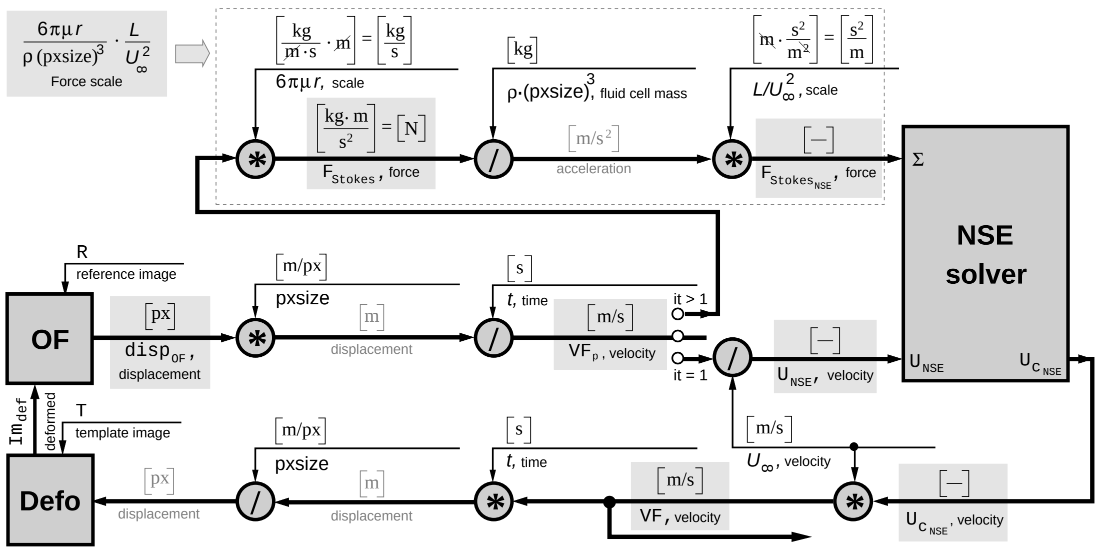
After the first correction step the velocity computed by the NSE-solver is an approximation of the flow field that carries particles in one time step from their initial positions to their destinations. In terms of image deformation, this means that the particles advected with this approximated flow field within the deformed image did not yet arrive at their expected positions in the reference image. In order to “move” the particles closer to their destinations, external forces have to be applied. These forces are in the opposite direction to the *drag forces*, which resist the movement of a solid object through a fluid. For viscous flows with $Re \leq 1$, which is the case, for example, for flows induced by microorganisms, the external force required to move a spherical particle of radius r through the fluid of viscosity μ at velocity $\mathbf{V}$ is defined according to Stokes’ law [Lig75] as:

$$\begin{aligned} \mathbf{F_{Stokes}} &= 6\,\pi\,\mu\,r\,\mathbf{V} \;\Rightarrow \\ &= 6\,\pi\,\mu\,r\,\mathbf{VF_p} = 6\;\pi\;\mu\;r\;\cdot(\mathbf{disp_{OF}} \cdot \mathrm{pxsize}/\Delta t)\,. \end{aligned} \tag{3.17}$$

Thus, on all subsequent iterations the OF solution is first scaled according to Equation 3.16 and Equation 3.17 and then used as external force field in the NSE-solver[5]:

$$\mathbf{F}_{\mathbf{Stokes}_{\mathbf{NSE}}} = \frac{\mathbf{F}_{\mathbf{Stokes}} \cdot L/U_\infty^2}{\Delta \mathrm{V}} = \frac{6\,\pi\,\mu\,r \cdot (\mathbf{disp}_{\mathbf{OF}} \cdot \mathrm{pxsize} \cdot L/\left(\Delta t \cdot U_\infty^2\right))}{\Delta \mathrm{V}}. \quad (3.18)$$

In order to get a global pictures of the transformation and transfer of physical and mathematical quantities between the modules of the reconstruction algorithm, it can be useful to draw a scheme similar to circuit diagrams in electronics [Hor89], taking into account multiple inputs/outputs and interpreting the modules and scaling operations as electrical elements. The diagram constructed according to these rules, including the physical interpretation, measurement units, and appropriate scaling factor for each variable, is presented in Figure 3.10.



$\frac{6\pi\mu r}{\rho\,(\mathrm{pxsize})^3} \cdot \frac{L}{U_\infty^2}$
Force scale
$\left[\frac{\mathrm{kg}}{\mathrm{m \cdot s}} \cdot \mathrm{m}\right] = \left[\frac{\mathrm{kg}}{\mathrm{s}}\right]$
$6\pi\mu r$, scale
$\left[\frac{\mathrm{kg \cdot m}}{\mathrm{s}^2}\right] = [\mathrm{N}]$
F_{Stokes}, force
[kg]
$\rho \cdot (\mathrm{pxsize})^3$, fluid cell mass
$[\mathrm{m/s}^2]$
acceleration
$\left[\mathrm{m} \cdot \frac{\mathrm{s}^2}{\mathrm{m}^2}\right] = \left[\frac{\mathrm{s}^2}{\mathrm{m}}\right]$
L/U_∞^2, scale
[–]
$F_{\mathrm{Stokes}_{\mathrm{NSE}}}$, force
Σ
NSE solver
U_{NSE}
$U_{C_{\mathrm{NSE}}}$
R
reference image
OF
[px]
$\mathrm{disp}_{\mathrm{OF}}$, displacement
[m/px]
pxsize
[m]
displacement
[s]
t, time
[m/s]
VF_p, velocity
it > 1
it = 1
[–]
U_{NSE}, velocity
$\mathrm{Im}_{\mathrm{def}}$
deformed
T
template image
Defo
[px]
displacement
[m/px]
pxsize
[m]
displacement
[s]
t, time
[m/s]
VF, velocity
[m/s]
U_∞, velocity
[–]
$U_{C_{\mathrm{NSE}}}$, velocity

Figure 3.10: *Circuit diagram of the reconstruction algorithm: transfer of variables between the modules of the reconstruction algorithm.*

The boundary conditions for the NSE-solver should be carefully adjusted according to a particular flow situation. In the case of obstacles immersed into the flow, they should be first extracted from the original image pairs (if parameters of the obstacles are not provided separately) and forwarded to the NSE-solver as a bit mask. The type of boundary conditions on the border of the interface fluid–obstacle is set to *no-slip* (cf. Section 3.1.3) for all flow situations considered in Chapter 4. The treatment of the boundary conditions is described in more detail in Sections 4.2-4.3 for each test case.

[5] In the NSE described by Equations 3.1-3.3, the force is a dimensionless quantity [GDN98].

3.2.5 Image Deformation

Prior to each iteration, the template image T is displaced towards the reference image R. Conceptually, in a forward mapping every pixel of the template is displaced along the respective velocity vector. Technically, in order to avoid invalid deformations in the transformed image as they typically arise in forward mapping approaches, backward mapping of the template image $\mathtt{Im}_\mathtt{t}$[6] using the inverse velocity field is employed. This means that the value of the pixel at position $[\mathtt{x} + \mathtt{disp}(\mathtt{x},\mathtt{y}).\mathtt{x},\ \mathtt{y} + \mathtt{disp}(\mathtt{x},\mathtt{y}).\mathtt{y}]$ in the template image $\mathtt{Im}_{\mathtt{t}+1}$ is written into the pixel at position $[\mathtt{x},\mathtt{y}]$ in the deformed image $\mathtt{Im}_\mathtt{def}$, where $\mathtt{disp}(\mathtt{x},\mathtt{y})$ corresponds to the displacement field computed in every iteration. The image deformation procedure for both forward and backward mapping is depicted in Figure 3.11. Since the deformation process is performed on the GPU as well, fetching respective values from the target positions can greatly benefit from hardware-assisted 2D texture interpolation.

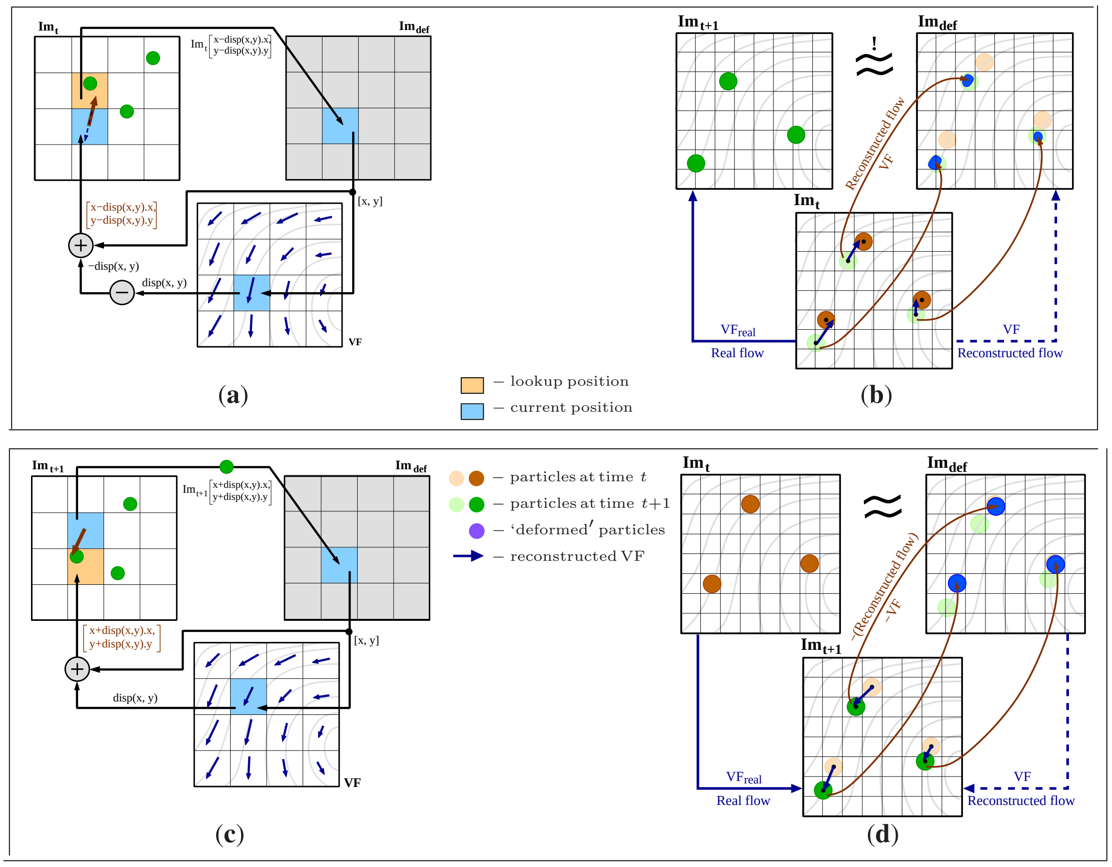


Figure 3.11: *Image deformation procedure (execution and example): (**a**,**b**) forward mapping and (**c**,**d**) backward mapping.*

[6] The reference image R and the template image T correspond to $\mathtt{Im}_\mathtt{t}$ and $\mathtt{Im}_{\mathtt{t}+1}$ with respect to acquisition time t.

As it is illustrated in Figure 3.2 and Figure 3.3, in the very first iteration of the algorithm the input for the OF computation is the original image pair. In subsequent iterations the predicted vector field $\mathtt{VF_p}$ is computed by the OF approach from the initial reference image $\mathtt{Im_t}$ and the deformed image $\mathtt{Im_{t+1}}$, $\mathtt{Im_{def}} = \mathtt{D(Im_{t+1})}$. As the magnitudes of subsequent displacements are gradually decreasing in general, so do the additional external forces considered by the NSE-solver in every correction step. To avoid the accumulation of roundoff errors and errors caused by the used interpolation scheme on a discrete pixel grid, the deformation is always performed on the initial template image using the accumulated displacement values, i. e. the accumulated velocity field. Furthermore, as suggested by Scarano [Sca02], advanced interpolation schemes can be adopted to achieve more accurate results for image deformation.

3.2.6 Stopping Criteria

The iterative correction process is stopped once the average magnitude of the vectors predicted in iteration it becomes smaller than a user-defined threshold[7]. This threshold is dependent on the flow scenario, and it is specified relative to the average magnitude of the predicted displacement field computed in the very first iteration. In all the experiments investigated in Chapter 4 the user-defined threshold was set to 5%.

To improve the precision of the error test, the relative displacements are computed per pixel and *then* averaged. As in every iteration the difference between the reference image and the template image with respect to a suitable distance measure is reduced, the predicted displacement field $\mathtt{VF_p^{(1)}}$ has the largest magnitudes in the first iteration compared to all subsequent iterations. Thus, it makes sense to:

1. Scale the magnitudes of $\mathtt{VF_p}^{(it)}$ with the corresponding magnitudes of $\mathtt{VF_p^{(1)}}$,
2. Calculate the average, which can be efficiently done on the GPU using so called *vector reduce* operation [KW03].

Mathematically, for $it > 1$ the preliminary computations for the error test operation can be summarized as follows:

$$\varepsilon_{it}\left(\mathtt{VF_p}^{(it)}\right) = \frac{1}{m \times n} \sum_{j=1}^{n} \sum_{i=1}^{m} \left\{ \left\| \mathtt{VF_p}^{(it)}[i,j] \right\|_2 \Big/ \left\| \mathtt{VF_p^{(1)}}[i,j] \right\|_2 \right\}. \tag{3.19}$$

[7] Iteration it is the current iteration. Since $\mathtt{T}$ is becoming closer to $\mathtt{R}$ in each iteration, the magnitudes of the vector field computed by the OF over these images are supposed to decrease with the number of iterations.

For noisy real-world image sequences, it can happen that the graph of the error $\varepsilon_{it}\left(\mathtt{VF_p}^{(it)}\right)$ contains a number of sharp peaks. In this case, in order to increase the robustness of the error test, we can compute the average error $\tilde{\varepsilon}_{it}$ over the three previous iterations. Note that on the first three iterations the computation procedure of the error is slightly different. The calculation of $\tilde{\varepsilon}_{it}$ for the first four iterations is illustrated in Figure 3.12.

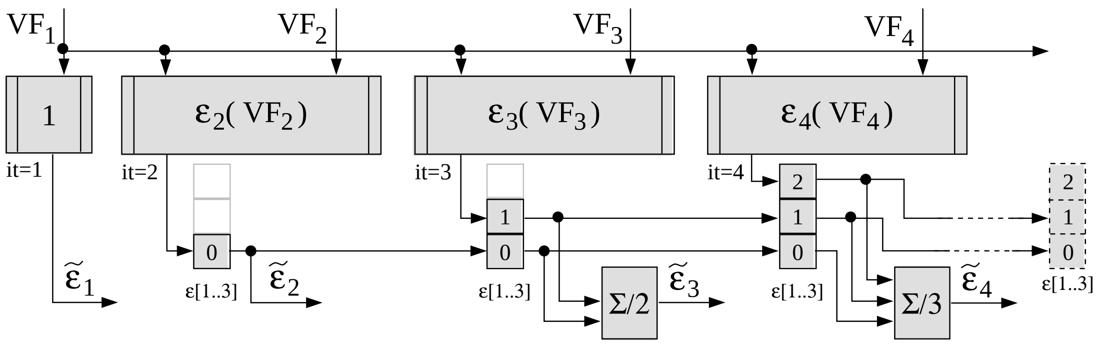


Figure 3.12: *Computation of $\tilde{\varepsilon}_{it}$, the average error over the three previous iterations: example for the first four iterations.*

It is worth mentioning that besides the error metric given by Equation 3.19 any other suitable error metric can be used to stop the iterative prediction-correction process. For example, instead of vector magnitudes of the predicted vector field, the angle difference between the spatially corresponding vectors of *corrected* vector fields $\mathtt{VF_c}^{(it)}$ and $\mathtt{VF_c}^{(it-1)}$ could be of interest.

Chapter 4

Validation of the Reconstruction Method

In this chapter we compare the proposed predictor-corrector algorithm to other approaches. We show that the proposed method can estimate motion vector fields at high quality and performance, and we demonstrate the advantages of the OF-based approaches over the CC-based approaches for the prediction of the displacement field in every iteration of the predictor-corrector scheme.

4.1 Introduction into Validation Techniques

The quality of the reconstruction algorithm is estimated based on the degree of accuracy of the reconstructed vector field, on the robustness of the algorithm to noise, and, in general, on its performance. As it was pointed out by Okamoto et al. [ONSK00], given as input only an image sequence obtained from a real-world experiment, there is no standard way to evaluate the accuracy of the reconstructed data. In order to enable the validation of the reconstruction method, the authors suggested to use *standard images* for particle-image velocimetry.

Standard images for validation along with the corresponding ground truth velocity fields are typically synthesized by numerical methods of computational fluid dynamics. In this way, the vector fields reconstructed by different techniques can be compared to the ground truth velocity field. The degree of mismatch between these vector fields is used as a quality measure of the reconstruction process.

There also exists another type of standard images, i.e., PIV images of well-studied flow situations obtained from real-world experiments, for which reliable estimates are available. Due to measurement imperfections and discretization errors, the real-world PIV images always contain a certain degree of noise. Therefore, applying the reconstruction method in these complicated conditions allows for testing out the robustness of the method to noise. However, for an arbitrary real-world experiment the flow under investigation is typically unknown, and thus no reliable estimate is available. In this case, neither precise validation nor quantitative comparison to the other methods can be done, and only a *qualitative* error analysis is possible [BT05].

In this sense, we have chosen two groups of image sequences for validation of our reconstruction algorithms: synthetic image sequences and real-world images obtained from experiments on living microorganisms. Within each group, we have tested PIV images with different particle image densities and different types of particle tracers (cf. Section 2.2.2). The synthetic image sequences include PIV images with classical particle image density (cf. Figure 2.4) and high particle image density, where no separate particles can be distinguished (*dye advection* or *passive scalar images*). For both synthetic sequences the ground truth velocity field is known. The real-world experimental image sequences were recorded using two different modes: classical PIV and PTV. Since the correct vector field is unknown for experimental image sequences, we have compared the results of the reconstruction from these images to the velocity vectors of several well-distinguished particles, which were computed by using the PT technique.

4.1.1 Error Measures

There exist many different ways to evaluate the quality of the reconstructed vector field if the ground truth velocity flow field is available. One commonly used technique is to evaluate the *per-pixel error map* using some error measure. Appropriate error measures for velocity vector fields are *absolute length difference* and *angular deviation* of the vectors without taking into account their lengths [BFB94, BK02].

Another technique is to calculate the *overall error* for the whole image using the per-pixel error map [GMN+98]. A number of different standard error measures, such as *average*, *root-mean-square*, *signal-to-noise ratio* [WDG97], etc., from statistics can be used for this purpose. Interestingly, some authors also use the L_1 norm for length error computation instead of the L_2 norm. Due to the fact, that in this case the error is computed along the vector components (catheti) and not along the vector itself (hypothenuse), the resulting error for the L_1 norm is lower. The error measures used for validation of our reconstruction method are listed in Table 4.1.

In Table 4.1, $\mathtt{V_{ref}}$ corresponds to the ground truth velocity vector field and $\mathtt{V}$ to the velocity vector field computed using the proposed model-based reconstruction method.

Table 4.1: *Error measures used for validation of the reconstruction algorithm.*

	Name	Formula	Units
Per pixel maps	Angular difference	$\varepsilon_{\text{ang}}[i,j] = \begin{cases} \arccos\left(\dfrac{\mathtt{V_{ref}}[i,j]}{\lVert\mathtt{V_{ref}}\rVert_2} \bullet \dfrac{\mathtt{V}[i,j]}{\lVert\mathtt{V}\rVert_2}\right), & \text{if}\{\mathtt{V_{ref}}[i,j]\neq\vec{0}, \mathtt{V}[i,j]\neq\vec{0}\} \\ 0, & \text{otherwise} \end{cases}$	o
	Absolute length difference	$\varepsilon_{\text{len}}[i,j] = \left\lVert \mathtt{V}_{\text{ref}}[i,j] - \mathtt{V}[i,j] \right\rVert_2 \cdot \Delta t/\text{pxsize}$	px
Overall errors	Average angular difference	$\tilde{\varepsilon}_{\text{ang}} = \dfrac{1}{m \times n} \sum_{j=1}^{m} \sum_{i=1}^{n} \varepsilon_{\text{ang}}[i,j]$	o
	Average length difference	$\tilde{\varepsilon}_{\text{len}} = \dfrac{1}{m \times n} \sum_{j=1}^{m} \sum_{i=1}^{n} \varepsilon_{\text{len}}[i,j]$	px
	Signal-to-noise ratio	$\text{SNR} = 10\log_{10}\left\{\dfrac{\text{RMS}_{\text{signal}}{}^2}{\text{RMS}_{\text{noise}}{}^2}\right\} = 10\log_{10}\left\{\dfrac{\text{RMS}_{\text{ref}}{}^2}{\text{RMS}_{\text{dif}}{}^2}\right\}$, where $\text{RMS}^2_{\text{ref}} = \dfrac{1}{m \times n} \sum_{j=1}^{m} \sum_{i=1}^{n} \left\lVert \mathtt{V_{ref}}[i,j] \right\rVert_2^2$ $\text{RMS}^2_{\text{dif}} = \text{MSE} = \dfrac{1}{m \times n} \sum_{j=1}^{m} \sum_{i=1}^{n} \left\lVert \mathtt{V_{ref}}[i,j] - \mathtt{V}[i,j] \right\rVert_2^2$	dB
	Peak signal-to-noise ratio	$\text{PSNR} = 10\log_{10}\left\{\dfrac{\max\limits_{i,j}\left(\left\lVert \mathtt{V_{ref}}[i,j] \right\rVert_2^2\right)}{\text{RMS}_{\text{dif}}{}^2}\right\}$	dB

RMS is the *root-mean-square* error and MSE is the *mean-square* error. Note that for convenience reasons, the length difference is measured for displacement vectors rather than for velocity vectors themselves.

4.1.2 Software Tools Used for Reconstruction

For comparative analysis of the accuracy of the reconstruction process provided by the proposed model-based algorithm, we estimate the velocity fields from synthetic and real-world image pairs using the following approaches:

CC:	multi-pass cross-correlation,
OF:	optical flow (OF),
CC+NS:	proposed model-based predictor-corrector with CC as predictor,
OF+NS:	proposed model-based predictor-corrector with OF as initial predictor.

In these comparisons a *multi-pass CC approach* (EDPIV of the National Center of Physical Acoustics at the University of Mississippi[1]) was used. An initial interrogation window size of 32×32 pixels, a final window size of 8×8, and a grid cell size of 2×2 pixels were selected for best quality. Advanced median filter and target vector techniques were used for the detection and correction of erroneous vectors. The OF approach we employed is based on the classical OF algorithm with first-order regularization [HS81]. The proposed predictor-corrector step is carried out several times to demonstrate the convergence behavior of our approach.

In order to extract the obstacles within the fluid domain from a given image, any suitable *segmentation algorithm* can be employed. In the investigated test cases, the obstacles are present in experimental image sequences, which depict a part of a real microbiological environment. The most appropriate segmentation tool in this case is a tool fine-tuned for extraction of biological structures from medical images. In our work we use an efficient segmentation method based on random walks [GSAW05].

4.2 Results of Reconstruction: Synthetic Image Pairs

In this section, we study synthetic image pairs of two different 2D flow scenarios, for which the velocity fields are known: a Navier-Stokes flow that was produced by numerical simulation of a swirling motion induced by user-defined external forces and the Lamb-Oseen viscous vortex flow [CW05].

4.2.1 Navier-Stokes Simulation (Dye Advection)

For this test case, the synthetic images were obtained by passively advecting dye[2] in a Navier-Stokes flow (cf. Section 3.1.3) and by capturing the dye intensity at two different time steps. The velocity field is the numerical solution to the NSE equipped with a user-defined swirl-like external force field. The numerical simulation is the same as the one integrated into the model-based correction step. The flow was simulated using the following parameters: grid size of 256×256 pixels, the Reynolds number $Re = 10$, density $\rho = 1000\ \mathrm{kg/m^3}$, dynamic viscosity $\mu = 0.001\,\mathrm{kg/(m \cdot s)}$, time between two exposures $\Delta t = 0.775$ ms, and pixel size pxsize $= 3.906\,\mu\mathrm{m}$. The simulation yields maximum and average displacements of 1.1 pixels and 0.2 pixels between two exposures, respectively. Figure 4.1 shows the ground truth velocity profile overlaid on (a)

[1] The EDPIV software package is available at (`http://opticallab.ncpa.olemiss.edu/EdpivPE3_intro.htm`).

[2] Further on, we refer to this test case as “dye advection”.

one of the synthetic images, (b) an intensity coding of the velocity magnitude and (c) the vorticity magnitude[3].

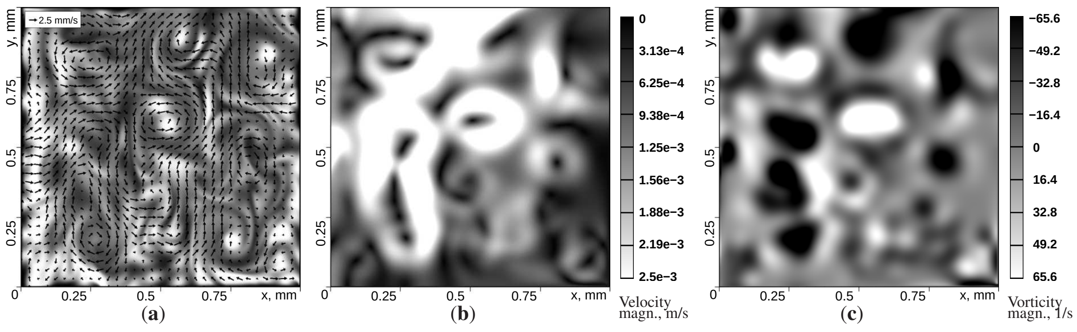


Figure 4.1: *Navier-Stokes flow: (**a**) Ground truth velocity field overlaid on one of the original images. (**b**) Intensity coding of the velocity magnitude and (**c**) the vorticity magnitude.*

The appropriate parameters of the flow model used in the reconstruction algorithm for this test case are as follows. First of all, we use the full version of the NSE-solver. The Reynolds number is not much larger than 1, thus we still can scale the external forces according to Equation 3.17. In order to use this scaling, we should first define the size of the particles seeded in the flow. However, in the recorded images of advected dye no separate particles can be distinguished. Therefore, without loss of generality, we choose the size of hypothetical particles to be equal to the size of a pixel. Since there are no obstacles present within the flow domain and there are no walls around the outer flow domain, *outflow* boundary conditions are imposed.

Figure 4.2 visualizes the spatial error distributions of the velocity magnitude $\varepsilon_{\texttt{len}}[i, j]$ and the velocity direction $\varepsilon_{\texttt{ang}}[i, j]$ reconstructed by all four approaches. The model-based techniques (16 predictor-corrector iterations) clearly outperform the CC and OF approaches. In particular, the model-based approach using the OF predictor yields the best result ($\tilde{\varepsilon}_{\text{len}}$ = 0.084 pixels and $\tilde{\varepsilon}_{\text{ang}}$ = 13°).

The per-pixel map of the estimated vorticity magnitude is depicted in Figure 4.3. The intensity coding of the vorticity magnitude reveals the fact that both the CC approach and the OF approach underestimate the vorticity. Both model-based approaches compensate for the loss in angular momentum. In particular, the OF+NS approach yields the best results: the reconstructed vorticity pattern almost matches the ground truth vorticity pattern.

A comparative visualization of the reconstructed velocity fields using overlaid arrow plots is shown in Figure 4.4. Black arrows correspond to the ground truth velocity field. Red arrows depict the vector fields reconstructed with (a) CC, (b) CC+NS, (c)

[3] The vorticity magnitude in the 2D case is computed as: $\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$.

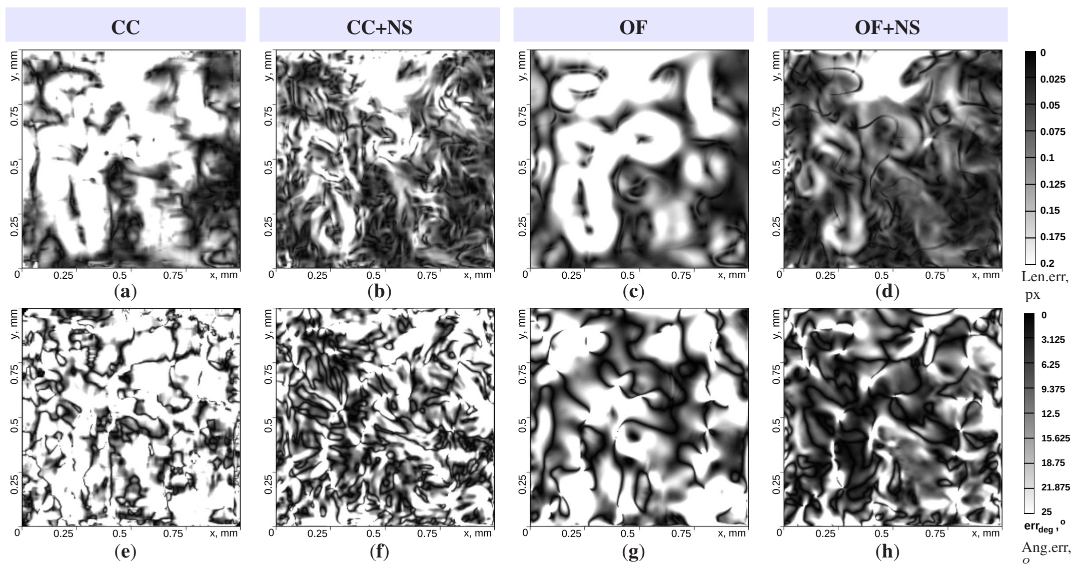


Figure 4.2: *Navier-Stokes flow:* ***Top row:*** *Spatial error distribution of the velocity magnitude* $\varepsilon_{\texttt{len}}[i,j]$. *(**a**) CC,* $\tilde{\varepsilon}_{\text{len}}$ *= 0.196 px. (**b**) CC+NS,* $\tilde{\varepsilon}_{\text{len}}$ *= 0.12 px. (**c**) OF,* $\tilde{\varepsilon}_{\text{len}}$ *= 0.169 px. (**d**) OF+NS,* $\tilde{\varepsilon}_{\text{len}}$ *= 0.084 px.* ***Bottom row:*** *Spatial error distribution of the velocity direction* $\varepsilon_{\texttt{ang}}[i,j]$. *(**e**) CC,* $\tilde{\varepsilon}_{\text{ang}}$ *= 32°. (**f**) CC+NS,* $\tilde{\varepsilon}_{\text{ang}}$ *= 22°. (**g**) OF,* $\tilde{\varepsilon}_{\text{ang}}$ *= 24°. (**h**) OF+NS,* $\tilde{\varepsilon}_{\text{ang}}$ *= 13°.*

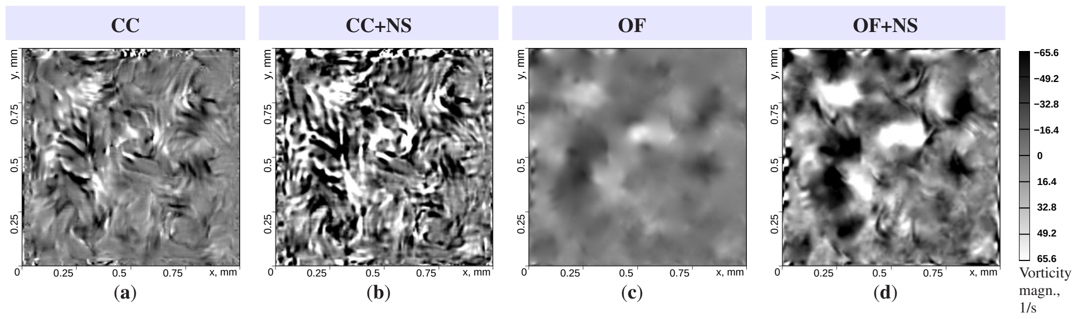


Figure 4.3: *Navier-Stokes flow, estimated vorticity magnitude: (**a**) CC, (**b**) CC+NS, (**c**) OF, and (**d**) OF+NS.*

OF, and (d) OF+NS. While the CC+NS approach can reconstruct most of the main flow structures, it cannot compensate for the high rate of erroneous velocities introduced by the CC predictor. The OF+NS approach yields the best results and recovers the flow at extremely high accuracy.

Figure 4.5 shows the graph of the average error of the velocity magnitude $\tilde{\varepsilon}_{\text{len}}$, the graph of the average angular error $\tilde{\varepsilon}_{\text{ang}}$, and a combined graph of SNR and PSNR statistics, as functions of the number of iterations for each of the evaluation methods. Due to the highly non-uniform nature of the flow, including multiple small-scale structures, the CC approach has severe problems in reconstructing the velocity field. After the first iteration the model-based approaches already yield better results than the CC and the

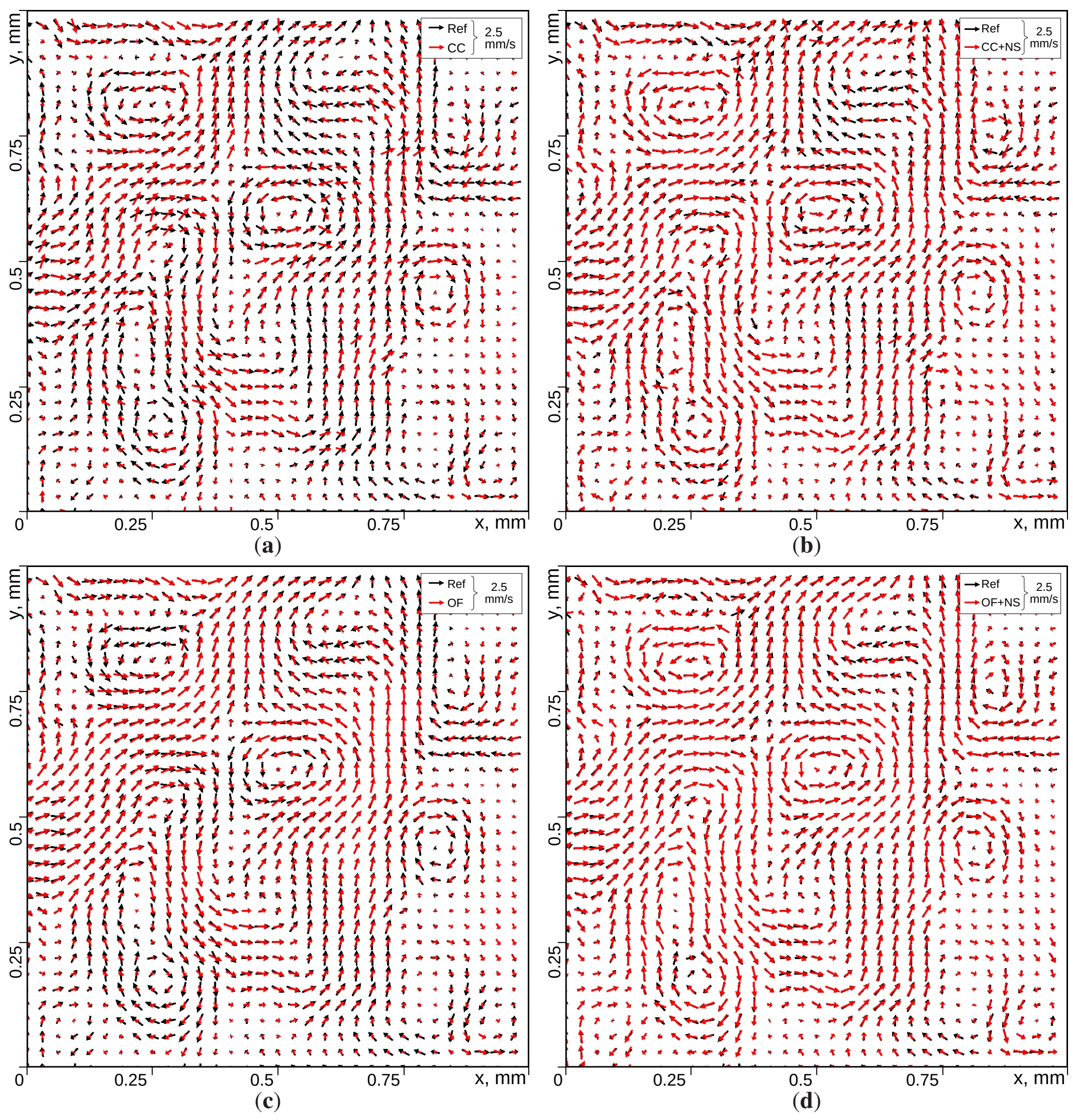
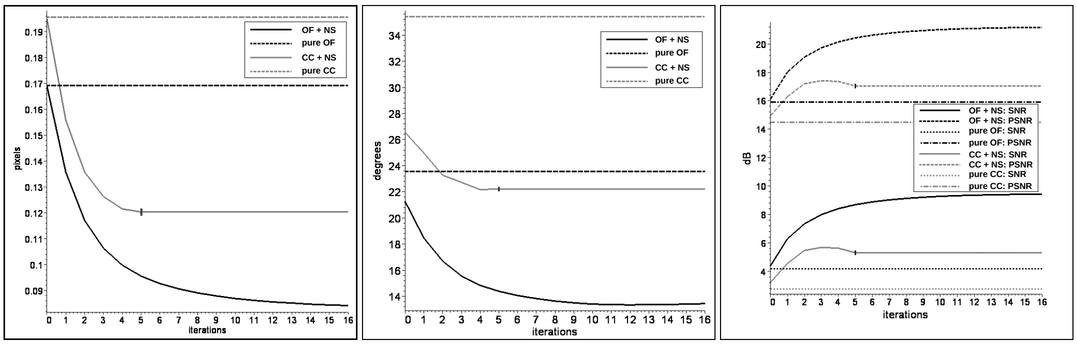


Figure 4.4: *Navier-Stokes flow: Ground truth vector field (black arrows) versus reconstructed vector fields (red arrows) using (**a**) CC, (**b**) CC+NS, (**c**) OF, and (**d**) OF+NS. Visually, the less of the black arrows can be seen the more accurate is the reconstruction with respect to direction and length.*

OF methods. With increasing number of iterations the predictor-corrector approaches improve the reconstruction quality even further, with the OF+NS approach showing a much faster convergence than the CC+NS approach. Moreover, the CC+NS approach stops improving the results after the 5th iteration. This is due to the fact that after some iterations the corrected displacements become too small and heterogeneous for the CC approach (the prediction step) to detect the resulting deformations.



(a) (b) (c)

Figure 4.5: *Error plots as functions of the number of iterations:* ***(a)*** *average error of the velocity magnitude* $\tilde{\varepsilon}_{\text{len}}$, ***(b)*** *average angular error* $\tilde{\varepsilon}_{\text{ang}}$, ***(c)*** *combined SNR and PSNR statistics.*

4.2.2 Lamb-Oseen Vortex

Particle distributions in the Lamb-Oseen vortex flow are given on a 256$\times$256 pixel grid, which corresponds to $1\times1\ \text{m}^2$ fluid domain. The fluid in this particular flow situation is air, for which $\rho\!=\!1.205\,\text{kg/m}^3$ and $\mu\!=\!1.821\cdot10^{-5}\,\text{kg/(m}\cdot\text{s)}$. The characteristic length is equal to the radius of the Lamb-Oseen vortex $L = 0.167\,\text{m}$. The time step between two exposures is $\Delta t = 1$ s, and the pixel size is $\text{pxsize} = 3.906\,\text{mm}$. The experimental setup yields maximum and average displacements of 0.55 and 0.29 pixels between two frames, respectively. Figure 4.6 (a) shows one of the synthetic images. The ground truth velocity profile overlaid on the intensity coded per-pixel maps of the velocity and the vorticity magnitudes is shown in Figure 4.6 (b) and Figure 4.6 (c), respectively.

The parameters of the flow model for the reconstruction algorithm appropriate for the Lamb-Oseen vortex flow are as follows. First of all, we use the full version of the NSE-solver. The Reynolds number (Re = 52) is quite small[4], thus we can still scale the external forces according to Equation 3.17. The radius of the particle tracers is $2.93\,\text{mm}$. Since there are no obstacles present within the flow domain and there are no walls around the outer flow domain, outflow boundary conditions are imposed.

The same qualitative analysis as in the first experiment was carried out using 18 (OF+NS) and 8 (CC+NS) predictor-corrector steps in the model-based approach. Figure 4.7 shows the per-pixel error in the estimated velocity magnitudes $\varepsilon_{\texttt{len}}$ and velocity directions $\varepsilon_{\texttt{ang}}$, and it confirms our previous results on the quality of the four evaluation methods. Again, the model-based approach using the OF predictor yields the best results ($\tilde{\varepsilon}_{\text{len}}$ $=$0.026 pixels and $\tilde{\varepsilon}_{\text{ang}}$ $=$0.79°).

[4] In general, Stokes' law is valid only for $Re \leq 1$. However, our goal is to use the model within the filter operation. Therefore, it is acceptable that the model describes the flow *approximately* and *not necessarily precisely*.

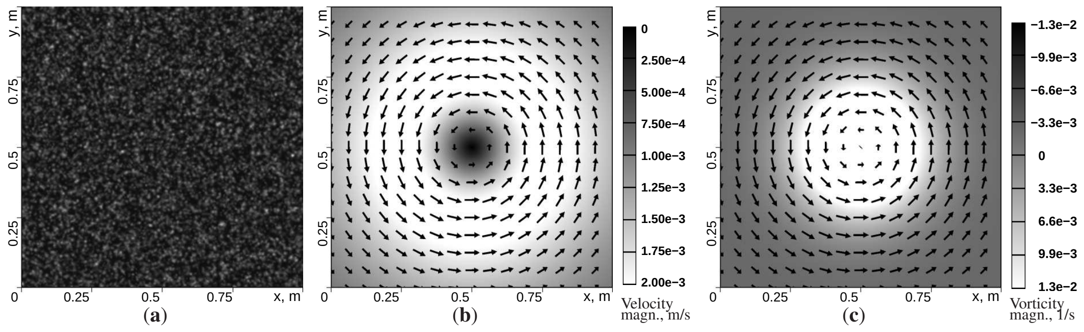


Figure 4.6: *Lamb-Oseen vortex flow: (**a**) One of the original images. Ground truth velocity field overlaid on an intensity coded per-pixel map of (**b**) the velocity magnitude and (**c**) the vorticity magnitude.*

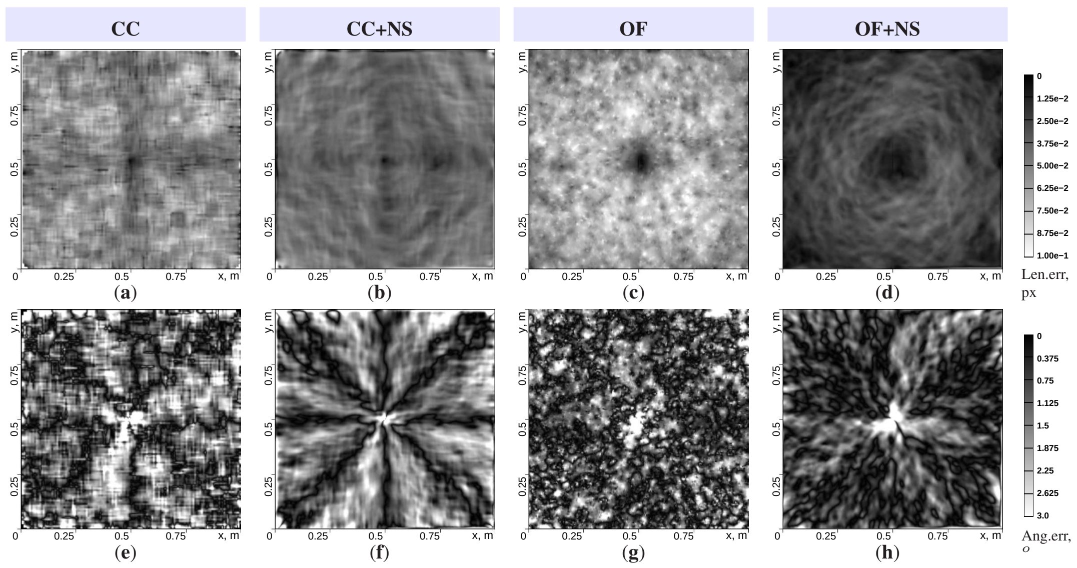


Figure 4.7: *Lamb-Oseen vortex flow: **Top row:** Spatial error distribution of the velocity magnitude $\varepsilon_{\texttt{len}}[i,j]$. (**a**) CC, $\tilde{\varepsilon}_{\text{len}}$ = 0.054 px. (**b**) CC+NS, $\tilde{\varepsilon}_{\text{len}}$ = 0.042 px. (**c**) OF, $\tilde{\varepsilon}_{\text{len}}$ = 0.065 px. (**d**) OF+NS, $\tilde{\varepsilon}_{\text{len}}$ = 0.026 px. **Bottom row:** Spatial error distribution of the velocity direction $\varepsilon_{\texttt{ang}}[i,j]$. (**e**) CC, $\tilde{\varepsilon}_{\text{ang}}$ = 1.24°. (**f**) CC+NS, $\tilde{\varepsilon}_{\text{ang}}$ = 1.23°. (**g**) OF, $\tilde{\varepsilon}_{\text{ang}}$ = 0.99°. (**h**) OF+NS, $\tilde{\varepsilon}_{\text{ang}}$ = 0.79°.*

Figure 4.8 shows the distribution of the vorticity magnitude for all four approaches. The intensity coding reveals that the OF method underestimates this quantity, whereas the CC method yields a slight overestimation. In either case the predictor-corrector approach compensates for this error.

As illustrated in Figure 4.9, for the Lamb-Oseen flow the convergence behavior of all four approaches is similar to the behavior we observed in the dye advection experiment (cf. Figure 4.5). Although CC+NS starts with a lower reconstruction error in this example, OF+NS still significantly outperforms CC+NS due to its faster convergence and the stagnation of CC+NS after a few iterations. Interestingly, the graph of

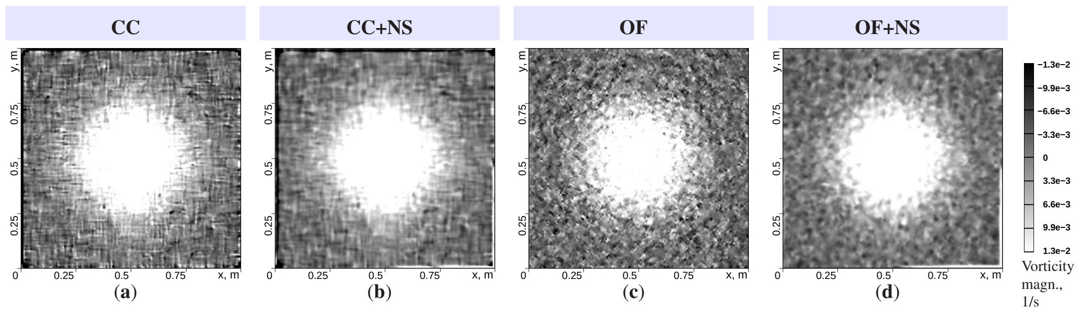


Figure 4.8: *Lamb-Oseen vortex flow, estimated vorticity magnitude: (**a**) CC, (**b**) CC+NS, (**c**) OF, (**d**) OF+NS.*

the average angular error $\tilde{\varepsilon}_{\mathrm{ang}}$ for the CC+NS approach starts below the average angular error level for the CC approach but then rapidly approaches this level and stops there. Such behavior is mainly influenced by the large spatial region of the flow field with strong velocity gradients around the center of the vortex, where the correction step cannot compensate for the inability of the CC method in the prediction step to reconstruct highly heterogenous motion within the interrogation window.

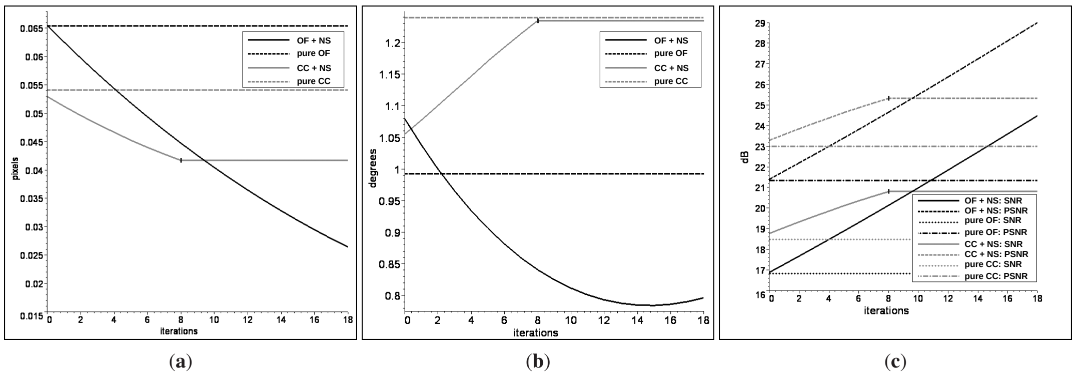


Figure 4.9: *Error plots as functions of the number of iterations: (**a**) average error of the velocity magnitude $\tilde{\varepsilon}_{\mathrm{len}}$, (**b**) average angular error $\tilde{\varepsilon}_{\mathrm{ang}}$, (**c**) combined SNR and PSNR statistics.*

Moreover, as can be concluded from a comparative analysis of the error graphs presented in Figure 4.5 and Figure 4.9, the convergence rate of the reconstruction algorithm for dye advection flow is significantly higher than that for Lamb-Oseen flow. There are several possible explanations for such a phenomenon. First of all, Lamb-Oseen flow field spans a high dynamic range of velocity magnitudes. Due to discretization errors of the imaging process in the regions of low velocity magnitude (in the center of the vortex), the recorded particle images are randomly distorted and thus the restoration of reliable information from such images can be complicated. On the other hand, in the particle images obtained for the dye advection experiment, there are no particles as such, and thus even in the presence of low velocity magnitudes, the image distortion is not that severe. Second, Stokes' law is not very well suited for the Lamb-Oseen flow with $Re = 52$, yet the algorithm still converges at a sufficiently high rate.

A comparative visualization using overlaid arrow plots is shown in Figure 4.10. Even though no significant difference in the velocity fields can be observed from these plots, the graph of convergence behavior (Figure 4.9) as well as the intensity-coded spatial error distributions of the velocity magnitude $\tilde{\varepsilon}_{\text{len}}$ and the velocity direction $\tilde{\varepsilon}_{\text{ang}}$ (Figure 4.7) clearly show the advantage of our model-based approach with OF as predictor over the other approaches.

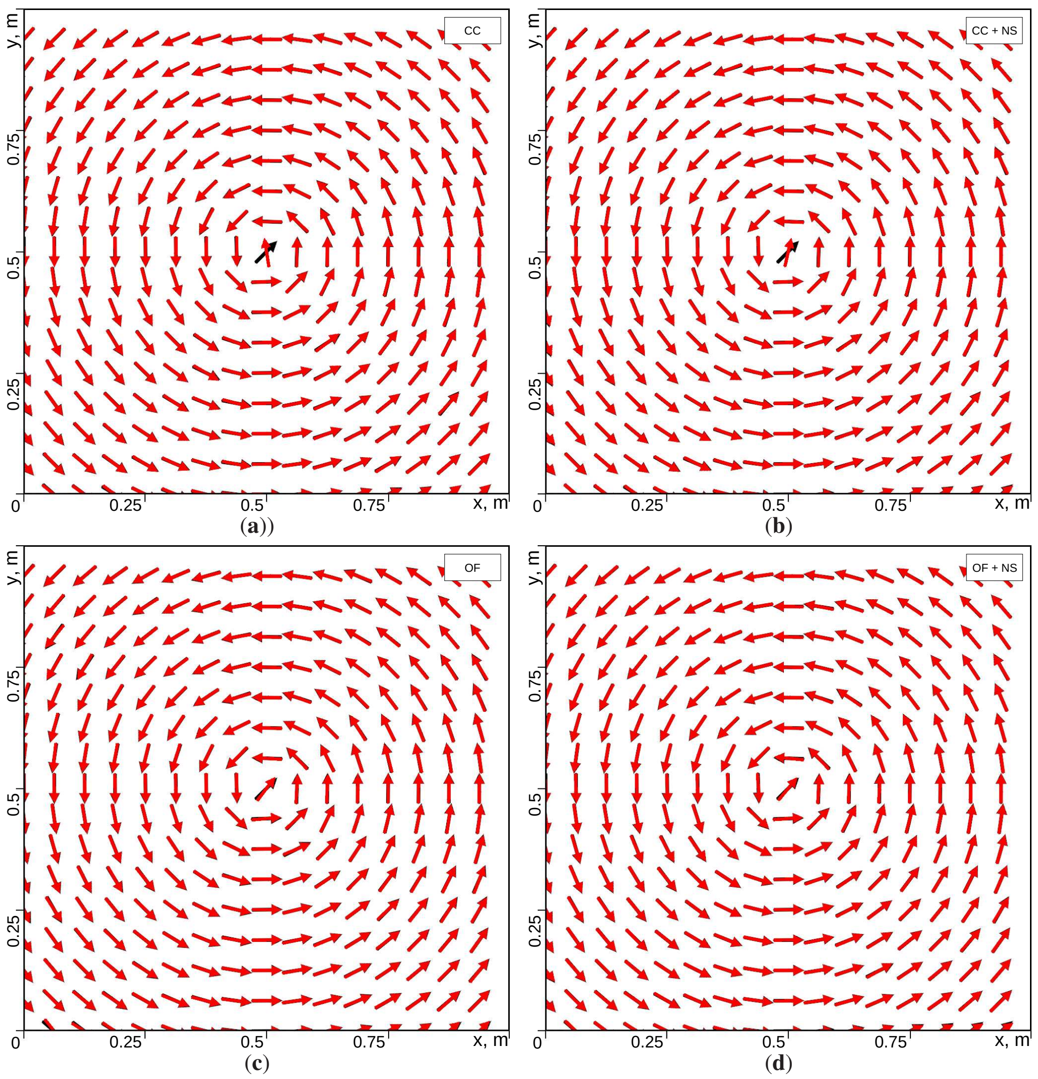


Figure 4.10: *Lamb-Oseen vortex flow: Ground truth vector field (black arrows) versus reconstructed vector fields (red arrows) using (**a**) CC, (**b**) CC+NS, (**c**) OF, and (**d**) OF+NS. Visually, the less of the black arrows can be seen the more accurate is the reconstruction with respect to direction.*

4.3 Results of Reconstruction: Experimental Image Pairs

From a wide range of application areas for real-world PIV-experiments, we have chosen a group of *micro-flow experiments* on living microorganisms. Given a typical biologically active liquid environment naturally seeded with nutrient particle tracers, the PIV method offers an excellent opportunity to investigate the motion and fluid exchange characteristics of microorganisms. The motivation for this investigation is two-fold: first of all, the flow model underlying the micro-flows, the *Stokes flow*, is very simple yet challenging, due to its natural instability [HOP^{+}07] and limitations of the imaging and particle seeding processes resulting from the necessity to maintain the *biocompatibility* (cf. Section 2.2.3). Second, gaining insight into the natural *biomechanical processes* and *bioconvection* involving the microorganisms is of vital interest to the community.

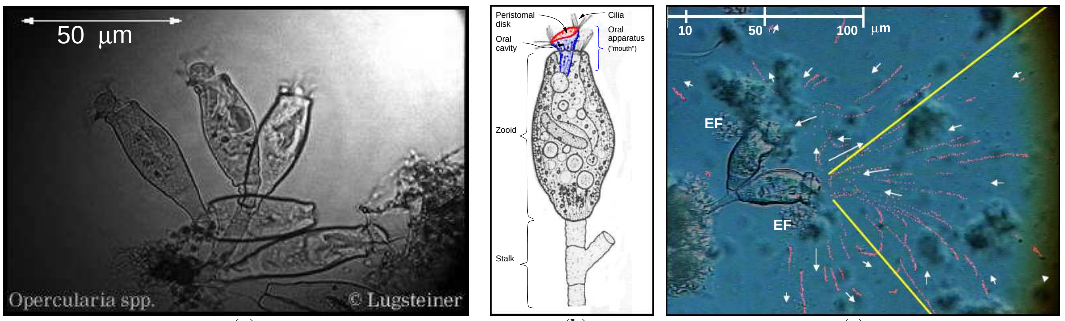
Due to restrictions imposed on *biocompatible microscopic PIV systems*, the reconstruction of admissible flow fields from captured images is highly complicated. In this section, we demonstrate the effectiveness of our model-based approach for the reconstruction of flow fields induced by living microorganisms. In particular, we show that due to the combination of experimental procedures and advanced computing technology we can reconstruct flow fields that conform to the characteristic flow patterns observed in nature.

We should mention here that all of our computations are performed under the assumption that the particle movement and the reconstructed flow patterns are *two-dimensional*. In the experiments we perform on living organisms this assumption is not valid in general, and as a consequence the proposed method cannot always reconstruct the fluid motion in a reliable way. On the other hand, as we will describe in Sections 4.3.1 and 4.3.2, the experimental setup we used was chosen in such a way as to enforce the two-dimensional condition. Even though this might prohibit the accurate reconstruction of the real flow patterns, it allows us to effectively validate the results of our approach in a real-world scenario.

The world of microorganisms is incredibly diverse and includes bacteria, fungi, archaea, and protists, as well as some microscopic plants and animals. Thus, instead of referring to the investigation of flow patterns initiated by microorganisms, we should be more specific. In this thesis we study the fluid flows induced by *sessile* ciliates[5] *Opercularia asymmetrica* which form their colonies on the surface of granular activated sludge

[5] Sessile microorganisms are organisms which are usually permanently attached via stalk to a solid substrate of some kind (e.g., granular activated sludge). The name ciliate comes from the presence of hair-like organelles ("organs" of microorganisms) called *cilia*.

growing in bioreactors for biological wastewater treatment. A detailed description of these species, their geometries as well as the fluid flows induced by the motion of these organisms can be found, e.g., in [BOB98, FL03, Utz03, HOP^{+}07]. Figure 4.11 (a),(b) shows a single tree-like colony and a schematic anatomical structure of these species. Figure 4.11 (c) illustrates a characteristic flow pattern induced by ciliates [FL03].



(**a**) (**b**) (**c**)

Figure 4.11: *Sessile peritrich ciliate Opercularia asymmetrica: (***a***) a single tree-like colony, (***b***) schematic anatomical structure, (***c***) characteristic flow pattern [FL03].*

In Figure 4.11 (c) arrows indicate the direction of particles motion, straight lines confine the region where particles are moving towards the mouth of one ciliate, and "EF" denotes egested food. Two microscopic PIV experiments with different parameter settings have been carried out to obtain the images of seeded particles in the flow investigated in Sections 4.3.3-4.3.4.

4.3.1 Experimental Apparatus

In the experiments a light transmitted microscope Axiotech 100 (Carl Zeiss, Germany) with 20× and 50× optical magnification was used. The observation volume was illuminated by a standard light source integrated into the microscope, and *biocompatible flow tracers* (yeast cells with characteristic size of 5-10 μm) were seeded into the volume. Images were captured by a CCD camera with macro-zoom lens at a frame rate of 25 s^{-1} [KZD07].

From the experiments it could be observed that the fluid motion is induced by natural bioconvection processes, and that the microorganisms have different strategies of flow control via cilia beat and contraction of the *zooid*[6] itself. The typical flow pattern is characterized by clearly distinguishable *large-scale viscous eddies* in the close vicinity of the *oral cavity* of the ciliates. The distance of the vortex center from the substrate surface, where the ciliates reside, can be used as the characteristic length L,

[6] Zooid (or protozoon) is the main body of the microorganism (see Figure 4.11 (b)).

which is of the same order of magnitude as the length of the organisms (50 $\mu\mathrm{m}$). The measured flow field exhibits maximum velocities of 25 $\mu\mathrm{m/s}$ at around 20 $\mu\mathrm{m}$ ahead of an individual ciliate, and it extends 200 $\mu\mathrm{m}$ from the location of the ciliate. The *nodding motion*[7] of the protozoon is found not to obey any periodic law. In groups of protozoa an instationary flow field is produced by alternating cilia beats.

Since the fluid surrounding the microorganism is water[8], the Reynolds number, computed according to Equation 3.4, is around 1.25×10^{-3}. This implies that the underlying flow is governed by the Stokes equation (cf. Section 3.1.3). It was proven that the *Stokes drag force* (cf. Equation 3.17) dominates by far all other forces acting on the tracer particles, and that it is in the order of magnitude of 10-100 pN. Thus, the forces exerted on the fluid by a ciliate to induce and maintain the fluid motion against viscous dissipation should also be around 10-100 pN [HOP+07].

4.3.2 Flow Model

The bioconvection processes caused by sessile organisms suggest that these organisms exert a net force on the fluid towards or away from the boundary. The boundary is the organism itself and the substrate where it resides. This process leads to a single *toroidal eddy* above the oral cavity of the organism, which is very similar to a toroidal flow pattern corresponding to a stream function given by Equation 3.9 and illustrated in Figure 3.4 [OYB01, BOB98]. The experimental investigation of the fluid flow induced by such microorganisms is usually carried out between a microscope slide and cover plate, which represent impenetrable plane boundaries to the fluid and squeeze the toroidal eddy in direction perpendicular to these planes as depicted in Figure 4.12 [LB81, BO96, BOB98].

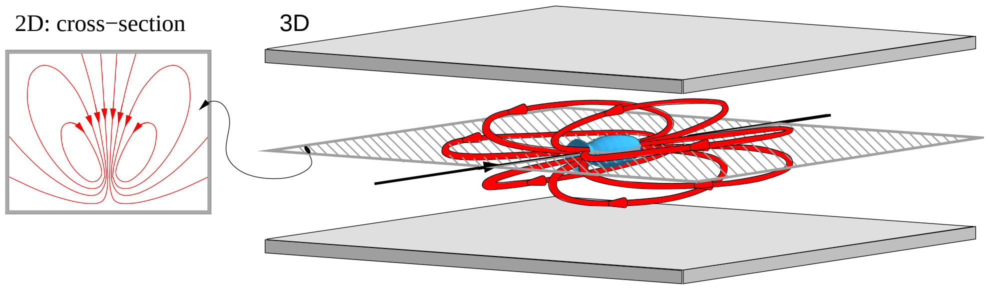


Figure 4.12: *Toroidal eddy (red tubes) induced by the microorganism (blue bell) which resides between the microscope slide and the cover plate. The cross-section plane passes through the main axis of the microorganism.*

[7] Being a sessile microorganism, a ciliate can vary its inclination angle with respect to its initial orientation by adjusting the stalk. This motion is referred to as *nodding motion* [HOP+07].

[8] For water $\mu = 10^{-3}\,\mathrm{kg/(m \cdot s)}$ and $\rho = 10^{3}\,\mathrm{kg/m^{3}}$.

As suggested in [Lir82] and [BOB98], the action of a single sessile microorganism residing between the slide and the cover plate can be modeled by a single stokeslet (cf. Section 3.1.3) acting normal to a plane boundary and parallel to two infinite parallel plates. The stokeslet is situated at the tips of the cilia (approximately 10 $\mu\mathrm{m}$ from the *peristomal disk*[9] of the ciliate), aligned along the main axis of the microorganism and pointing towards the oral cavity of the organism as illustrated in Figure 4.13 (b). The magnitude of the point force is set according to the maximum of the velocity magnitudes computed in the very first iteration. This model is simple, but according to a quantitative analysis performed in [HOP$^+$07] it reproduces the main structures of the fluid flow very well. Therefore, we incorporate this model in combination with the Stokes flow assumption into our model-based reconstruction method.

In case of a single toroidal eddy it is possible to select a slice in the investigated fluid volume where the fluid motion is in a two-dimensional plane. The flow pattern in this cross-section consists of two counter-rotating eddies. Since the toroidal vortex is usually situated right above the oral cavity of the microorganism, the best choice for a cross-section plane is the plane through the main axis of the microorganism (cf. Figure 4.12). Since the depth of the focusing plane of the microscope used in our experiments does not exceed 10 $\mu\mathrm{m}$, the average diameter of the microorganism's zooid is approximately 20 $\mu\mathrm{m}$, and the flow field extends about 200 $\mu\mathrm{m}$ from the location of the microorganism [PKD$^+$07], the necessary condition for two-dimensionality can be provided. Additionally, in order to satisfy the condition for cross-sectional flow to be *in plane*, only the specimens located nearly *parallel* to the microscope slide and the cover plate should be considered [HOP$^+$07].

One additional question that should be addressed is how to determine the shape of the fluid boundaries considered in our approach. Currently, the microorganisms themselves are considered as obstacles, and the shape of the organisms is extracted from image pairs by a standard segmentation technique [GSAW05]. On the other hand, especially the oral apparatus is very complicated and, what is even more important, the position and shape of organelles within the oral apparatus can suddenly change [Slá81, ZC01, Utz03]. Moreover, as it was observed in several research studies, the flow in the close vicinity of the oral cavity is highly unsteady and erratic [May00, HOP$^+$07]. According to these observations, some particle tracers can pass aborally while others move straight into the oral cavity, some of them might be even rejected by the microorganism. The precise modeling of such phenomena is still an open area of research, and it is in particular not within the scope of this thesis to develop such a

[9] Peristomal disk is the part of the oral apparatus of the organism covering the oral cavity (see Figure 4.11 (b)).

model or to simulate the fluid flow based on such a model. Instead, we use the *large scale model* suggested by Blake and Otto [BOB98] in our model-based reconstruction method. Since the outer part of the oral apparatus of the microorganism resembles a flexible funnel (see Figure 4.11 (a),(b)), which can suddenly change its shape, dilate and close its stem, the "mouth" of the organism is modeled as a rather shallow rounded cavity (see Figure 4.13 (b) and Figure 4.16 (c)).

Concluding the discussion about the fluid boundaries, the boundary conditions applied in the NSE-solver are as follows. Since the liquid surrounding the microorganisms is neither penetrating their zooids nor the substrate, *no-slip* boundary conditions can be applied to the extracted microbiological structures. As only a small part of the whole fluid environment of the microorganisms is covered in our experiments, *outflow* boundary conditions have been enforced for the outer domain border.

4.3.3 Experiment in Water Environment

In the first experiment [PKD$^+$07], the images were acquired using 20$\times$ optical magnification. The recorded particle images have a high signal-to-noise ratio and a resolution of 860$\times$1024 pixels corresponding to 505 μm $\times$ 602 μm. Yeast cells with a characteristic size of 5-10 μm served as biocompatible tracer particles; a 1:100 yeast to water solution was used. Figure 4.13 (a) shows one of the recorded images. Since the colony of microorganisms is located only in the left upper corner of the image, sub-images of size 256$\times$256 pixels were cut out of the original images (see Figure 4.13 (b) , *top*).

In Figure 4.13 the following objects can clearly be distinguished: *large dark objects* indicating the substrate on which ciliates form their colonies, *bell-shaped objects* with clearly visible outer contours (species of Opercularia asymmetrica), and *round small objects* (yeast cells). From the observation that some particles have sharp contours while others are blurred one can conclude that some particles are situated outside the focus plane of the microscope. Figure 4.13 (b) , *bottom,* shows the segmented structures that were used as flow boundaries in the numerical solution of the NSE.

The reconstructed velocity fields for all four evaluation methods are shown in Figure 4.14. In the OF+NS approach, 8 predictor-corrector iterations were performed. As expected, the CC approach failed to reconstruct an admissible velocity field due to the low number of tracer particles. As a consequence, we stopped the evaluation with the CC+NS approach after the first iteration. The OF and the model-based OF+NS approaches are capable of recovering a flow field that conforms to the characteristic pattern of the fluid flow observed in nature. However, in regions where particles are not present, the OF approach reconstructs velocities close to zero. Moreover, it fails

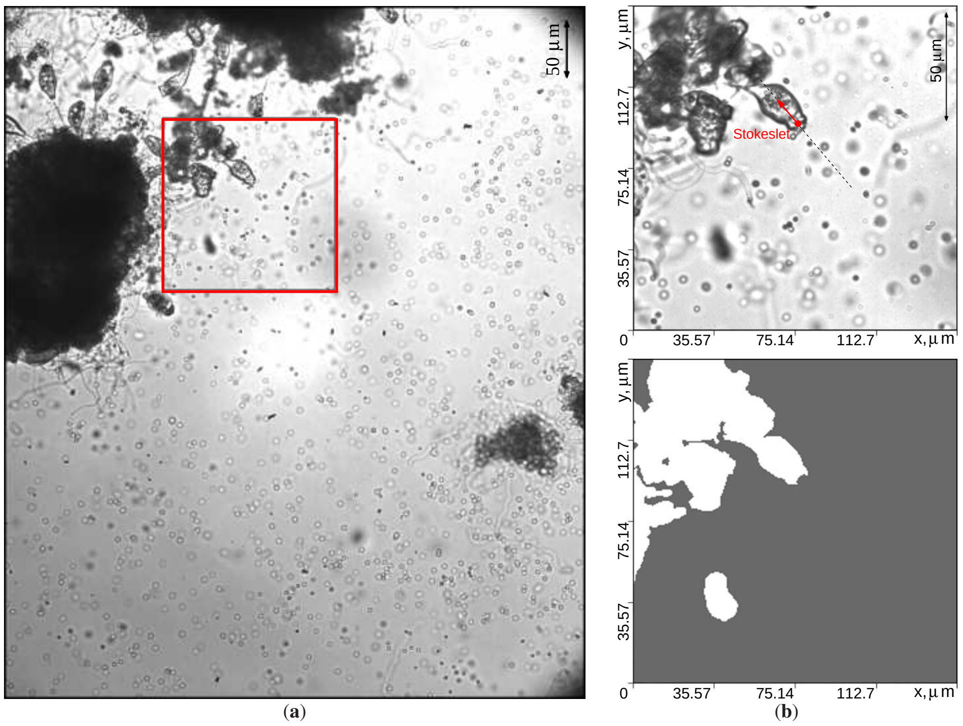


Figure 4.13: *Experiment in water environment: (**a**) Recorded image of particles in the flow induced by living ciliates. (**b**) Cut out image (top) and segmented flow boundaries (bottom).*

to recover a strong viscous eddy induced by the movement of the active ciliate. The predictor-corrector approach resolves both problems as it extracts a strong viscous vortex to the left of the ciliate and estimates velocity distributions according to the laws of fluid dynamics in regions with low particle density.

To demonstrate the reliability of the estimated velocity field, we used a *Particle Tracking* (PT) technique to determine the velocity at selected positions in the domain, and we then analyzed the deviation of our solution from these velocities. Particles were first segmented manually and correspondences between particle pairs in consecutive images were established. Velocities were then reconstructed from the trajectories of these particles over time. Figure 4.15 shows a graphical illustration of the segmented particles and their velocities (red circles with arrows) overlaid on the velocity field reconstructed with (a) OF and (b) OF+NS. In this experiment, the maximum and average displacement magnitudes of all of the tracked particles are 3.1 and 0.41 pixels, respectively. Our measurements indicate that the velocity magnitudes estimated with OF deviate significantly from the values computed with PT ($\tilde{\varepsilon}_{\text{len}} = 0.4$ pixels). The velocity distribution obtained with OF+NS, on the other hand, matches the reference distribution very well ($\tilde{\varepsilon}_{\text{len}} = 0.1$ pixels). This result also indicates the improvements that can be achieved by using the proposed model-based predictor-corrector scheme.

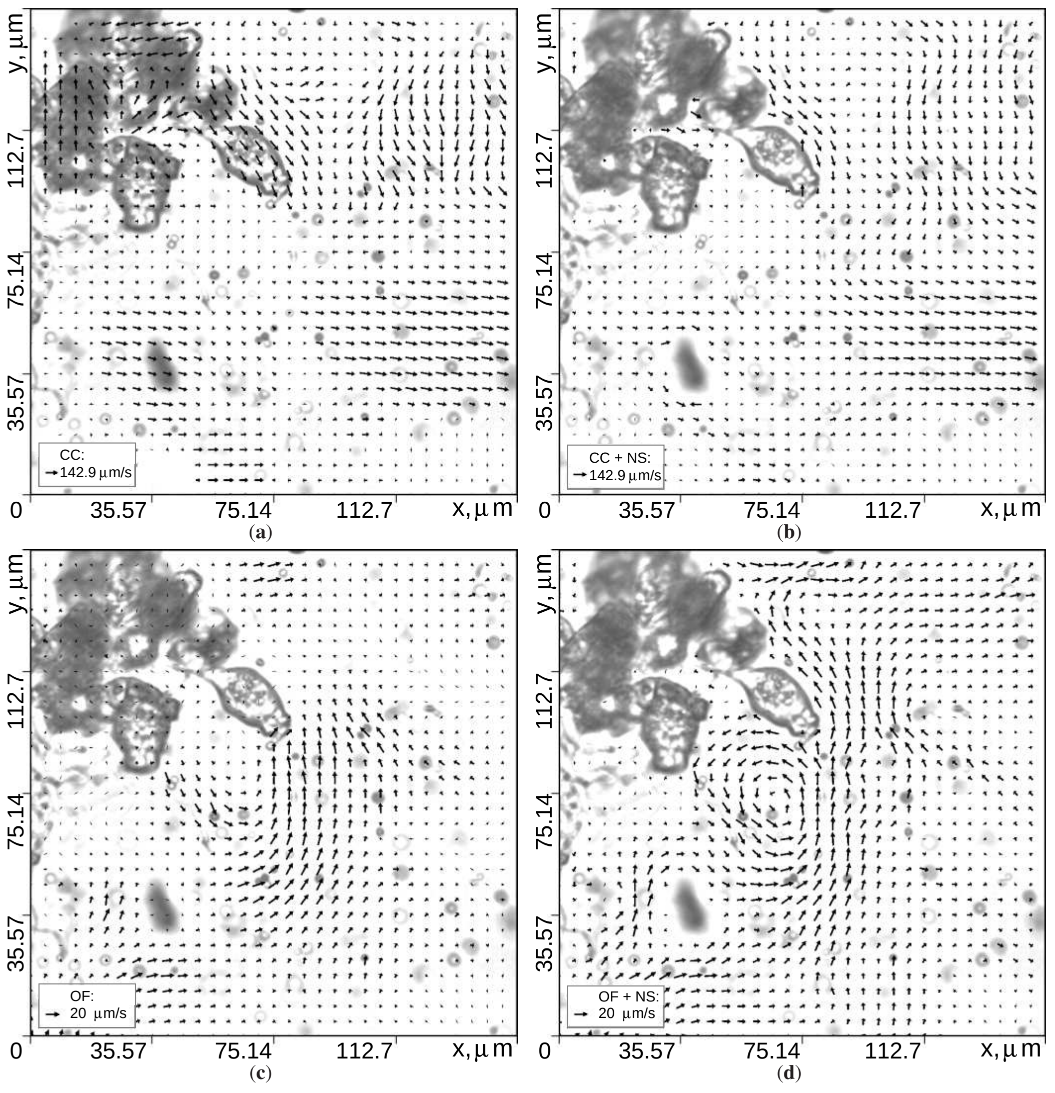


Figure 4.14: *Experiment in water environment: Reconstructed velocity fields using (**a**) CC, (**b**) CC+NS, (**c**) OF, and (**d**) OF+NS. Only the model-based OF+NS approach can reconstruct one of the vortices typically observed in nature.*

4.3.4 Experiment in Milk Solution

In the second experiment [KZD07], the same imaging device as in the first experiment with 50× optical magnification was used. Sub-images of resolution 860×860 pixels were cut out of the original scans. A 1:3 milk to water solution was used as tracer medium, with fat and proteins of characteristic size 0.3-3 $\mu\mathrm{m}$ serving as tracer particles[10]. The light transmission properties of milk prevent particle motion in out-of-focus layers to be captured. On the other hand, low image contrast combined with high

[10] A 1:3 milk to water solution has viscosity $\mu = 1.3 \cdot 10^{-3}\,\mathrm{kg/(m \cdot s)}$ and density $\rho = 1.01 \cdot 10^{3}\,\mathrm{kg/m^3}$.

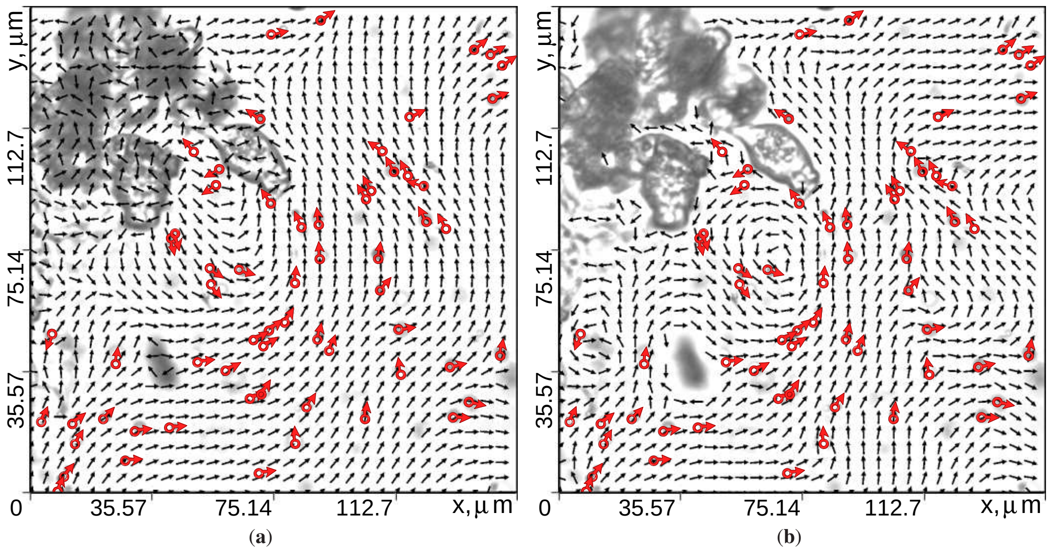


Figure 4.15: *Experiment in water environment: Comparison of particle velocities computed with PT (red arrows) and (**a**) OF (black arrows), (**b**) OF+NS (black arrows). The circles indicate tracked particles and the red arrows visualize their velocities. Note that black and red arrows are constant in length for better visibility.*

particle density introduces additional difficulties for the reconstruction algorithms. Figure 4.16 shows (a) one of the real-world images, (b) a superposition of 9 images with additional contrast enhancement to emphasize particle trajectories in the image, and (c) the segmented domain boundaries. The red circles with arrows in Figure 4.16 (b) correspond to tracked particles and their velocities.

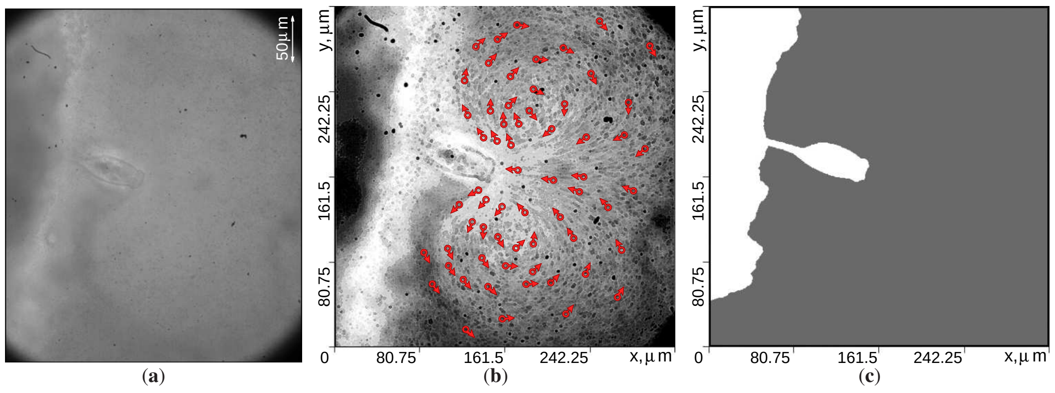


Figure 4.16: *Experiment in milk solution: (**a**) Recorded image of particles in the flow induced by living ciliates. (**b**) Contrast-enhanced superposition of 9 subsequent images and tracked particles with their velocities (red circles with arrows). (**c**) Segmented flow boundaries.*

Figure 4.17 shows that all four evaluation methods were able to reconstruct the main structures in the flow: two nearly axisymmetric viscous eddies separated by the extended medial axis of the ciliate. However, in some regions the velocity vectors reconstructed by the CC approach are not aligned with the corresponding particle traces in the superposed image. For example, non-zero velocities are extracted across the microorganism and the velocity direction is slightly deviating from the superposed particle traces in the vicinity of the vortex centers, where the velocity magnitudes are low and the velocity directions within a small area are heterogeneous.

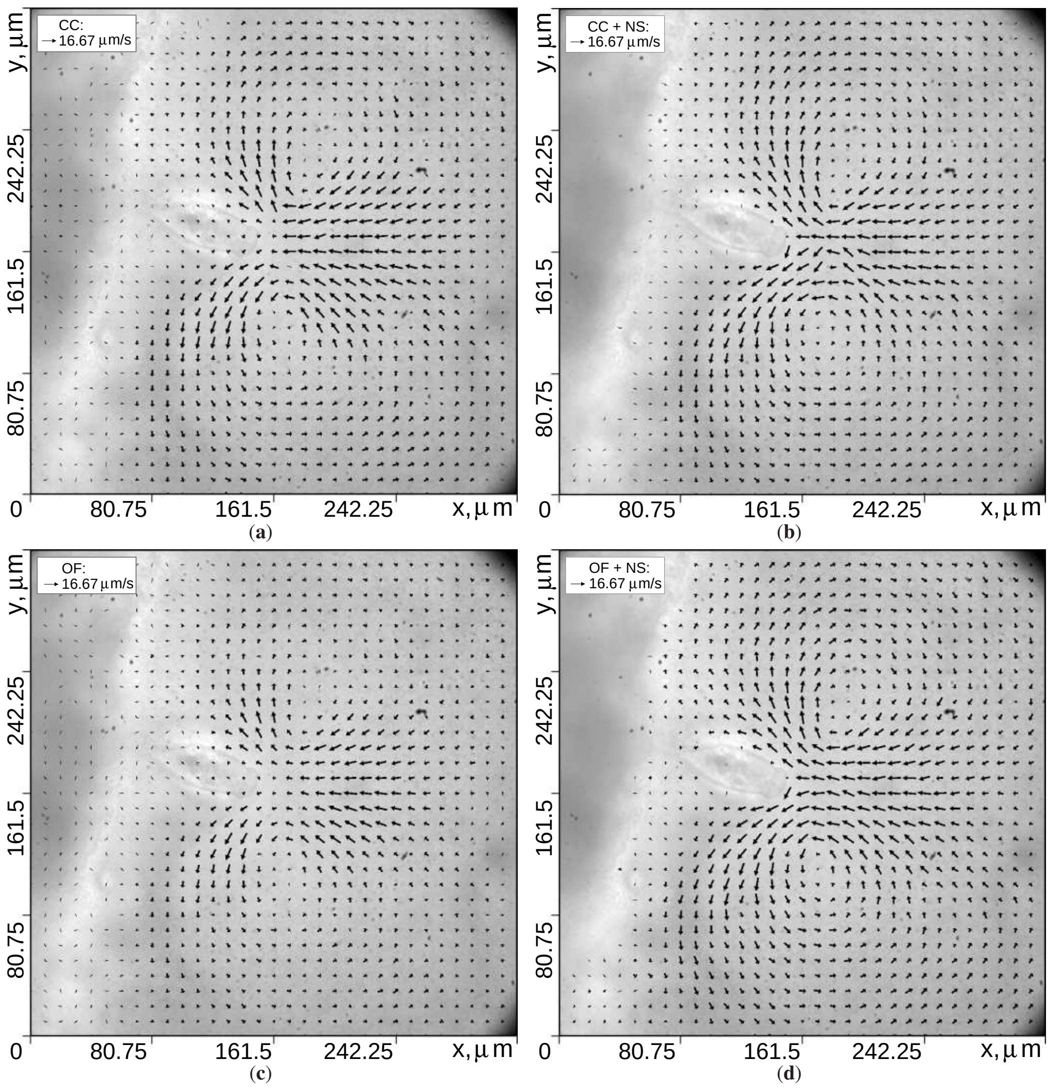


Figure 4.17: *Experiment in milk solution: Reconstructed velocity fields using (**a**) CC, (**b**) CC+NS, (**c**) OF, and (**d**) OF+NS. All four approaches are capable of reconstructing two nearly axisymmetric viscous eddies. Only OF+NS is able to reconstruct the true velocity in the vicinity of the oral cavity of the ciliate and around the vortex centers.*

Although both the CC and the OF approaches can reconstruct the characteristic flow patterns in most regions of the domain, in the vicinity of the oral cavity of the ciliate the velocity is unexpectedly low[11]. In these regions, the velocity magnitudes are so high (up to 8 pixels per exposure time) that both approaches fail to recover the particle movement. The model-based approaches, on the other hand, can successfully reconstruct the velocity distribution even in these regions. Furthermore, close to the vortex centers, where the velocity magnitudes are low, the advantages of the OF+NS approach can be also clearly seen.

The accuracy of the OF+NS approach is analyzed using the PT-based technique described in the first experiment (cf. Section 4.3.3). Figure 4.18 compares the reference velocities obtained with PT to the results using (a) the CC+NS and (b) the OF+NS approaches. While the CC+NS approach yields an average error in the velocity magnitude $\tilde{\varepsilon}_{\text{len}} = 0.15$ pixels, the velocity magnitudes estimated by the OF+NS approach are almost equal to the reference values ($\tilde{\varepsilon}_{\text{len}} = 0.08$ pixels). The comparison confirms our previous results, and it demonstrates the capability of the model-based approach to significantly improve the reconstruction process in real-world scenarios.

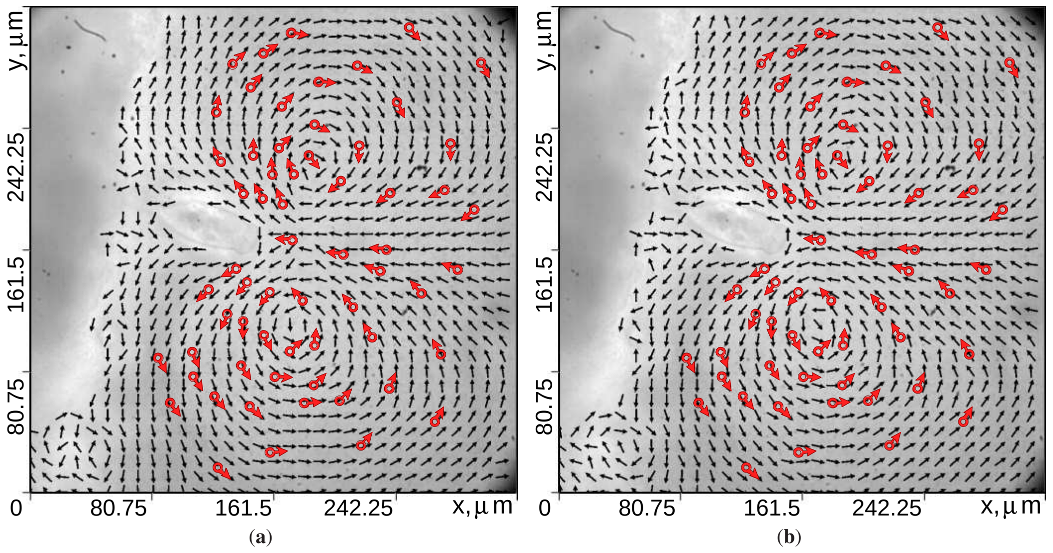


Figure 4.18: *Experiment in milk solution: Comparison of particle velocities computed with PT (red arrows) and (**a**) CC+NS (black arrows), (**b**) OF+NS (black arrows). The circles indicate tracked particles and the red arrows visualize their velocities. Note that black and red arrows are constant in length for better visibility.*

[11] According to the observations in nature, the largest velocity magnitudes are expected in the vicinity of the oral cavity of the microorganism.

4.4 Performance Evaluation

All our evaluations were run on a standard 3 GHz Intel Pentium 4 CPU with 2 GB RAM and a GeForce 8800 GTX graphics card with 768 MB local video memory. Timing statistics including the number of CG-iterations (NSE-solver) and V-cycles (multigrid OF-solver) performed in one predictor-corrector cycle are presented in Table 4.2. Since the OF computation and the numerical solution of the NSE are the most time consuming parts of our approach, timings for these parts are shown separately in columns two and three, respectively. The total time required to complete one iteration is given in the fourth column. In order to estimate the performance for larger resolutions we run our method on the full resolution images (860×1024) available from the first experiment. The timings are shown in the third row. All timings are given in seconds.

Table 4.2: *Timings for one predictor-corrector iteration on the GPU.*

Data set	OF Multigrid runtime [s] (#V-cycles)	NSE runtime [s] (#CG iterations)	Total runtime [s]
Experiment 1: (256×256)	0.037 (5)	0.027 (20)	0.067
Experiment 2: (512×512)	0.160 (5)	0.112 (40)	0.300
Reference: (860×1024)	0.693 (5)	0.503 (80)	1.341

The timings show that the technique we propose still performs at nearly interactive rates even for high resolution images of up to 1024×1024 pixels. Thus, the system offers the possibility for in situ reconstruction of flow fields from consecutive image sequences. Even more importantly, for the first time it is now possible to interactively control specific parameters of both the flow model and the reconstruction process, and to obtain direct feedback of the imposed changes.

The interface assigned to the reconstruction process enables the user to change physical properties of the underlying flow model such as *Re*, μ, and ρ, the smoothness criterion α, and experimental settings (time interval Δt between frames, pixel size, etc.). In particular, in experimental fluid dynamics, where the exploration of fluid flow is often an iterative process based on rather vague assumptions on the underlying models, we expect this flexibility of increasing importance.

It is worth noting that with respect to previous hierarchical implementations of the OF computation on CPUs, the proposed GPU-based implementation shows excellent performance. Compared to the most recent timings given in [BWF$^+$05] and [RKSN05], the GPU-implementation outperforms advanced CPU solutions by a factor of about 2-3 and a factor of 8-10 respectively.

In addition, as both the OF computation and the NSE-solution run on the GPU, they can be directly coupled with an interactive visualization module running on the same architecture [KKKW05]. In the current application this module provides to the user multiple visualization options for 2D flow fields, including particle tracing and stream lines or stream balls visualization. There is also an opportunity to choose among the various types of differential quantities[12], such as vorticity $\boldsymbol{\omega}$ and shear-strain ϵ_{xy}, for filling up the overlay map rendered behind the visualized geometry primitives. Some results are shown in Figure 4.19. In this way, we can entirely avoid any data transfer between the CPU and the GPU, which otherwise slows down the overall system performance.

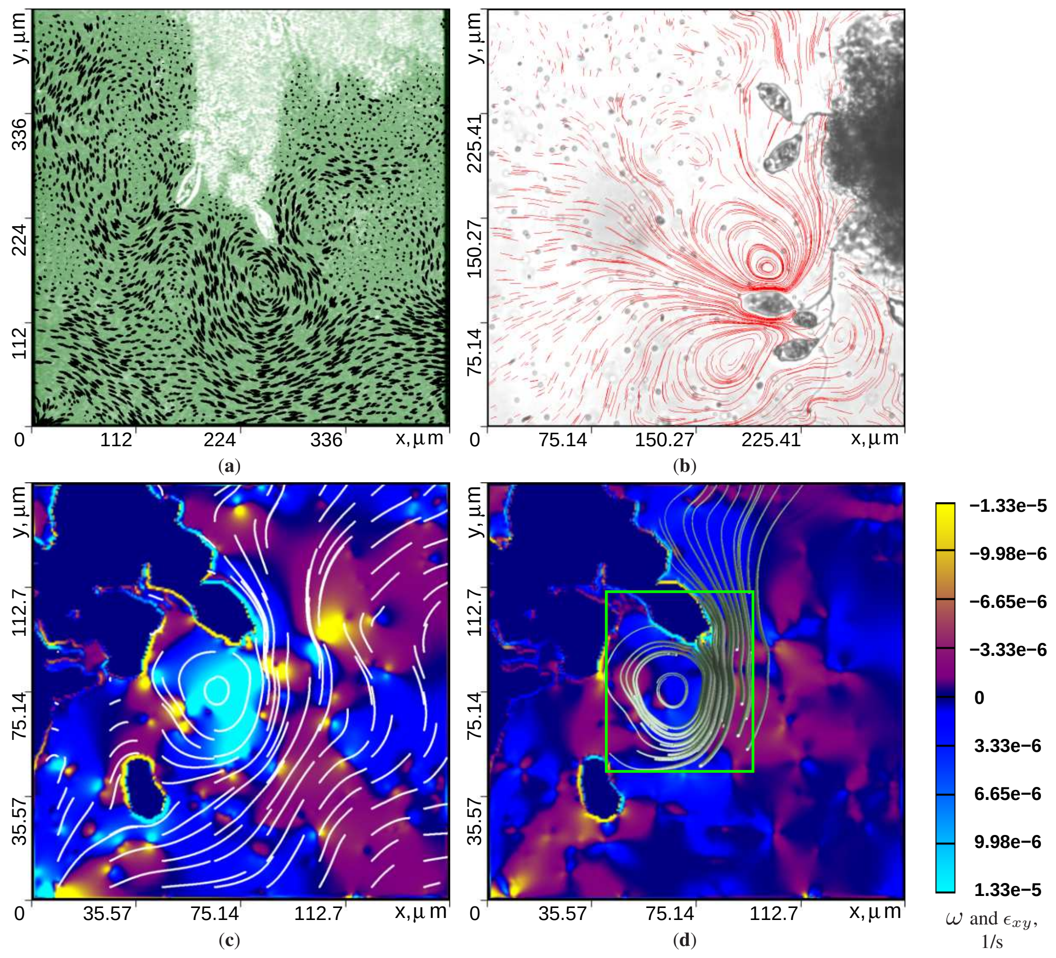


Figure 4.19: *Visualization modes available in the GPU-based framework for visual flow exploration: (**a**) particle tracing, (**b**) stream lines, (**c**) stream lines overlayed on the vorticity map, (**d**) stream balls seeded in a small probe (green square) and overlayed on the shear-strain map.*

[12] The shear-strain in the 2D case is computed as: $\epsilon_{xy} = \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}$.

Part II

Interactive Visualization of Flow Fields

Chapter 5

State-of-the-Art Techniques for Flow Visualization

Flow exploration plays an important role in research in many fields of science from industrial experiments in wind tunnels (aerodynamics) to investigation of micro-flow in blood vessels (medicine). As mentioned in Section 1.1, flow exploration encompasses three main procedures: flow measurement and reconstruction of flow fields from measured data, or, alternatively, simulation of flow fields using numerical codes of computational fluid dynamics, and visualization of flow fields. This chapter is dedicated to the latter aspect – visualization.

It is a well known fact that, except for some unique individuals, human beings are not able to gain much insight from an excessive amount of digital information presented solely in numbers. Therefore, efficient methods are required in order to visualize the numbers and to enable the observer to see the unseen. Scientific visualization is a field of science, dedicated to the problem of mapping the numbers to visual representations allowing to understand the data under investigation. A particular branch of scientific visualization suited specifically for fluid flows is *flow visualization*. It is worth mentioning that techniques used for efficient flow visualization sometimes can be successfully applied in areas which do not explicitly deal with fluid flows.

Flow visualization has a centuries-long history and probably exists as long as fluid flow research itself. Until the development of flow measurement techniques allowing for quantitative flow analysis, the only way to investigate the real flow was its direct experimental visualization. An overview of such methods along with beautiful flow pictures can be found, for example, in [Dyk82, Mer87, SLBS03]. Undoubtedly, direct flow exploration is appealing from an experimental point of view: it allows the researcher to get an impression about the flow without any calculations; the amazing flow patterns it reveals inspire the development of new theories for fluid flows, and it enables the qualitative verification of a new theory or flow model. However, there are several problems preventing the use of these methods in all applications. Firstly, flow is inevitably affected by experimental equipment in most cases. Moreover, not all parameters can be directly visualized. Finally, experimental equipment for the investigation of large scale flows, such as wind tunnels, is not affordable for every fluid mechanics laboratory. Since visualization does not require expensive equipment and does not change the natural flow conditions while providing a range of digital information about the flow, it opens new opportunities for quantitative flow exploration.

In this chapter, we briefly summarize the state-of-the-art techniques used for flow visualization. More information about the ongoing research in this area as well as a rich collection of references related to this topic can be found in [PvW93, PVH^{+}02, PVH^{+}03, LHZP05, Lar08]. Furthermore, we provide a description of an efficient implementation of fundamental flow visualization algorithms on the programmable graphics processing units (GPU), which lies at the core of our particle engine. This information is intended to help the reader understand the implementation issues described in Chapters 6 and 7 without looking through a detailed description of the particle engine given in [KKKW05]. Finally, we present an extension of the GPU particle engine for additional visualization modes allowing for feature-based focus and context visualization of flow fields.

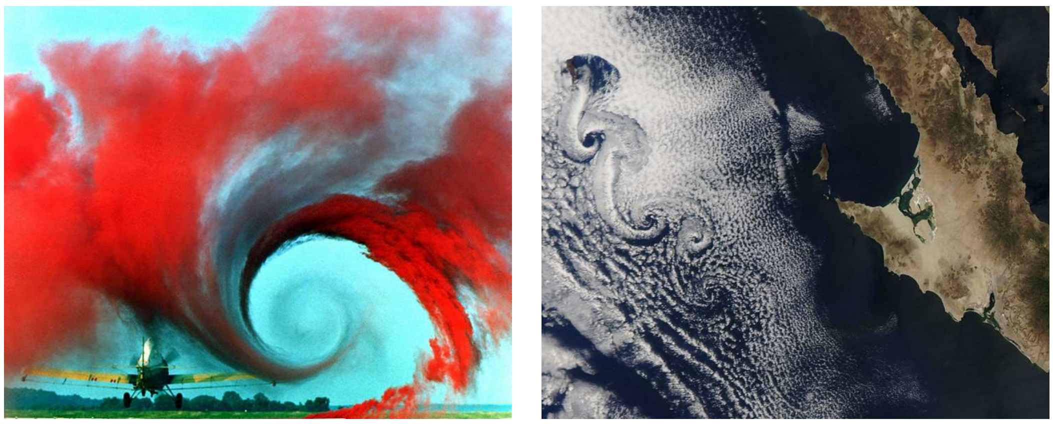

(**a**) (**b**)

Figure 5.1: *Amazing flow phenomena: (**a**) wake vortex study at Wallops Island; (**b**) the Von Karman vortex street at the coast of Baja California, Mexico.*

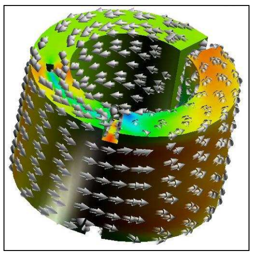
5.1 Introduction to the Visualization of Vector Fields

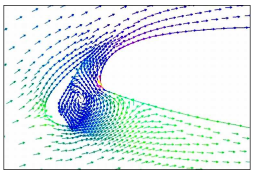
Many of the quantitative flow visualization methods developed in the last two decades draw on the early-days experimental flow visualization techniques: path lines, streak lines, and time lines. These techniques *mimic* the experimentally obtained flow pattern at the same time incorporating additional quantitative information into the visualizations. The process of flow visualization can be represented as a pipeline consisting of the following stages [PvW93]:

- *Low level data processing:* modification or selection of the data, in order to reduce the amount or improve the information content of the data. Typical examples of low level data processing are domain transformation, interpolation, sampling, and noise filtering.
- *Visualization mapping:* translation of the physical data and derived quantities to suitable visual primitives and attributes. This is the most challenging part of the process since it is not a trivial task to visualize the physical meaning, which probably has never been seen, through geometrical objects on the screen.
- *Rendering and display:* transformation of the mapped data into displayable images and their output onto the screen. Typical operations performed on this stage are viewing transformations, lighting calculations, and hidden surface removal.

Sometimes, flow data generation is considered to be a part of the first stage of the visualization pipeline [MDB87, PvW93]. However, in our terminology, the entire process including the flow field acquisition is referred to as *flow exploration*, and, in turn, the visualization pipeline is denoted as *visual flow exploration*. Thus, we will use *our* definitions to avoid any ambiguity. All currently existing approaches for quantitative flow visualization can be classified into the following groups [Lar08]:

Direct: The whole flow field is portrayed using minimum computations in the most straightforward way. Typical techniques belonging to this class are 2D plots and 3D volumes seeded with arrow-shaped primitives at given positions within the flow domain or filled up with color-coded scalar flow properties such as velocity magnitude at every pixel/voxel. The main advantages and disadvantages of this class of algorithms are:



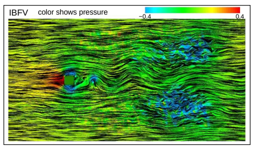


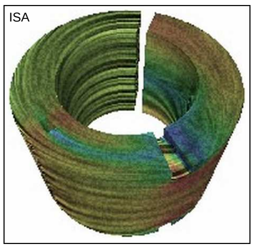
⊕ immediate and intuitive exploration of flow data in 2D,

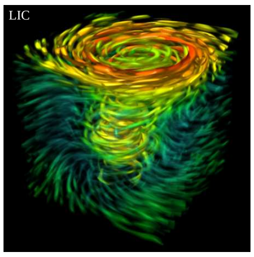
⊖ visual complexity and cluttering in 3D;

Dense, texture-based: A dense representation of the flow is generated by using the texture derived from the vector field. The direction of flow motion is taken into account via smearing of texture values along the vector field using the convolution function. The most successful techniques belonging to this class are: spot noise, line integral convolution (LIC), image space advection (ISA), and image-based flow visualization (IBFV) [LHD^{+}04]. These techniques have the following benefits and drawbacks:

- ⊕ detailed view, clear perception of flow features,
- ⊕ advantages of direct and geometric techniques,
- ⊖ time-consuming and prone to aliasing,
- ⊖ visual complexity and cluttering in 3D;

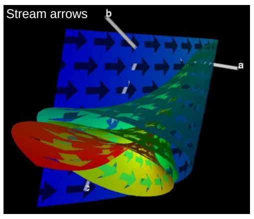


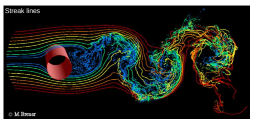


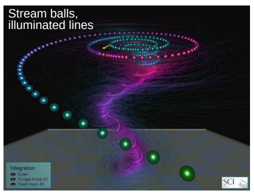


Geometric: The geometric objects are used to represent the flow and its dynamics. The flow features are first extracted via integration of the flow field and then displayed using geometric objects. Typical examples include stream lines, streak lines, and path lines. Stream surfaces and stream particles also belong to this class [WE04]. The pros and cons of geometric flow visualization are:

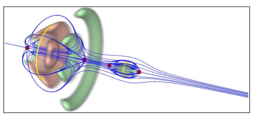
- ⊕ intuitive flow visualization,
- ⊕ clear perception of flow features,
- ⊖ efficient placement of geometric objects in both 2D and 3D is a non-trivial task,
- ⊖ visual complexity and cluttering in 3D;

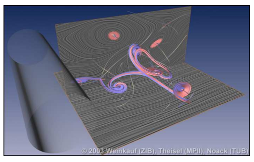






Feature-based: Core flow features such as topological skeleton and coherent structures are extracted from the flow field and used for visualization. This allows to reduce the amount of displayed information, while emphasizing the prominent structures of the flow. We will describe these techniques in more detail in Section 5.3. The main disadvantage of this class of techniques is high computational costs for feature extraction.





One important criterion for classification of visualization techniques is the *level* of displayed information. According to Hesselink et al. [HPvW94], the following three categories of methods can be distinguished. *Elementary* approaches display detailed information at a single point, without taking into account information from neighboring points. *Local* techniques incorporate information about the neighborhood of some point into a representative geometric object. Finally, *global* methods represent the whole vector field via a few primitives.

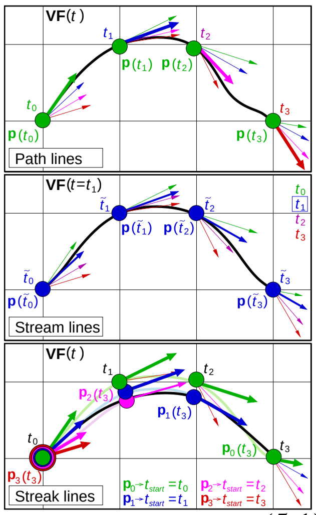
For example, arrow plots and line-type approaches are considered as elementary techniques, and stream surfaces as local [HJ04]. Since the main topological features of the vector field, critical points and separatrices, completely determine the nature of the flow, the topology-based methods fall under the category of global.

For the sake of clarity, in this thesis we will use only two classes: *local* and *global*. In this way, glyph plots are treated as local and topology-based methods as global. Classifying line-type approaches as local, in our opinion, is somewhat misleading. Since lines are computed over the region, they *do* show the structures within the flow. Thus, we will refer to lines also as global technique according to our classification.

5.1.1 Particles Tracing: Basics

Since particle tracing is at the core of our GPU particle engine, it makes sense to briefly describe the basic mathematics behind this paradigm. In Section 2.1, we have already introduced the notion of *flow lines*, or *integral curves*, or also *characteristic curves*, which are the lines traced by the particles immersed into the flow. The velocity vectors are tangent to these lines at every position in space (cf. Equation 2.2). In this case, given a velocity vector field the flow lines can be constructed via integration:

$$\begin{aligned}
\mathbf{p}_{\text{path}}(t;\, \mathbf{p_0},\, t_0) &= \mathbf{p_0} + \int_t^{t_0} \mathbf{V}\big(\mathbf{p}_{\text{path}}(s;\, \mathbf{p_0},\, t_0),\, s\big)\, \mathrm{d}s, \\
\mathbf{p}_{\text{stream}}(t;\, \mathbf{p_0},\, t_0) &= \mathbf{p_0} + \int_t^{t_0} \mathbf{V}\big(\mathbf{p}_{\text{stream}}(s;\, \mathbf{p_0},\, t_0),\, \tau\big)\, \mathrm{d}s, \\
\mathbf{p}_{\text{streak}}(s;\, \mathbf{p_0},\, t) &= \mathbf{p}_{\text{path}}(t;\, \mathbf{p_0},\, s), \quad s \in [t_{\min}, t],
\end{aligned} \tag{5.1}$$



where *path line* represents a trace left by a particle in the flow; *stream line* is a trajectory drawn in the “frozen” flow field, i.e., the instantaneous flow field at the specified time τ; and *streak line* is a pattern made by dye particles being injected into the flow for some period of time $s \in [t_{\min}, t]$ at a fixed position $\mathbf{p_0}$.

In general, a given vector field is specified at discrete grid points. Hence, numerical integration schemes should be employed to reconstruct the integral curves from this vector field. One of the simplest integration scheme is the Euler scheme of first-order. Applying this scheme to the path line formula in Equation 5.1, each position on the path line is computed as follows:

$$\mathbf{p}\,(t + \Delta t) = \mathbf{p}(t) + \mathbf{V}\big(\mathbf{p}(t)\big) \cdot \Delta t. \tag{5.2}$$

The Euler scheme is fast and simple, however, it does not always provide satisfactory results in terms of accuracy. When accuracy is a critical point, higher-order integration schemes can be exploited, e.g., Runge-Kutta of fourth-order [PTVF07].

It is worth mentioning that different rendering modes lead to different visualizations of the same integral curves. For example, if the computed positions are connected by line segments, we get a piecewise representation of an integral curve. On the other hand, if instead of line segments spheres are rendered an their size is modulated by the magnitude of velocity at corresponding positions, so called *stream balls* are depicted. Finally, if instead of displaying static line segments or balls, a single discrete particle is traced along the integral curve, an animated visualization is achieved. All of these rendering options are included in the GPU particle engine and are briefly described in the next section.

5.2 Efficient Particle Tracing on the GPU

The first version of the described particle engine on GPUs was created in 2004 inspired by progress going on at that time in many areas of science. The advent of powerful computers and the development of new numerical techniques to simulate flow as well as experimental techniques to measure flow at high speed resulted in a huge amount of flow field data sets, which require efficient means for their visual exploration. The most efficient visualization methods at that time were based on the particle tracing paradigm [PVH$^+$02] since this approach intuitively represents the flow structures in a similar way, as it is done by experimental flow visualization techniques. In order to accelerate the output of extracted flow features and to enable interaction with the data, most techniques suggest to precompute the flow properties and upload them onto the GPU for rendering [BKHJ01]. The remaining approaches exploit the efficiency of parallel architectures [BL91, ZH97] or data partitioning techniques and caching strategies for particles [USM96, CE97]. None of these techniques provide the computation of particle trajectories on the fly and tracing millions of them at interactive rates. On the other hand, due to the rapid progress of highly parallel graphics architectures, GPUs have been introduced as powerful tools not only for computer graphics purposes, but also for scientific computations [LBS03, TvW03, KSE04, KLRS04]. These observations motivated us to develop an efficient method, which reveals the flow dynamics in a most intuitive way, while at the same time computing on the fly and displaying the quantitative information about the flow properties, such as vorticity or velocity magnitude, exploiting the efficiency of parallel computations on the GPUs.

The particle engine, which we briefly describe in this section, is suited for interactive visualization of steady 3D flow fields on uniform grids. For more details, the interested reader is referred to [KKKW05]. To fulfill the requirement of interactivity, we trace all the particles in parallel on the GPU, which has been proven to be more than three times as efficient as a CPU-based solution. Moreover, the time-consuming transfer operation from CPU to GPU is avoided by implementing the particle tracing completely on graphics hardware . In this way, the proposed GPU particle engine allows for interactive streaming and rendering of millions of particles, and thus, enables virtual exploration of high resolution fields in a way similar to real-world experiments.

The ability to display the dynamics of large particle sets using visualization options like shaded points or oriented texture splats provides an effective means for visual flow analysis that is far beyond existing solutions even nowadays. For each particle, flow quantities like vorticity magnitude and the λ_2 criterion[1] can be computed and displayed on demand. Built upon a previously published GPU implementation of a sorting network, visibility ordering of semitransparent particles is implemented. To provide additional visual cues, geometric objects such as stream ribbons are constructed and displayed using the GPU.

5.2.1 Particles: Advection and Rendering

On recent programmable graphics cards it is possible to allocate and use two types of memory objects for read and write operations: *vertex buffers* and *textures* [Gra03]. It is worth mentioning that GPUs allow to store the volumetric data in 3D textures and support automatic tri-linear interpolation for such textures. The possibility to create

[1] λ_2 is the second largest eigenvalue of the matrix $\mathbf{\Omega}_{\mathrm{sym}}^2 + \mathbf{\Omega}_{\mathrm{asym}}^2$, where $\mathbf{\Omega}_{\mathrm{sym}}$ and $\mathbf{\Omega}_{\mathrm{asym}}$ are the symmetric and antisymmetric parts of the velocity gradient matrix. λ_2 can be used, e.g., for vortex detection [JH95].

custom functions (*shaders*) for vertex and fragment processing stages extends the traditional graphics pipeline and offers new opportunities to programmers as depicted in Figure 5.2. This functionality enables the construction, manipulation, and rendering of geometric data on the GPU. Hence, particle tracing can be entirely performed on the GPU without any read back to application memory.

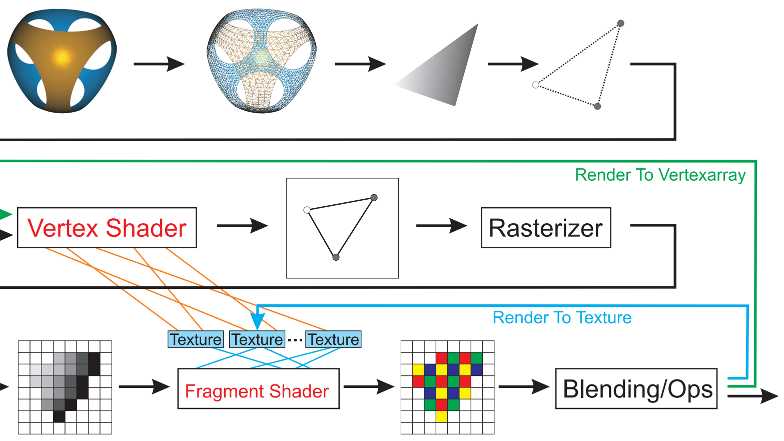


Figure 5.2: *Rendering Pipeline with Programmable GPU.*

In general, rendering on programmable GPU is performed as follows. The *vertex shader* processes the input information about the vertices of some geometric primitive stored in an associated vertex buffer, projects the vertex positions into screen space, and outputs them. In addition, a number of auxiliary parameters for each vertex can be output as well. This information is transferred to the *fragment shader*, which performs the per-fragment operations and outputs the result onto the screen.

GPU-based particle tracing employs several 2D textures of the same size $m \times n$ that store the updatable attributes for each particle $\mathbb{P}[i, j]$[2]. Moreover, a discrete steady velocity field is stored in the separate 3D texture, `VFTex3D`. The particles are traced in space according to the vector field values fetched from `VFTex3D` using the numerical integration schemes, such as the Euler scheme (cf. Equation 5.2). The computation of some particle attributes, such as position, requires the value of these attributes at the previous step. Since simultaneous read and write operations into the texture are not available on current GPUs, a *ping-pong technique* has to be employed in order to keep information from the previous step. Several attributes can be efficiently computed and stored in different textures using a *multiple render target technique* (MRT). It is worth mentioning that each texture object can have up to four 32-bit components on current graphics hardware.

[2] Since particle has many attributes we denote it with a double letter $\mathbb{P}$ in the same way as the other multivariate quantities such as matrices and tensors.

The particle tracing algorithm involves the following basic operations:

- *Birth-death-incarnation:* At birth, a randomly selected life time is assigned to each particle. This time is decreasing in each update operation, and when it becomes zero, the particle dies and gets reincarnated at its initial position stored in the starting position texture `StartPosTex` together with its starting life time. When the particle goes out of the flow domain, it also dies and gets reincarnated.
- *Advection:* In each advection step the current particle position is fetched from the position texture `PosTex`$(t-1)$ and advected. Additional particle attributes can be updated as well. The result of such update operation for position and life time is stored in `PosTex`(t), and for additional attributes in auxiliary textures.
- *Rendering:* Particles are rendered according to their positions stored in `PosTex`(t) using different visualization modes, including shading, lighting, and color-coding based on their attributes, e.g., vorticity or λ_2.

The initial particle positions can be specified using different spatial distribution functions, e.g., regular or random distributions over the rectangular seeding probe. In order to achieve the impression of homogeneously seeded particles following the flow, the life times are randomized. In this way, no "mass death" or "mass reincarnation" occur, which means that particles are appearing and disappearing randomly in such a way that it does not distract the user's attention from the moving particles. The position and size of the seeding probe can be arbitrarily modified by the user. The result of such a modification is the transformation matrix $\mathbb{M}$, which is applied to the starting particle positions. Moreover, the user can arbitrarily modify the number of particles to be traced. In this case, the size of the particle attribute textures as well as the starting particle positions have to be modified.

The update operations – including the advection for particle attributes – are performed using the *off-screen rendering* technique as follows. A stream of $m \times n$ fragments corresponding to a contiguous block of pixels in screen space is generated. The fragment output is rendered into an equally sized render target associated with the particle attribute texture at time step t. Figure 5.3 illustrates this procedure. For time-dependent attributes, each fragment $[i,j]$ gets as input the corresponding pixel $[i,j]$ from the attribute texture computed at the previous time step $(t-1)$. Different operations $\mathtt{f}(\mathtt{tex}_0,\ldots,\mathtt{tex}_\mathrm{n})$ can be performed over the fragments before they are finally output into the render target.

In the case of *advection*, it is necessary to know the current particle's velocity stored in `VFTex3D`. Thus, first of all, for each output fragment with texture coordinates $[i,j]$,

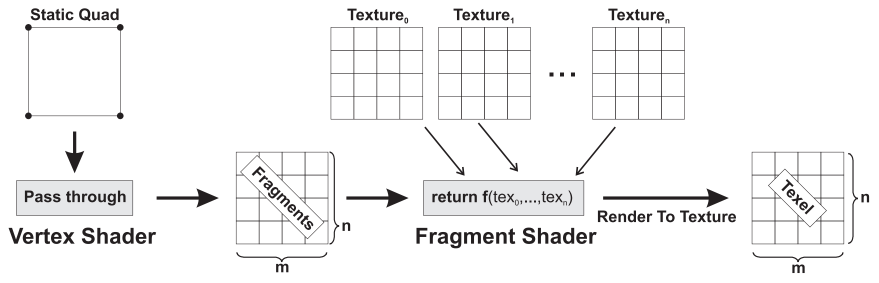


Figure 5.3: *Update of particle attributes using the off-screen rendering technique.*

the particle position is fetched from $\texttt{PosTex}[i, j]$. Then, the vector field value is looked up at this position in $\texttt{VFTex3D}[\texttt{PosTex}[i, j]]$.

It is worth mentioning that the advection step occurring directly after the particle birth and incarnation slightly differs from the subsequent advection steps. First of all, the positions, which are fetched from StartPosTex, are specified over the unit cube. In order to fit them into the user-defined seeding probe, these positions should be transformed using the transformation matrix $\mathbb{M}$ of the seeding probe. In this way, all positions are transformed into world-space, and thus, further actions can be performed in the same way as in the following advection steps.

Once all the necessary particle attributes are computed, the particles can be rendered using the capabilities provided by DirectX Shader Model 3.0 or higher [Gra03]. Each entry of the vertex buffer is associated with a single particle. Since all particle attributes are stored in textures, a special function is required which maps the 1D index identifying a particle within the vertex buffer to the 2D index related to the texture coordinates assigned to this particle within the attribute textures. To find this mapping function, the texture coordinates assigned to a particle are written into the vertex buffer entry associated with this particle. Thus, in the rendering stage, all required attributes connected to a particle have to be extracted from the attribute textures in the vertex shader and forwarded to the fragment shader, where the attribute-based color is assigned to the output vertex.

There are two general ways, how the particles can be represented: simple one-pixel points or geometric objects centered at the particle positions. The latter case can be further classified as simple glyphs without directional information and glyphs aligned along the flow to emphasize the direction of the velocity vector at the current particle position. The points are primitives, which do not require significant GPU power to render them. On the other hand, geometric objects typically consist of a number of primitives. Thus, rendering them puts considerable burden on the rendering process.

Sprite Atlas

Since using the geometric objects (meshes) allows for displaying additional important information, such as flow direction, which is not possible by using a static point, it is of interest to employ objects of different shapes in the visualization. An efficient technique to eliminate the time-consumption for geometry processing of meshes is to use the conception of *virtual geometry* represented via point sprites. Technically, a *point sprite* is a textured quad rendered at a specified position on the screen. The geometry processing time for such point sprites is significantly lower than that for real geometric objects.

Virtual geometry is a "photograph" of a real object, which can be attached to a point sprite. More precisely, virtual geometry represents a projection of a real object oriented according to the specified transformation matrix and written into the texture. A single picture obtained from a certain position does not describe a variety of possible object orientations in space. Therefore, a number of projections captured from different points of view onto the real object is stored in a *sprite atlas*.

Since the rotation about the Z-axis takes place in the screen plane, it can be obtained via rotation of texture coordinates, assigned to the fragments of the point sprite. On the other hand, rotations about X and Y axes extend beyond the screen space, and thus, in general, both of them should be taken into account. However, if the object is symmetric about the X-axis, it is sufficient to store only the rotation about the Y-axis. In this way, only one "view-dependent" row of object projections is required to recover all the possible orientations of this object.

The oriented point sprites usually have elongated shape in order to better visualize the directional information. To make the rendered object longer or shorter, scaling along its major axis is performed. This longitudinal deformation is taken into account by adding several "view-dependent" rows with different scaling factors into the sprite atlas.

For rendering the point sprites, an appropriate sub-image from the whole sprite atlas should be selected. This selection is performed in a fragment shader via the transformation $\mathbb{M}_{\text{dir}}$ of texture coordinates $\texttt{tc}[x, y]$ assigned to the point sprite by the GPU. $\mathbb{M}_{\text{dir}}$ is a 2×2 matrix computed as follows:

$$\begin{aligned} \mathbb{M}_{\text{dir}} &= \mathbb{M}_{\text{trans}} \cdot \mathbb{M}_{\text{rot}}, \\ \text{where } \mathbb{M}_{\text{trans}} &= \text{Translate}\big(\arcsin(\hat{z}),\ \|\mathbf{V}\|_2\big) \\ \mathbb{M}_{\text{rot}} &= \begin{bmatrix} \hat{x} & \hat{y} \\ -\hat{y} & \hat{x} \end{bmatrix}, \end{aligned} \tag{5.3}$$

where $\mathbf{V}$ is the velocity vector of the particle represented by a point sprite, and $(\hat{x}, \hat{y}, \hat{z})$ are the components of the normalized $\mathbf{V}$. Since for all fragments of the point sprite $\mathbf{V}$ is constant, $\mathbb{M}_{\text{dir}}$ can be computed in the vertex shader (thus, once per sprite) and passed to the fragment shader as a parameter. In this way, efficient rendering of virtual oriented glyphs is performed.

Note that all the point sprites by default have the same size, which can be programmed. To give the impression of perspective projection, this size is scaled with the z-component of the particle position projected into screen space. Examples of visualizations using point sprites are depicted in Figure 5.4.

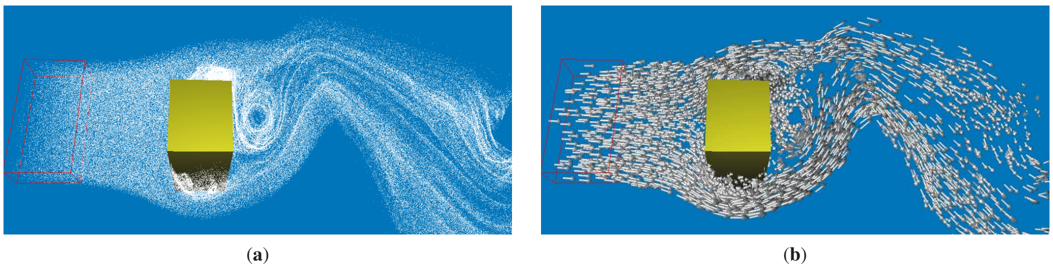


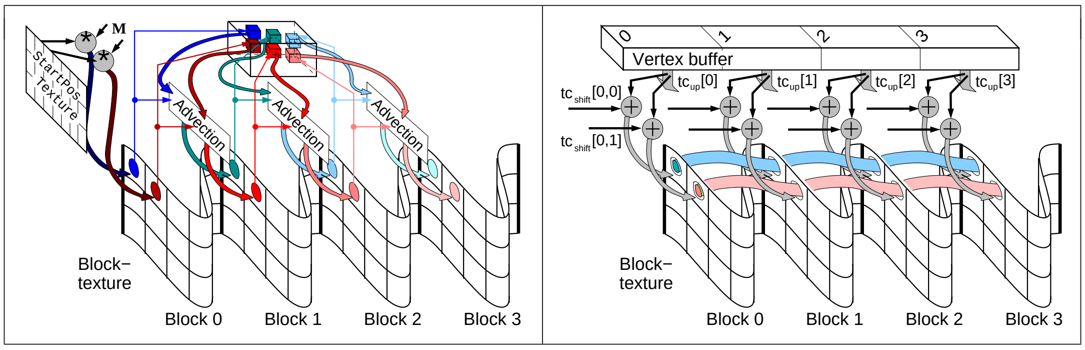
Figure 5.4: *Visualization using point sprites: (**a**) simple points, (**b**) virtually aligned point sprites.*

In order to perform correct rendering of semi-transparent particles, they should be depth-sorted first. Since sorting is a computationally expensive procedure, its efficient implementation is a critical point for maintaining the interactivity. We have implemented an efficient bi-tonic merge sorting algorithm as suggested in [KSW04]. It is worth mentioning that in many cases an *approximate* rendering without sorting using the *additive alpha blending* can be employed.

5.2.2 Lines: Tracing and Rendering

From an implementation point of view, the algorithm for tracing lines on the GPU is different than the algorithm for tracing particles. This is because in order to construct a line, *all* positions on this line must be computed. Afterwards, these positions can be connected with line segments, or some other rendering mode can be employed. In the current implementation, line tracing is performed as follows.

Assume that $m \times n$ lines of the same length l need to be traced. Line tracing can be done in parallel for all lines by updating the position for each line at each time step t and storing it into the texture `LPosTex`$[t]$ of size $m \times n$. Performing this operation l times for the full line length, a stack of l textures is obtained. Instead of storing l textures of size $m \times n$, they can be packed block-wise into one texture `FullLPosTex` of size $m \times (n \cdot l)$. In the particle tracing algorithm, the ping-pong technique is used to work around the inability of current GPUs to perform simultaneous read and write operations into the same texture. In this case, the previous positions are stored in the auxiliary texture. For lines, the previous positions are fetched from the last block stored within the block-texture `FullLPosTex` (see Figure 5.5 (a)).



(**a**) (**b**)

Figure 5.5: *Lines in the GPU particle engine: (**a**) construction, (**b**) rendering.*

Rendering of lines is performed in a different way than the rendering of particles. The vertex buffer is used to store the texture coordinates $\mathtt{tc}_{\mathtt{up}}[t]$ of the upper left corner of the t^{th} block within the block-texture `FullLPosTex`, which corresponds to the currently rendered t^{th} line segments. Since all the lines have the same length, one vertex buffer is sufficient. While rendering the lines, the cycle runs through all of them, processing in each iteration an entire line. The shift in texture coordinates $\mathtt{tc}_{\mathtt{shift}}[i, j]$ within the block for each line is computed using the 2D index $[i, j]$ of the line and passed to the vertex shader, where it is summed up with the offset of the t^{th} block within `FullLPosTex` (see Figure 5.5 (b)).

In addition to standard lines, advanced line-rendering techniques, such as stream ribbons and shaded lines, are implemented in our particle engine as well. If point sprites are rendered instead of line segments, the appearance of stream balls is achieved. Different line-based visualizations are shown in Figure 5.6.

5.2.3 Extension for Unsteady Flow Fields

It is worth mentioning that we have extended our particle engine for the interactive visualization of large unsteady 3D flow fields on uniform grids [BSK+07]. For this purpose, a novel dual-core approach has been proposed in order to asynchronously stream such fields from the CPU, thus enabling the efficient visual exploration of large time-resolved sequences. This approach decouples visualization from data handling,

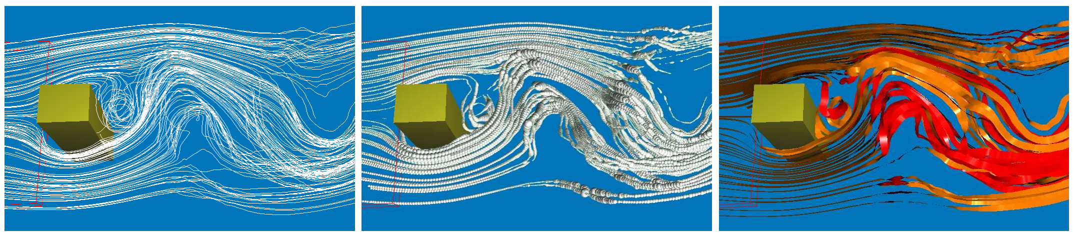


Figure 5.6: *Visualization using lines: (**a**) simple stream lines, (**b**) stream balls, (**c**) stream ribbons.*

resulting in interactive frame rates. Moreover, new strategies for the visualization of integral curves have been developed. To provide additional visual cues, GPU-based volume rendering of scalar flow features as well as focus and context techniques for polygonal meshes have been integrated. The efficiency of the proposed techniques has been demonstrated via the visual analysis of the Terashake 2.1 earthquake simulation data [BSK+07].

5.3 Feature-Based Extensions for the GPU Particle Engine

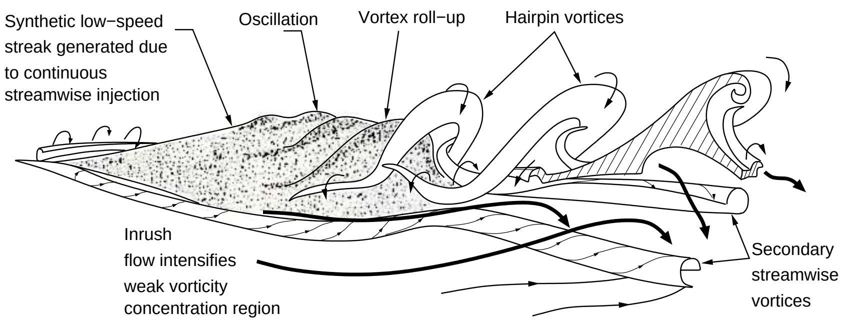

Synthetic low−speed streak generated due to continuous streamwise injection
Oscillation
Vortex roll−up
Hairpin vortices
Inrush flow intensifies weak vorticity concentration region
Secondary streamwise vortices

Figure 5.7: *Flow features example: schematic of the break-up of a synthetic low-speed streak generating hairpin vortices [AS87].*

While traditional techniques for flow visualization such as path lines or oriented glyphs visualize the vector field in a simple way, feature-based techniques raise to a higher level of abstraction, by extracting physically meaningful structures from the underlying flow field. Based on these features only the important and interesting information is depicted to the user. Thus, the amount of data displayed at once can be significantly reduced. As a consequence, the visual perception of the characteristic flow structures is improved. While studying this abstract definition, two reasonable questions arise: what is the feature and which regions should be considered as “interesting” or “important”?

Taken from the dictionary, the definition of a *feature* is as follows: a prominent or distinctive aspect, quality, or characteristic. In this way, in fluid flows the vortices, vortex cores, separation regions, heterogeneous regions, critical points, etc. can be considered as features. After these features are detected, the regions where they appear are treated as “important”.

A fundamental method of feature-based visualization is *selective visualization* [van95]. The feature extraction pipeline lying at the core of this method is depicted in Figure 5.8.

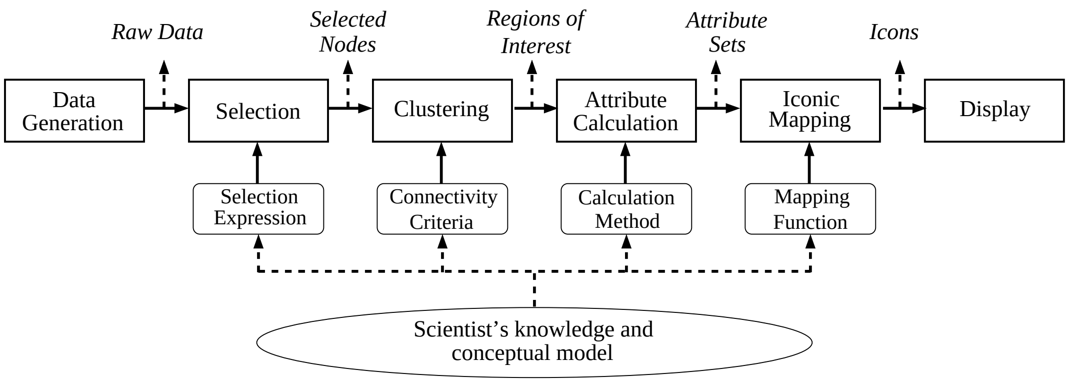


Figure 5.8: *The feature extraction pipeline.*

The *selection step* corresponds to a binary segmentation of the original data set based on some criterion (threshold). If the voxel is selected according to this threshold, the corresponding bit in the bit-mask is set to “1”, otherwise to “0”. In general, a number of thresholds can be used. Another way to select data is to allow the user to do this according to his/her needs [FG98]. Moreover, to achieve a smooth visual appearance in transition regions between the selected and unselected voxels a transparency blending function can be employed. In the *clustering step* all selected points are combined into coherent regions [HWHJ99], which are considered as objects. The attributes, such as position, volume, and orientation, are computed for these objects in the *attribute calculation* step. Finally, the calculated properties are depicted using some geometric objects (*iconic mapping step*). In this way, all the unselected/uninteresting regions can be automatically discarded.

Exploiting powerful importance-driven approaches, such as the *focus and context* techniques, which were recently adopted for flow visualization, the unimportant regions can be rendered as a context information, e.g., using volume rendering. This enables the visualization of connections and transitions between the different regions within the flow domain and helps to understand the flow under investigation in all its integrity.

Other importance-driven approaches allow for reduction of the amount of information based on varying the density of the displayed primitives [TvW99, MTHG03], or their shape [DH02]. In order to increase the effectiveness of these methods even more, additional depth cues, such as fogging, motion parallax, depth-of-field, halos, shading and lighting, can be employed [PVH$^+$02].

Inspired by the efficiency of importance-driven techniques employed for the visualization of complex data sets from various scientific areas [VGB$^+$05, BSK$^+$07], we adopted these techniques for interactive flow visualization. In the next sections we briefly describe which extensions we have made in the particle engine in order to enable the interactive and intuitive feature-based visual exploration of flow fields.

5.3.1 Feature-Based Techniques for the Particle Engine

To enable focus and context techniques in particle-based flow visualization, we use the following two strategies. Firstly, the spatial density of the visualized particles is adapted according to the importance classification. Secondly, the appearance of the rendered particle primitives reflects the importance of the region they are traveling through. In the following, we describe how to adjust the shape and appearance of the rendered particle primitives based on their importance.

Scale-Space Particles

This method exploits the spatial importance function $\boldsymbol{Imp}(\mathbf{p})$, which is defined by a combination of a spatial attenuation function $\boldsymbol{att}(\mathbf{p}, \mathbf{p_f})$ decreasing with the distance to a user-specified focus point $\mathbf{p_f}$ and an importance scalar volume `impVol` containing some arbitrary importance measure:

$$\boldsymbol{Imp}(\mathbf{p}) \mapsto [0,1] = \boldsymbol{att}\left(\mathbf{p}, \mathbf{p}_{\mathrm{f}}\right) \oplus \texttt{impVol}\left(\mathbf{p}\right), \tag{5.4}$$

where $\oplus$ is an operator combining values of the functions $\boldsymbol{att}(\mathbf{p}, \mathbf{p_0})$ and $\boldsymbol{impVol}(\mathbf{p})$[3]. It is assumed, that "0" corresponds to unimportant regions and "1" to important regions.

In the rendering stage the number of rendered particle primitives is reduced based on the values of $\boldsymbol{Imp}(\mathbf{p})$. For this purpose an approach similar to illustrative volume rendering [YC04] is employed. A random value $\texttt{rnd}[i,j] \in [0..1]$ is assigned to every particle $\mathbb{P}[i,j]$ seeded into the flow at position $\mathbf{p}[i,j]$. This random value is used as a selection threshold. In the current implementation, a particle is rendered if $\boldsymbol{Imp}\left(\mathbf{p}[i,j]\right) > \texttt{rnd}[i,j]$. Thus, more and more particles are removed with decreasing

[3] The value at any position $\mathbf{p}$ in a scalar volume `impVol` can be obtained using trilinear interpolation.

importance. To enable clear distinction between the important and unimportant regions, as well as to emphasize characteristic flow structures, additional modulation operations are performed on the particle size ($size$), color (col), and transparency ($transp$). The overall modulation function is specified in Figure 5.9.

```
Modulate($[i,j]$, rnd$[i,j]$, p$[i,j]$, $size_0$, $C_{\text{size}}$, $C_{\text{transp}}$) {
    if($\boldsymbol{Imp}(\mathbf{p}[i,j]) > \text{rnd}[i,j]$) {                    // Selection
        transp$[i,j]$ = $(1 - \boldsymbol{Imp}(\mathbf{p}[i,j])) \cdot C_{\text{transp}}$;      // Transparency modulation
        size$[i,j]$   = $size_0 + (1 - \boldsymbol{Imp}(\mathbf{p}[i,j])) \cdot C_{\text{size}}$;   // Size modulation
        col$[i,j]$    = LUT[transp$[i,j]$];                          // Color modulation
    } else
        discard;                                                 // Do not render
}
```

Figure 5.9: *The overall modulation function for focus and context importance-based rendering.*

In Figure 5.9, $size_0$ is the basis size for particles, C_{size} and C_{transp} are the user-defined thresholds specifying how fast particles are vanishing and fading out according to the decreasing importance. Finally, the color of every particle can be modulated by means of a user-defined color transfer function LUT$\left[\text{transp}[i,j]\right]$. An example of a flow field visualization using the overall modulation function is shown in Figure 5.10 (a).

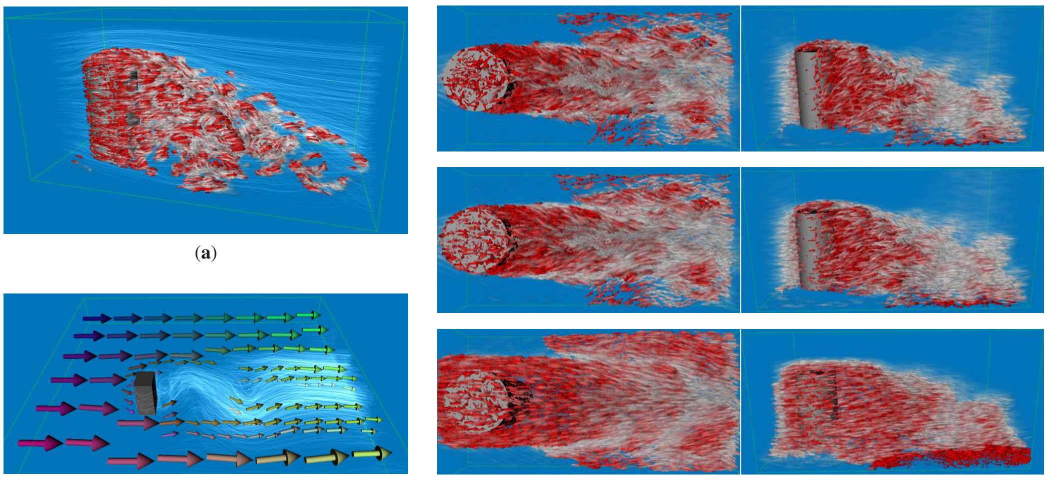


Figure 5.10: *Importance-driven feature-based visualization: (**a**) scale-space particles; (**b**) cluster arrows; (**c**) two views of velocity magnitude as importance measure shown at different resolution levels, where coarser levels are from top to bottom, red-opaque primitives correspond to low and white-transparent ones to high velocity magnitudes.*

Feature-Based Measures

To enable the intuitive visualization of flow features, we have integrated a number of different importance measures based on physical properties of the flow. In addition to the velocity and its magnitude, the following scalar quantities of the vector field $\boldsymbol{\mathcal{F}} = (F_x, F_y, F_z)^T$ are used:

- *Vorticity magnitude* – the magnitude of the flow circulation:

$$\begin{aligned}\boldsymbol{\omega} &= (\omega_x, \omega_y, \omega_z)^T \\ &= \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right)\hat{\mathbf{x}} + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right)\hat{\mathbf{y}} + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\hat{\mathbf{z}}, \\ \omega &= \|\boldsymbol{\omega}\|_2 \end{aligned} \tag{5.5}$$

- *Helicity density* – the extent to which corkscrew-like motion occurs:

$$h = \boldsymbol{\omega} \bullet \mathbf{F} = \omega_x F_x \ + \ \omega_y F_y \ + \ \omega_z F_z \tag{5.6}$$

- *Streamwise vorticity* – the component of the vorticity $\boldsymbol{\omega}$ that is parallel to the mean velocity vector $\widetilde{\mathbf{F}}$:

$$w_s = \boldsymbol{\omega} \bullet \widetilde{\mathbf{F}} / \|\widetilde{\mathbf{F}}\|_2 \tag{5.7}$$

- *Maximum Finite-Time Lyapunov Exponent* – the maximum rate of separation of particles in a dynamic system (see Section 6.1 for more details on this quantity):

$$\boldsymbol{\sigma}_{t_0}^{\Delta t}(\mathbf{p}) = \ln \sqrt{\boldsymbol{\lambda}_{\mathbf{max}}(\mathbb{C})} \ / \ |\Delta t|$$

All these measures are hierarchically encoded in a pyramidal data structure. We use a $min-max$ and an $average$ pyramids of volumes, where the first level is of the size of the original vector field and each successive volume is reduced by a factor of two in each dimension. The type of the importance measure stored in the pyramid and the level that should be considered as `impVol` can be specified by the user. By using linear interpolation between the levels of the hierarchy, such a multi-resolution data representation allows to define the importance measure at arbitrary degree of fidelity (see Figure 5.10 (c)).

Cluster Arrows

In order to further improve the importance-driven visualization mode, we have introduced *cluster arrows*, a sparse and static visualization metaphor. Cluster arrows are geometric primitives that represent dynamically *similar* regions in the flow. The similarity criterion is based on angular difference $\varepsilon_{\text{ang}}[i,j]$ between the velocity $\mathbf{V}[i,j]$ and the average velocity $\widetilde{\mathbf{V}}$ within the region $\mathfrak{N}_n$ of n voxels:

$$\begin{aligned}\varepsilon_{\text{ang}}[i,j] &= \arccos\left[\left(\frac{1}{n}\sum_{\{i,j\}\in\mathfrak{N}_n}\mathbf{V}[i,j]\right)\bullet\mathbf{V}[i,j]\right] \\ &= \arccos\left(\widetilde{\mathbf{V}}\bullet\mathbf{V}[i,j]\right). \qquad (5.8)\end{aligned}$$

The centers of the clusters are computed in a preprocess step using a region growing approach. To find a cluster, i.e., the region satisfying the condition $(\varepsilon_{\text{ang}} < thresh_{\text{ang}}\ \forall\{i,j\} \in \mathfrak{N}_{n=26})$, we randomly select a grid point that has not yet been processed, and we inspect the velocities of all of its 26 neighbors ($\mathfrak{N}_{26}$) in the grid. If none of the velocities $\mathbf{V}[i,j] \in \mathfrak{N}_{26}$ diverges more than a given angle $thresh_{\text{ang}}$ from the average $\widetilde{\mathbf{V}}$ of all velocities in that region, we continue to grow the cluster until no further expansion is possible. The average velocity $\widetilde{\mathbf{V}}$ for all grid points within the determined region is stored as a representative for this region. The process is continued until the entire domain is partitioned into clusters.

During rendering, $\widetilde{\mathbf{V}}$ is used to draw an oriented geometric primitive, scaled according to the cluster size. A minimum and maximum size of the clusters can be chosen by the user. The cluster information is also used as an additional importance metric for the rendered particle primitives. Therefore, for every sample point in the grid we store the size of the corresponding cluster, and we fade out rendering primitives with increasing cluster size as demonstrated in Figure 5.10 (b).

5.3.2 Rendering Issues for Focus and Context Visualization

In addition to the overall modulation function `Modulate(...)` described in Figure 5.9, we have proposed an efficient approach – *particle morphing* – which allows to adjust the shape of the primitives according to the importance of the regions, in which they are rendered. To enable this visualization mode, we have extended the sprite atlas described in Section 5.2.1 by the form factor. More precisely, we build q of these atlases, where q is the number of prototype glyphs of a particular shape. Each atlas is stored in a single slice of a 3D texture.

To continuously morph from one primitive into another one, we interpolate between the respective views of both primitives using 3D texture interpolation (see Figure 5.11 (a)). The color and transparency of the interpolated views can be further modulated using the transfer function `Modulate(...)` defined in Figure 5.9. It should be mentioned here that the proposed technique can only be used if the views of two primitives that are morphed into each other are stored in successive 3D texture slices. As the morphing in the current implementation is always in a strict order, this requirement does not pose any limitations. An example of a visualization using the particle morphing technique is shown in Figure 5.11 (b, c).

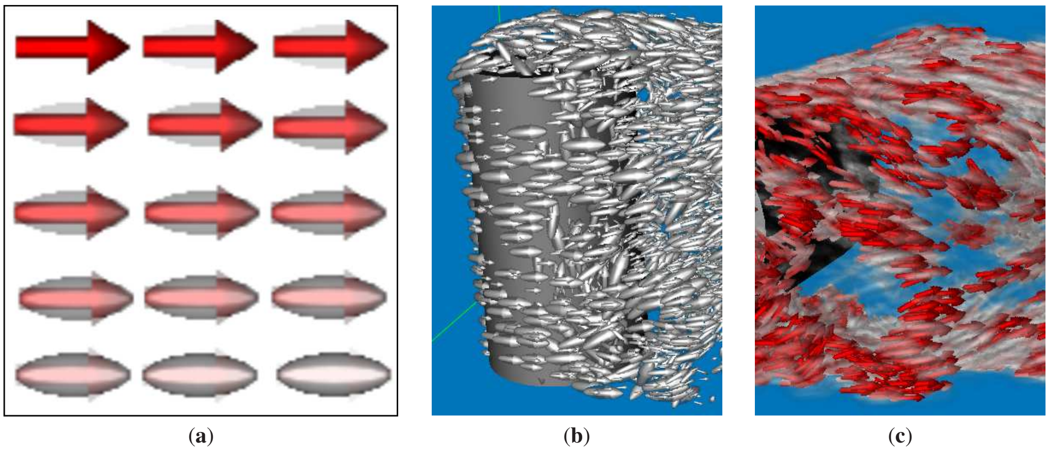


Figure 5.11: *Particle morphing. (**a**) Image-based morphing from an arrow into an ellipsoid. Examples of visualizations using: (**b**) pure particle morphing, (**c**) particle morphing combined with overall modulation function.*

It is worth mentioning that the proposed technique does not result in any noticeable artifacts, if the prototype objects have *similar* geometries. Additionally, the high visual quality is achieved because we first blend between two views and then perform the scaling of the result to the adequate size of the primitives. Furthermore, it is clear that we can easily build additional atlases for *arbitrary* primitives in-between the given basic shapes and store them in a 3D texture. This will yield even more flexibility in selecting particular shapes and their appearance in regions that cannot be clearly classified in terms of importance and unimportance.

Chapter 6

Flow Visualization with Anchor Lines

As shown in Chapter 5, the combination of a particle tracing method with feature-based focus and context techniques provides an efficient means for intuitive 3D visual flow exploration enabling the extraction of local flow features while still revealing the global flow information. However, in complex 3D flow scenarios, such as turbulent flows, even these sophisticated rendering techniques with importance-blending options still deliver an excessive amount of visual information and thus do not allow to discover the interesting formations among the other densely interweaved and possibly unimportant flow structures. To overcome this problem, we propose *anchor lines*, a new focusing strategy for particle tracing that allows to enhance characteristic flow structures based on spatial attraction and separation of particles within the flow.

The idea behind anchor lines stems from the observation that one is typically not interested in a detailed visualization of flow regions in which the seeded particles do not diverge. In order to describe the flow structure in such regions, it is sufficient to render only a few representative primitives. On the other hand, it is of interest to highlight the regions with more complex behavior of particle traces, for instance, at separatrices or at critical points of the flow fields, such as saddles or sources, where the trajectories of the neighboring particles significantly deviate from each other over time.

In order to define the degree of divergence of particle traces, we use the *finite-time Lyapunov exponent* – a scalar quantity that measures the rate of separation of infinitesimally close particles in the flow. This quantity is used to guide the seeding of anchor lines. The local flow features in the regions of interest are emphasized by accompanying the anchor line with glyphs, rendered only when the corresponding particles (*accompanying particles*) leave the anchor, thus allowing for substantial reduction of visual information depicted at once. The proposed anchor line technique is incorporated into the GPU-based particle engine framework described in Section 5.2. The effectiveness of this tool for the visualization of 3D flow fields is demonstrated using synthetic as well as real simulation data.

6.1 Finite-Time Lyapunov Exponent

The notion of the *Lyapunov exponent* is inherited from the theory of dynamical systems, where it is used to study the asymptotic behavior of infinitesimally close particles in time-independent systems [Lia66, BP02]. At this point, let us remind that the fluid flows can be also considered from the perspective of dynamical systems (cf. Section 2.1.3).

Motion of particles in dynamical systems is often studied in terms of the trajectories they trace in phase space [Wig92]. In application to fluid flows, this approach is referred to as *Lagrangian*, in contrast to the *Eulerian* point of view traditionally used in fluid dynamics, which considers every quantity at a fixed point in the spatial and temporal domain (cf. Section 2.1.1). Interestingly, even for the simple case of a velocity field, the solution of an ordinary motion equation $\dot{\mathbf{p}}_t(t) = \mathbf{V}\big(\mathbf{p}(t),\, t\big)$ (cf. Equation 2.2), can be described by quite complex and seemingly chaotic trajectories.

In general, a velocity field in fluid dynamics is only known over a finite time interval. This is the case, for example, for both flow fields numerically simulated according to the Navier-Stokes equations and experimentally measured flow fields. The time-dependent flow fields are typically characterized by the emergence of pronounced flow structures, which modify the particle trajectories. These suddenly appearing flow formations are often referred to as *coherent structures*. In time-independent systems, coherent structures correspond to unstable and stable manifolds, which attract and repel the particles, respectively (see Figure 6.1 (a)). Since coherent structures divide dynamically different regions in the flow and reveal the global flow geometry, they allow for quantitative and qualitative analysis of time-dependent flow fields. It is worth noting that in time-dependent systems, coherent structures themselves vary in time.

Due to its asymptotic nature, the classical Lyapunov exponent is only suited for studying time-independent systems, which are defined over an infinite period of time. In many practical applications, however, time-dependent dynamical systems, especially unsteady flow fields, are typically observed over a limited period of time, beyond of which their behavior is unknown. The analysis of such dynamical systems is enabled by using the finite version of the Lyapunov exponent, i.e., the *Finite-Time Lyapunov Exponent* (FTLE). In particular, the FTLE can be used to find coherent structures (also called *separatrices*) in unsteady flows [SLM05]. The effectiveness of the FTLE fields[1] for the investigation of arbitrary instationary dynamical systems is demonstrated, e.g., in [VHG02, ISM05, LCM+05].

6.1.1 Theoretical Aspects

Before giving the definition of the FTLE, let us first slightly rewrite the flow field definitions given in Section 2.1, taking into account the initial conditions: particle position $\mathbf{p}_0$ and the starting time of observation t_0. Thus, Equation 2.2 becomes:

$$\begin{cases} \dot{\mathbf{p}}(t;\, t_0, \mathbf{p}_0) & = & \mathbf{V}\big(\mathbf{p}(t;\, t_0, \mathbf{p}_0),\, t\big), \\ \mathbf{p}(t_0;\, t_0, \mathbf{p}_0) & = & \mathbf{p}_0. \end{cases} \tag{6.1}$$

Note that according to Equation 6.1, the particle trajectory is a function of its initial position $\mathbf{p}_0$, initial time t_0, and time t. Given a position $\mathbf{p}_0$ in 3D space, initial time t_0, and final time t_1, Equation 2.4 describing the flow map can be rewritten as follows:

$$\phi_{t_0}^{t_1} : \mathfrak{R}^3 \to \mathfrak{R}^3 : \mathbf{p}_0 \mapsto \phi_{t_0}^{t_1}(\mathbf{p}_0) \;=\; \mathbf{p}(t_1;\, t_0, \mathbf{p}_0). \tag{6.2}$$

According to theorems on *local existence and uniqueness of solutions* [Arn92] of Equation 6.1, the flow map $\phi_{t_0}^{t_1}$ has the following properties:

$$\begin{cases} \phi_{t_0}^{t_0}(\mathbf{p}) & = & \mathbf{p}, \\ \phi_{t_0}^{t_1+\Delta t}(\mathbf{p}) & = & \phi_{\Delta t}^{t_1+\Delta t}\big(\phi_{t_0}^{\Delta t}(\mathbf{p})\big) = \phi_{t_1}^{t_1+\Delta t}\big(\phi_{t_0}^{t_1}(\mathbf{p})\big). \end{cases} \tag{6.3}$$

The Finite-Time Lyapunov Exponent

The motivation to use the FTLE for the computation of coherent structures is based on the following observations. The FTLE is a scalar value which characterizes the average separation between the trajectories over the time interval $(t_1 - t_0)$. On the other hand, analyzing the divergence between the particle trajectories over time allows to define the dynamically distinct regions within the flow, and thus the corresponding separation borders, which essentially are the coherent structures. The integrated nature of the FTLE is especially important for unsteady flow fields, because it allows to reveal the actual transport behavior. In contrast, the stream lines, which are drawn over the instantaneous vector field, can significantly deviate from the real particle paths.

[1] The FTLE varies as a function of space and time, thus it can be regarded as FTLE *scalar field*.

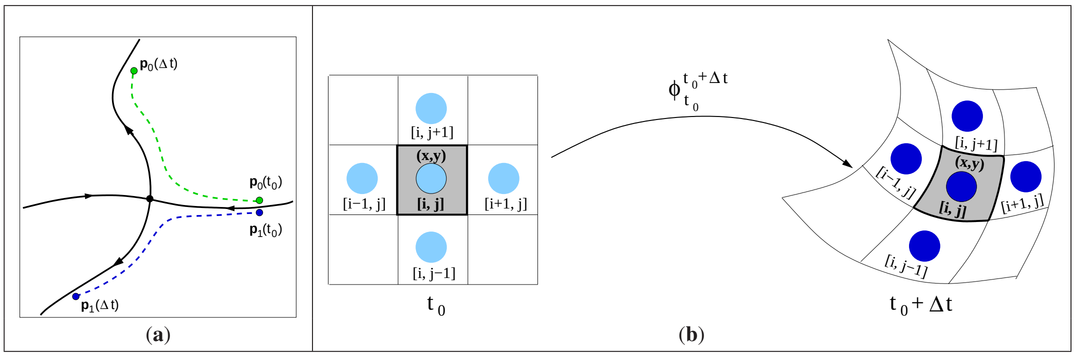


Figure 6.1: *Illustrations to separatrices and trajectory stretching: (**a**) two points on either side of a stable manifold; (**b**) Notation used for gradient* $\frac{d\phi_{t_0}^{t_0+\Delta t}(\mathbf{p})}{d\mathbf{p}}$ *computation (2D case).*

Theoretically, the phenomenon of separation of particle trajectories over time is described in terms of amount of *stretching about the trajectory*. In order to understand the notion of stretching about the trajectory, let us imaging a particle traveling from its initial position $\mathbf{p}$ at time t_0 to its destination $\phi_{t_0}^{t_0+\Delta t}(\mathbf{p})$, where it arrives after time interval Δt. While it moves, the particle gets perturbed by an arbitrarily oriented infinitesimal distortion $\delta\mathbf{p}(t)$ (see Figure 6.1 (b)). At time moment t_0 the perturbed point is described as $\bar{\mathbf{p}} = \mathbf{p} + \delta\mathbf{p}(t_0)$. After a time interval Δt the perturbation becomes:

$$\begin{aligned} \delta\mathbf{p}(t_0 + \Delta t) &= \phi_{t_0}^{t_0+\Delta t}(\bar{\mathbf{p}}) - \phi_{t_0}^{t_0+\Delta t}(\mathbf{p}) = \\ &= \frac{d\phi_{t_0}^{t_0+\Delta t}(\mathbf{p})}{d\mathbf{p}}\delta\mathbf{p}(t_0) + \mathcal{O}\big(\|\delta\mathbf{p}(t_0)\|^2\big), \end{aligned} \tag{6.4}$$

where the remainder term from the Taylor series expansion $\mathcal{O}\big(\|\delta\mathbf{p}(t_0)\|^2\big)$ can be neglected. Thus, the magnitude of the perturbation can be defined using the standard Euclidean norm as follows:

$$\|\delta\mathbf{p}(t_0 + \Delta t)\|_2 = \sqrt{\left\langle \delta\mathbf{p}(t_0), \left[\frac{d\phi_{t_0}^{t_0+\Delta t}}{d\mathbf{p}}\right]^{\mathrm{T}} \cdot \left[\frac{d\phi_{t_0}^{t_0+\Delta t}}{d\mathbf{p}}\right] \delta\mathbf{p}(t_0) \right\rangle}, \tag{6.5}$$

where $[\cdot]^{\mathrm{T}}$ denotes the transpose operation of $[\cdot]$. The symmetric matrix

$$\mathbb{C} = \left[\frac{d\phi_{t_0}^{t_0+\Delta t}}{d\mathbf{p}}\right]^{\mathrm{T}} \cdot \left[\frac{d\phi_{t_0}^{t_0+\Delta t}}{d\mathbf{p}}\right] \tag{6.6}$$

is the *finite-time* Cauchy-Green deformation tensor. Note that $\mathbb{C}$ is a function of t_0, Δt, and $\mathbf{p}$. These dependencies are omitted only for the sake of notational brevity.

If the perturbation $\delta\mathbf{p}(t_0)$ is aligned with the eigenvector corresponding to the largest eigenvalue $\boldsymbol{\lambda}_{\mathbf{max}}$ of $\mathbb{C}$, the *maximum stretching* is taking place. Thus, if we treat $\boldsymbol{\lambda}_{\mathbf{max}}$ of $\mathbb{C}$ as an operator, i.e., $\boldsymbol{\lambda}_{\mathbf{max}}(\mathbb{C})$, then the condition of maximum stretching can be written as:

$$\begin{aligned}\max_{\delta\mathbf{p}(t_0)} \|\delta\mathbf{p}(t_0+\Delta t)\|_2 &= \sqrt{\left\langle \overline{\overline{\delta\mathbf{p}}}(t_0),\ \boldsymbol{\lambda}_{\mathbf{max}}(\mathbb{C})\cdot\overline{\overline{\delta\mathbf{p}}}(t_0)\right\rangle} \\ &= \sqrt{\boldsymbol{\lambda}_{\mathbf{max}}(\mathbb{C})}\cdot\|\overline{\overline{\delta\mathbf{p}}}(t_0)\|_2, \qquad (6.7)\end{aligned}$$

where $\overline{\overline{\delta\mathbf{p}}}(t_0)$ is the perturbation aligned with the eigenvector associated with $\boldsymbol{\lambda}_{\mathbf{max}}(\mathbb{C})$. Equation 6.7 can be rewritten as:

$$\max_{\delta\mathbf{p}(t_0)} \|\delta\mathbf{p}(t_0+\Delta t)\|_2 = \exp\big(\boldsymbol{\sigma}_{t_0}^{\Delta t}\cdot|\Delta t|\big)\cdot\|\overline{\overline{\delta\mathbf{p}}}(t_0)\|_2, \qquad (6.8)$$

where

$$\boldsymbol{\sigma}_{t_0}^{\Delta t}(\mathbf{p}) = \ln\Big(\sqrt{\boldsymbol{\lambda}_{\max}(\mathbb{C})}\Big)\ /\ |\Delta t| \qquad (6.9)$$

is the largest finite-time Lyapunov exponent with a finite integration time Δt and associated to a point $\mathbf{p}$ at time t_0. Note that the $|\cdot|$ operator gives the possibility to perform the computations with both positive and negative time intervals Δt. For more detailed information about the FTLE and its application the reader is referred to [Hal01, SLM05, SP07].

Example of the FTLE Field

In order to illustrate the relation between the FTLE field and the particle trajectories, we will demonstrate a simple example. First, let us consider a generic saddle point and its associated stable and unstable manifolds (see Figure 6.1 (a)). If we integrate two neighboring points that are initially on either side of a stable manifold forward in time, then these points will eventually diverge from each other. Likewise, if we follow two neighboring points placed on either side of an unstable manifold, then these points will separate from each other, if we integrate backward in time. In this way, both stable and unstable manifolds serve as separatrices, i.e., they *separate* the qualitatively different trajectories.

At this point, let us consider a particular vector field in 2D and qualitatively describe its FTLE field. A simple flow scenario, showing two counter-rotating vortices (*double gyre* in [SLM05]) is depicted in Figure 6.2 (a). As can be observed from the picture, the trajectories of the neighboring particles that started near to the axis of symmetry between the two vortices diverge most. Taking into account the definition of the FTLE,

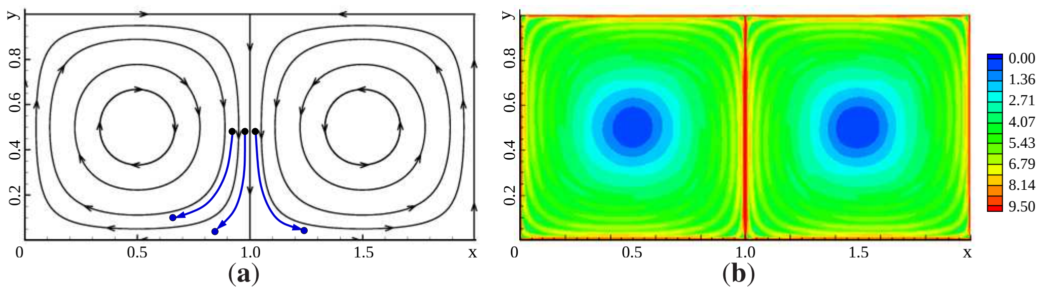


Figure 6.2: *FTLE for two counter-rotating vortices vector field: (**a**) stream lines of the vector field; (**b**) computed FTLE field.*

given by Equation 6.9, we can conclude that the FTLE field has the largest values along the axis of symmetry between the two vortices (see Figure 6.2 (b)).

6.1.2 Implementation Issues

This section provides a detailed algorithmic description of the FTLE computation, and discusses several problems we have encountered and solved on the implementation stage. The pseudo-code of the algorithm for the computation of the FTLE field from a given velocity field is presented in Figure 6.3.

```
Compute_FTLE_field($VF_{t0}$, ... , $VF_{t_0+\Delta t}$; $\Delta t$; $steps$) {
1.  Initiate particle positions on Cartesian grid:  $\mathbf{p}(t_0)$;
2.  $\gamma = \Delta t / steps$;
3.  // Compute flow map : $\phi_{t_0}^{t_0+\Delta t}$
4.  for( $t = t_0$;  $t < t_0 + \Delta t$;  $t = t + \gamma$ )
5.      $\phi_{t_0}^{t+\gamma}$ :  $\mathbf{p}(t+\gamma)$  =  AdvectParticles($\mathbf{p}(t)$; $VF_t$)
6.  Compute gradient of $\phi_{t_0}^{t_0+\Delta t}$:   $\frac{d\phi_{t_0}^{t_0+\Delta t}}{d\mathbf{p}}$;
7.  Compute Cauchy-Green tensor:  $\mathbb{C}$;                   // Equation 6.6
8.  Compute maximum eigenvalue of $\mathbb{C}$:  $\lambda_{\mathbf{max}}(\mathbb{C})$;
9.  Compute FTLE: $\sigma_{t_0}^{\Delta t}$;                          // Equation 6.9
}
```

Figure 6.3: *Algorithm for the computation of the FTLE field.*

As can be seen from Figure 6.3, the algorithm for the computation of the FTLE field is conceptually quite simple. However, several implementation aspects require particular attention. In the following, we will describe the most crucial issues regarding the computation of the FTLE fields in more detail.

Flow Map Computation

The most time consuming part is the computation of the flow map $\phi_{t_0}^{t_0+\Delta t}$ over the specified time interval Δt. In order to evaluate $\phi_{t_0}^{t_0+\Delta t}$, the particles should be advected from their initial positions $\mathbf{p}(t_0)$ along the flow field to their final positions $\mathbf{p}(t_0 + \Delta t)$. This can be done, using standard numerical integration schemes, i.e., Euler or Runge-Kutta. Both integration methods can be implemented straightforward on both CPU and GPU [KKKW05]. This raises the question about the choice of an appropriate algorithm with regard to computational efficiency and numerical accuracy. As it was shown by Shadden et al. [SLM05], the Euler scheme does not produce sufficiently accurate results for turbulent flows, thus the application of a more precise (and more numerically involved) schemes is required. In particular, in our current implementation we employ the Runge-Kutta integration scheme of fourth-order.

Regarding the computation costs of the algorithm, it should be taken into account that the velocity vector field of an unsteady flow is defined over a discrete set of points in both spatial and temporal domains. If we use linear interpolation for both spatial (3D) and temporal domains, the simplest Euler integration scheme requires 12 lookup operations into the velocity field (6 per one time step – for all direct neighbors of the voxel along the coordinate axes X, Y, Z). On the other hand, the accurate integration methods require several vector field values at different spatial and temporal locations per one integration step. Thus, the evaluation of the flow map for 3D flows using these schemes requires $6 \times \kappa_1 \times \kappa_2$ operations, where κ_1 corresponds to the number of queries in the spatial domain and κ_2 in the temporal domain. Moreover, several authors demonstrated that a linear interpolation scheme does not always yield sufficiently accurate results and suggested to use a third-order interpolator instead [SLM05, LM05]. It is worth mentioning that for our applications the *linear* interpolation scheme demonstrated satisfactory results.

While computing the flow map $\phi_{t_0}^{t_0+\Delta t}$, there is one important issue to be addressed: the influence of the integration time Δt on the quality of the FTLE fields. A thorough discussion on this matter can be found in [LL04]. In simple words, the shape of coherent structures in the FTLE field becomes more prominent when the integration time Δt increases. The explanation for this phenomenon is very simple: the further away from the center of the structure the neighboring particles are originated, the longer time is required for them to separate. However, fluid flows are usually observed over a finite time interval Δt; thus, the velocity field is known only within this limited period of time Δt and it does not make sense to trace the particles beyond it.

Regarding the finiteness of both spatial and temporal domains, there is another closely related problem to be solved. This problem occurs when the particles are leaving the domain. In this case, no reliable estimate of the vector field is available. One solution to this problem is to limit the area, where the particles will be initiated, and to choose the integration time Δt in such a way that no particle will leave the domain within this specified period of time. In this case, there is a risk though that the flow structures in the FTLE field will be smeared or will not be visible at all. The easiest way to treat this problem is to setup sufficiently large margins, where the particles will be initiated. However, for high-speed flows this method will result in undesirably wide empty borders around the data set, where the FTLE is not defined.

At this point we should explain why the treatment of the boundary regions is the problem per se. Let us imagine the situation when one particle is leaving the domain, but its neighboring particles still stay for some period of time within the domain. Note that the particle outside of the flow domain is still possibly traveling in the close vicinity of its inside-of-domain neighbors. If we will just stop advecting the particle outside of the flow domain but still use it to construct the Cauchy-Green tensor $\mathbb{C}$, the computed amount of stretching – and thus the FTLE values $\boldsymbol{\sigma}_{t_0}^{\Delta t}(\mathbf{p})$ for this particle and its neighbors – will be artificially high.

We have founded the following practical solution to the problem associated with the boundary regions, where the particles are leaving the domain (see Figure 6.4). First, we define the safety time threshold t_{safe}, based on the time interval needed for a particle traveling along the diagonal of the volumetric flow domain with velocity $\mathbf{V}$ equal to the average velocity $\widetilde{\mathbf{V}}$ of the vector field to leave the flow domain. In the presence of obstacles within the flow domain, the voxels occupied by the obstacles should be excluded from the consideration while computing the average velocity for the flow field. Then the time interval, arbitrarily chosen by the user t_{user}, is tested against t_{safe}, and the minimum is chosen: $\Delta t = \min(t_{\text{user}}, t_{\text{safe}})$. In this way a reasonable time interval for the integration can be set up. The particles leaving the domain are simply projected onto the borders of the flow domain and the velocities from these projected positions are selected respectively.

Note that the suggested treatment of the particles leaving the flow domain does not yield the exact FTLE values close to the domain borders, yet it still produces more plausible results than filling up the wide borders around the data set with zeros.

In close relation to the question of boundary treatment, a similar problem occurs on the border of the fluid-obstacle interface. Note that this problem is many-sided and there is still an open discussion on this matter. For instance, the FTLE values $\boldsymbol{\sigma}_{t_0}^{\Delta t}(\mathbf{p}_{\text{stagnation}})$ in the vicinity of stagnation points will be extremely high. Typically, it is assumed that

```
Handle_boundary_regions(VF; $\widetilde{\mathbf{V}}$; $t_{\text{user}}$; $\mathbf{p}_{\text{out}}$; $domainDiagonal$) {
1.  Compute $\widetilde{\mathbf{V}}$ excluding obstacles;
2.  Compute $t_{\text{safe}} = domainDiagonal/\|\widetilde{\mathbf{V}}\|_2$;
3.  $\Delta t$ = $\min(t_{\text{user}}, t_{\text{safe}})$;
4.  Clamp velocities of particles outside of the flow domain
    $\mathbf{p}_{\text{out}}$ to the velocities of their projections $\text{Proj}_{\text{domain}}(\mathbf{p}_{\text{out}})$
    onto the volumetric flow domain:
       $\mathbf{V}(\mathbf{p}_{\text{out}})$ = $\mathbf{V}(\text{Proj}_{\text{domain}}[\mathbf{p}_{\text{out}}])$;
}
```

Figure 6.4: *Treatment of particles leaving the domain and the choice for integration time Δt.*

high FTLE values are associated with the "interesting" regions in the flow. In regard to these two facts, the following problem can arise. One could argue that high FTLE values are a desired effect in this case since the flow *is* deformed by the obstacle, and the presence of this obstacle essentially introduces the new flow structures within the flow domain (imagine the turbulence patterns in the clouds behind the plane). On the other side, if we start a path line in a narrow region in the vicinity of a stagnation point, the result will not give much insight about the data set beyond of this narrow region around the stagnation point, and thus this region cannot be considered as the region of interest.

In our implementation we stop advecting the particles when they land on the obstacle, while still advecting the neighboring particles that successfully passed the obstacle. We leave it up to the user to decide whether to consider the regions with extremely high FTLE values on the border of the interface fluid-obstacle as interesting or not.

Cauchy-Green Tensor Computation

Once the flow map $\phi_{t_0}^{t_0+\Delta t}$ is available, its gradient $\frac{\mathrm{d}\phi_{t_0}^{t_0+\Delta t}(\mathbf{p})}{\mathrm{d}\mathbf{p}}$ should be computed. Since the particles are initiated at the grid points, the gradient can be computed using standard differencing schemes, e.g., central differences. If the central differences approach for the 3D case is applied the computation of the flow map gradient can be written as follows:

$$\left.\frac{\mathrm{d}\phi_{t_0}^{t_1}(\mathbf{p})}{\mathrm{d}\mathbf{p}}\right|_{\substack{\mathbf{p}_{[i,j,k]},\\ t_1=t_0+\Delta t}} = \begin{bmatrix} \frac{x^{(t_1)}_{[i+1,j,k]} - x^{(t_1)}_{[i-1,j,k]}}{x^{(t_0)}_{[i+1,j,k]} - x^{(t_0)}_{[i-1,j,k]}} & \frac{x^{(t_1)}_{[i,j+1,k]} - x^{(t_1)}_{[i,j-1,k]}}{y^{(t_0)}_{[i,j+1,k]} - y^{(t_0)}_{[i,j-1,k]}} & \frac{x^{(t_1)}_{[i,j,k+1]} - x^{(t_1)}_{[i,j,k-1]}}{z^{(t_0)}_{[i,j,k+1]} - z^{(t_0)}_{[i,j,k-1]}} \\ \frac{y^{(t_1)}_{[i+1,j,k]} - y^{(t_1)}_{[i-1,j,k]}}{x^{(t_0)}_{[i+1,j,k]} - x^{(t_0)}_{[i-1,j,k]}} & \frac{y^{(t_1)}_{[i,j+1,k]} - y^{(t_1)}_{[i,j-1,k]}}{y^{(t_0)}_{[i,j+1,k]} - y^{(t_0)}_{[i,j-1,k]}} & \frac{y^{(t_1)}_{[i,j,k+1]} - y^{(t_1)}_{[i,j,k-1]}}{z^{(t_0)}_{[i,j,k+1]} - z^{(t_0)}_{[i,j,k-1]}} \\ \frac{z^{(t_1)}_{[i+1,j,k]} - z^{(t_1)}_{[i-1,j,k]}}{x^{(t_0)}_{[i+1,j,k]} - x^{(t_0)}_{[i-1,j,k]}} & \frac{z^{(t_1)}_{[i,j+1,k]} - z^{(t_1)}_{[i,j-1,k]}}{y^{(t_0)}_{[i,j+1,k]} - y^{(t_0)}_{[i,j-1,k]}} & \frac{z^{(t_1)}_{[i,j,k+1]} - z^{(t_1)}_{[i,j,k-1]}}{z^{(t_0)}_{[i,j,k+1]} - z^{(t_0)}_{[i,j,k-1]}} \end{bmatrix}. \quad (6.10)$$

The notation used in Equation 6.10 is illustrated in Figure 6.1 (b). To avoid visual clutter, the 2D version of this notation is used. The distances between the neighboring points at time t_0 are simply the lengths of the grid voxel sides. The flow map operation $\boldsymbol{\phi}_{t_0}^{t_1}$ perturbs the particle stencil, and thus changes the distances between the neighboring points. The depicted stencil deformation in Figure 6.1 (b) visually illustrates the notion of *stretching* about the trajectory.

While computing the gradient flow map, special attention should be paid to boundary regions of the flow domain and to the flow-obstacle interface. Since not all of the six neighbors of the voxel are available, it makes sense to use forward and backward differences schemes in these regions.

After the gradient of the flow map $\frac{\mathrm{d}\boldsymbol{\phi}_{t_0}^{t_0+\Delta t}(\mathbf{p})}{\mathrm{d}\mathbf{p}}$ is computed, the evaluation of the Cauchy-Green tensor $\mathbb{C}$ is performed via standard matrix transpose and multiplication operations. Eigenvalues of the symmetric tensor $\mathbb{C}$ can be found using any standard method. Since the characteristic polynomial of the corresponding matrix is cubic in the 3D case, we exploit Cardano's solver to find its roots [PTVF07] and thus eigenvalues of the tensor $\mathbb{C}$. After sorting, $\boldsymbol{\lambda}_{\mathbf{max}}(\mathbb{C})$ is used to compute the FTLE value $\boldsymbol{\sigma}_{t_0}^{\Delta t}(\mathbf{p})$ according to Equation 6.9.

6.2 Anchor Lines

As can be seen from the images presented in Chapter 5, the importance-driven techniques proposed so far in the literature still output an excessive amount of information in both focus and context regions. In order to reduce the amount of data depicted at once, while emphasizing the interesting flow structures, we introduce *anchor lines*, a new technique for the extraction of locally important information encoded within the flow velocity field.

The anchor line technique is inspired by ideas presented in [LG98], where short stream lines called *streamlets* were placed in the vicinity of characteristic trajectories to show the local flow behavior along these trajectories. The proposed anchor lines are employed for local analysis of the vector fields based on the FTLE measure. In particular, they visualize the degree of separation between the neighboring particles over time and the rate of this separation. In this section we describe in detail the construction of anchor lines, their placement strategies, as well as rendering aspects of anchor lines and their accompanying particles.

6.2.1 Computation

In general, the FTLE is an averaged measure of an infinitesimally small deformation in the vicinity of a point in the phase space of a dynamical system. Applied to fluid flow fields, the FTLE quantifies the amount of stretching of a fluid element over a fixed period of time (see Section 6.1). Intuitively, the larger the FTLE value at a given point, the more stretching about the trajectory seeded at this point occurs. Furthermore, the FTLE allows to locate transport barriers and it has been studied for the analysis of transport and mixing characteristics in multi-dimensional flows [Hal01, LSM06].

Seeding

Conceptually, the anchor lines correspond to the path lines in the vector field. The starting positions of these lines can be *arbitrarily selected by the user* within the flow domain. Similarly to streamlets [LG98], the local information along the anchor line $\boldsymbol{\mathcal{A}}$ is visualized by additional particles (*accompanying particles*) $\mathbb{P}_{\boldsymbol{\mathcal{A}}}$ seeded in the vicinity of the starting position of this line. The amount of particle scattering Δr around the seeding positions of the corresponding anchor lines is selected by the user.

In addition to the user-controlled placement of anchor lines, we propose to select the starting points of these lines automatically, *based on the FTLE value*. In particular, we initiate the points randomly in the interior of a user-defined probe, and then reject those ones, which have the associated FTLE value $\boldsymbol{\sigma}_{t_0}^{\Delta t}(\mathbf{p})$ below a certain threshold $\sigma_{\max}$. It is worth noting that the FTLE is precomputed at every point of the given sampling grid.

The reason for restricting the placement of anchor lines to regions of high FTLE is as follows. While the FTLE characterizes the rate of separation of particles, it does neither indicate into which direction particles separate nor does it tell where the particles separate along a trajectory. Anchor lines placed in regions of high FTLE, on the other hand, are able to answer both questions and can thus be used for an improved analysis of the flow. Figure 6.5 demonstrates this property (the description of the visualized data set is given in Section 6.3.1).

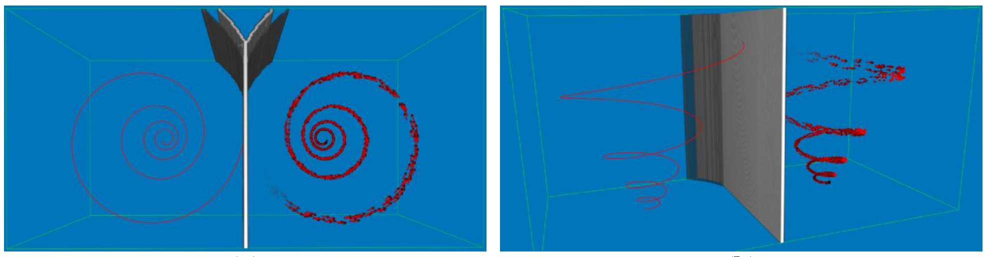

(a) (b)

Figure 6.5: *Anchor lines placed in regions of high finite-time Lyapunov exponent (depicted in grey) can effectively emphasize the dual vortex structure of the flow.*

Tracing

For efficiency reasons, anchor lines are always traced in parallel with their accompanying particles. In every integration step the Euclidean distance between a particle $\mathbb{P}_{\mathcal{A}}$ and the current point on the respective anchor line $\mathcal{A}$ is used as a measure for the deviation. Since in complex flow scenarios an anchor line and a trajectory of an accompanying particle can deviate from each other and again approach each other, it makes sense to consider the *maximum deviation* ($maxDev$) of a particle along its path. In some cases it can be also useful to compare the lengths of the particle trajectory and the corresponding anchor line[2]. The process of computation of both measures for the particle deviation is illustrated in Figure 6.6.

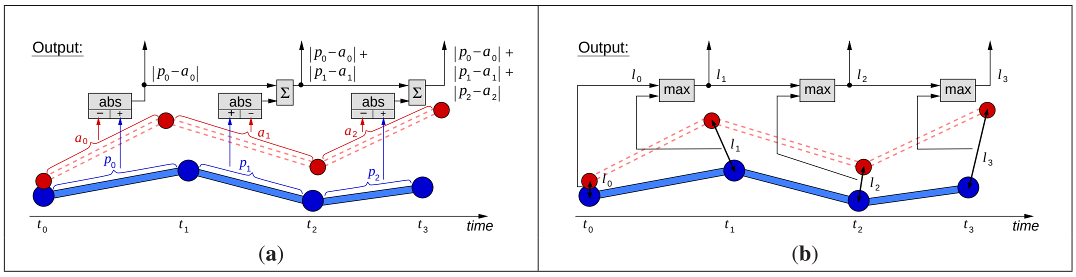


Figure 6.6: *Measures for the particle deviation:* (**a**) *based on the accumulated lengths of the particle trajectory* $\mathcal{P}_{\mathcal{A}}$ *and the corresponding anchor line* $\mathcal{A}$, (**b**) *maximum deviation* $maxDev$.

6.2.2 Details on the GPU Implementation

The attributes of particles and lines are stored in texture objects on the GPU. A set of attributes associated with each particle consists of the following quantities:

age	–	the age of the particle needed for particle tracing [KKKW05];
$maxDev$	–	the maximum deviation of the particle from its anchor line;
(x, y, z)	–	the particle's position $\mathbf{p}$;
(x_a, y_a, z_a)	–	the corresponding position $\mathbf{a}$ on the particle's anchor line.

The maximum number of components per pixel available in a GPU texture object is four. Thus all the attributes listed above can be stored in two 2D textures of the same size $m_p \times n_p$. For brevity, we will use the following notations for these textures:

PPosTex: (1 texel)

x	y	z	age
R	G	B	A

ALPosTex: (1 texel)

x_a	y_a	z_a	$maxDev$
R	G	B	A

[2] More precisely, the lengths of a *piecewise approximation* of these lines can be compared.

Thus, `PPosTex` is the texture storing the particle positions ($\mathbf{p}$) and their age (age), and `ALPosTex` is the texture containing the anchor lines positions ($\mathbf{a}$) and the maximum deviation ($maxDev$). As it was described in [KKKW05], particle tracing requires a pair of textures of the same size for particle positions, which are alternatively updated in every advection step (ping-pong technique). Thus, in total, four textures are needed to perform the parallel tracing of anchor lines and particles.

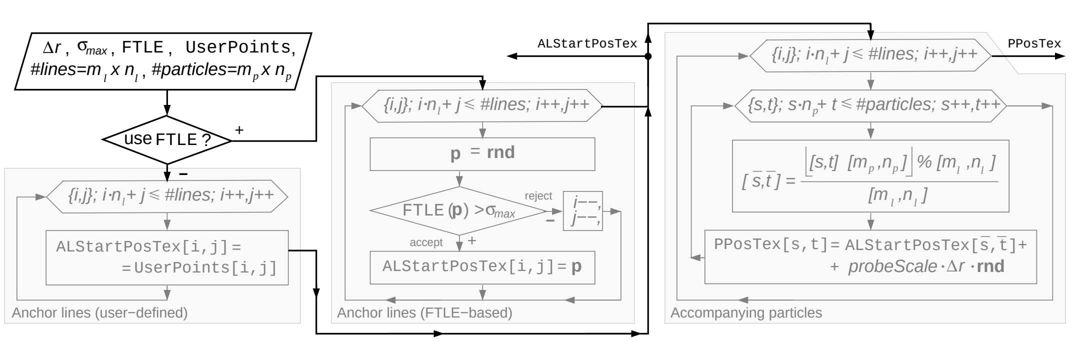
The multiple render target (MRT) technique allows to trace the anchor lines and the particle trajectories in parallel. In general, the number of anchor lines is supposed to be much smaller than the number of accompanying particles. However, to exploit the MRT technique, the render targets should be of the same size and format [DX9]. Thus, we have to "smear" the anchor line positions into the texture of the same size as the particle position texture, i.e., $m_p \times n_p$. In order to prove that this redundancy does not induce a significant drop of performance, the following observations should be taken into account:

- If the textures of the same size are used, no manipulation of texture coordinates should be performed to access the anchor line position in `ALPosTex` corresponding to a given particle position in `PPosTex`. In fact, the *same* texture coordinates are used for the texture fetch in this case.
- The costly texture lookup operation into `ALPosTex` is required for each particle position in any case, no matter which size of the texture is used to store the anchor line positions.
- Storing the anchor line positions in a texture of the same size as the number of lines, i.e., $m_l \times n_l$, and thus giving up the possibility to exploit the MRT technique, leads to the necessity to compute the particle trajectories and anchor lines in two passes, which is less efficient than using the MRTs.

Therefore, the usage of the "oversized" texture to store the anchor line positions does not lead to a drop of performance and we accept it in our implementation. On the other hand, for convenience reasons, the starting positions of the anchor lines are stored in the texture `ALStartPosTex` of size $m_l \times n_l$ corresponding to the actual number of anchor lines. A block diagram of the seeding procedure for anchor lines and accompanying particles using the texture notation is shown in Figure 6.7.

Technically, the maximum deviation $maxDev$ of particles from their anchor lines is computed as follows. We compute in each advection step t for each particle $\mathbb{P}_{\mathcal{A}}$ its next position $\texttt{PPosTex}[\texttt{i},\texttt{j}](t)$[3], the next position $\texttt{ALPosTex}[\texttt{i},\texttt{j}](t)$ of its anchor line

[3] In texture coordinates, spanning the range from 0.0 to 1.0, integer indices $[i, j]$ correspond to $\texttt{tc}[x, y]$.



Δr, σmax, FTLE, UserPoints, #lines=ml x nl, #particles=mp x np
use FTLE ?
+
–
{i,j}; i·nl+ j ≤ #lines; i++,j++
ALStartPosTex[i,j] = =UserPoints[i,j]
Anchor lines (user–defined)
{i,j}; i·nl+ j ≤ #lines; i++,j++
p = rnd
FTLE (p) >σmax
reject
accept
j––, j––,
ALStartPosTex[i,j] = p
Anchor lines (FTLE–based)
ALStartPosTex
PPosTex
{i,j}; i·nl+ j ≤ #lines; i++,j++
{s,t}; s·np+ t ≤ #particles; s++,t++
[s̄,t̄] = ⌊[s,t] [mp,np]⌋ % [ml,nl] / [ml,nl]
PPosTex[s,t]= ALStartPosTex[s̄,t̄]+ + probeScale·Δr ·rnd
Accompanying particles

Figure 6.7: *Block diagram of the seeding procedure for anchor lines and accompanying particles. The following parameters are specified by the user: Δr – the amount of particle scattering, $\sigma_{\max}$ – the threshold for the FTLE values,* `UserPoints` *– a set of points, and $probeScale$ – the scaling factor of the user probe-cell placed inside of the flow domain.*

$\mathcal{A}$, and the distance between these positions $distance(t)$. The distance $distance(t)$ is compared to this distance computed in the previous advection step $distance(t-1)$ and the maximum value is stored in `ALPosTex`(t)`.w`. The pseudo-code of the parallel tracing of the particle trajectories $\mathcal{P}_{\mathcal{A}}$ and the corresponding anchor line $\mathcal{A}$ including the computation of $maxDev$ for a single particle $\mathbb{P}_{\mathcal{A}}$ at time t is listed in Figure 6.8.

```
AdvectionStep ($t, \Delta t$, tc$[x,y]$, $m_l \times n_l$; VF($t$), ALStartPosTex, PPosTex($t-1$), ALPosTex($t-1$);
               $\Delta r$, $probescale$, PStartPosTex) {
1.  if(out of grid or too old) {
2.    tc$_{\text{anchor}}[x,y]$ = $\left(\left\lfloor \text{tc}[x,y] \cdot [m_p, n_p] \right\rfloor \% [m_l, n_l]\right) / [m_l, n_l]$;   // Texture coordinates for $\mathcal{A}$
3.    ALPosTex($t$).xyz = ALStartPosTex[tc$_{\text{anchor}}$].xyz;                // Init $\mathcal{A}$
4.    ALPosTex($t$).w   = 0;                                                       // Init maximum deviation
5.    randomShift       = ($probeScale \cdot \Delta r$) $\cdot$ PStartPosTex.xyz;  // Scatter positions
6.    PPosTex($t$).xyz  = ALPosTex($t$).xyz + randomShift;                         // Init position of $\mathbb{P}_{\mathcal{A}}$
7.    PPosTex($t$).w    = PStartPosTex.w;                                          // Init age of $\mathbb{P}_{\mathcal{A}}$
8.  } else {
9.    PPosTex($t$).xyz  = PPosTex($t-1$).xyz + VF(PPosTex($t-1$), $t$) $\cdot \Delta t$;     // Advect $\mathbb{P}_{\mathcal{A}}$
10.   PPosTex($t$).w    = PPosTex($t-1$).w − 1;                                              // Change age of $\mathbb{P}_{\mathcal{A}}$
11.   ALPosTex($t$).xyz = ALPosTex($t-1$).xyz + VF(ALPosTex($t-1$), $t$) $\cdot \Delta t$;   // Advect $\mathcal{A}$
12.   $distance$        = $|$PPosTex($t$).xyz − ALPosTex($t$).xyz$|$;
13.   ALPosTex($t$).w   = max (ALPosTex($t-1$).w, $distance$);                               // Maximum deviation $maxDev$
14. }
```

Figure 6.8: *Parallel tracing of the particle trajectory $\mathcal{P}_{\mathcal{A}}$ and the respective anchor line $\mathcal{A}$: per-pixel operations when the Euler integration scheme is exploited. Output: textures* `ALPosTex`*(t) and* `PPosTex`*(t).*

In Figure 6.8, line 5, the texture `PStartPosTex` contains the random starting positions of the particles within the unit cube and their randomized maximum allowed ages. The variable $probeScale$ denotes the scaling factor of the user probe-cell placed inside of the flow domain. The variable **randomShift** (line 5) stores a shift from a given position $\texttt{ALPosTex}[\texttt{i},\texttt{j}](t).\texttt{xyz}$ in a random direction $\texttt{PStartPosTex.xyz}$ by a random distance within a sphere of radius Δr. In texture read-write operations, the texture coordinate indices $\texttt{tc}[x, y]$ are omitted for brevity. Therefore, each operation of type $\texttt{texture}(t)$ should be interpreted as $\texttt{texture}[\texttt{tc.x},\texttt{tc.y}](t)$. The whole process of initialization and tracing of the accompanying particles and the respective anchor lines including their initialization is visualized in Figure 6.9.

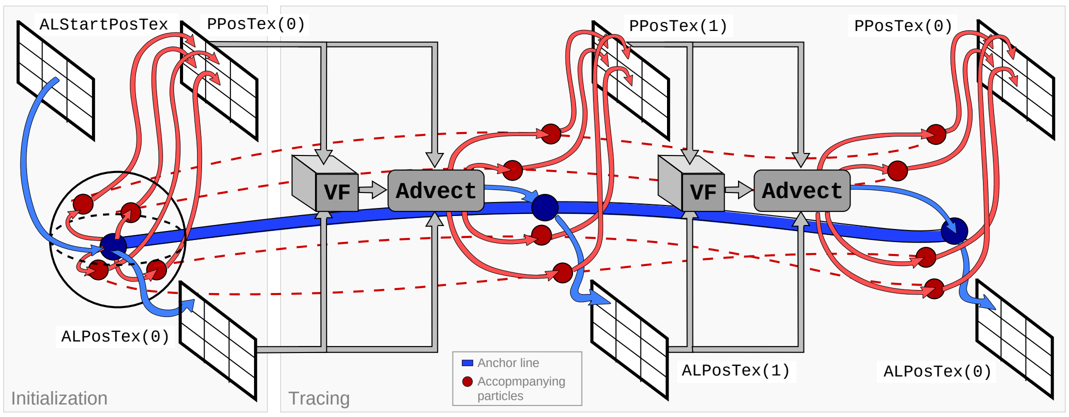


Figure 6.9: *Initialization and tracing of anchor lines and accompanying particles.*

If all the particles are launching at the same time in a close proximity to each other, it gives the impression that the particles are moving in packets of fixed size. In order to avoid this, the maximum allowed age of the particles is randomized [KKKW05]. If a particle's age exceeds this limit or if the particle moves outside of the specified domain, then this particle is reincarnated in the vicinity of the starting position of the respective anchor line (cf. Figure 6.8, line 6).

6.2.3 Rendering

Usually, a few anchor lines are already sufficient in order to visualize the structures of interest in the flow. We found out, that assigning distinctive colors to these lines allows to visualize several flow structures at once in a most intuitive way. Logically, the accompanying particles $\mathbb{P}_{\mathcal{A}}$ inherit the color of the respective anchor line $\mathcal{A}$.

Transparency modulation is a common technique used for highlighting and fading out the features in the visualized scenes. In this regard, we provide the user with a variety of modes for anchor lines rendering. In the *simple mode*, the anchor lines themselves are rendered using a standard stream balls technique for flow field visualization with enabled lighting [KKKW05]. Since the stream balls are represented by a 3D geometry object, its shape is adequately perceived via correct lighting computation. The accompanying particles are rendered as opaque 3D oriented glyphs. An example of anchor lines rendered using the simple mode is shown in Figure 6.5.

In order to emphasize the flow features using the transparency modulation (*focus-and-context mode*), the focus and context techniques available in the extended version of the GPU particle engine (see Section 5.3) can be enabled. Since the anchor lines represent a part of the topological skeleton of the flow field, they can be highlighted by being rendered absolutely opaque. In this case, the transparency of the accompanying particles can be modulated according to a particular transfer function (see Figure 6.12).

In addition to focus and context feature blending, the accompanying particles $\mathbb{P}_{\mathcal{A}}$ can be modulated based on their maximum deviation $maxDev$ from the corresponding anchor line $\mathcal{A}$ (*anchor-based mode*). Thus, particles close to the line can be faded out while they are rendered more and more opaque once they start to separate from their anchor. Since the $maxDev$ is used as a criterion for fading out, once the particle deviates more than a specified threshold from the corresponding anchor line, it is rendered opaque along the remaining path no matter whether it again approaches the respective anchor line or not.

Intuitively, in the regions characterized by a high similarity between the neighboring vector field values, the particles are remaining close to the respective anchor lines. Thus, assigning high transparency to these particles and showing only the corresponding anchor line (*anchor-based mode*), allows for automatic reduction of excessive information in such regions. On the other hand, in highly heterogeneous regions where the separation rate is considerably higher, the accompanying particles are emphasized (see Figure 6.10). From the transparency of particle $\mathbb{P}_{\mathcal{A}}$ it can be directly derived how much this particle separates from the respective anchor line $\mathcal{A}$ over time. The time required for a particle to deviate by a specified distance from the anchor is not directly encoded as a visual attribute in the current implementation, but it can be determined from the animation of particles over time. In general, it could be also encoded as an additional attribute like color or size.

6.3 GPU Particle Engine with Anchor Lines

The proposed anchor line technique for flow visualization has been integrated into the GPU particle engine described in Sections 5.2-5.3. In this section the efficiency of the particle engine equipped with anchor lines is validated using several simple flow

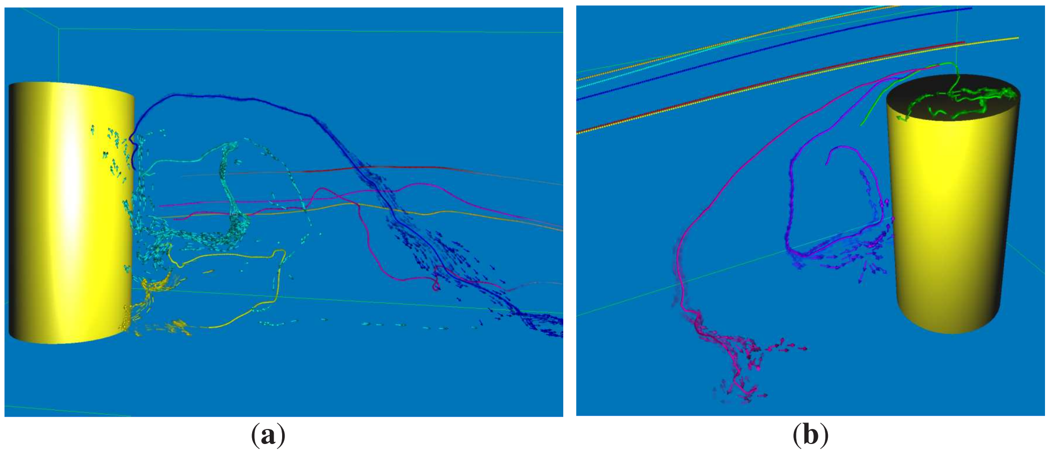


Figure 6.10: *Anchor lines (path lines) and accompanying particles seeded close to their starting points. While particles exactly follow some of the lines ((**a**): red, magenta, gold; (**b**): blue, red, yellow), along other lines the particles diverge from their anchor lines at different speed ((**a**): blue, light blue, yellow; (**b**): green, magenta, purple). To improve the visual perception of the correspondence between anchor lines and particles, for every anchor line $\boldsymbol{\mathcal{A}}$ is assigned a unique color that is inherited by the accompanying particles $\mathbb{P}_{\boldsymbol{\mathcal{A}}}$.* Anchor-based mode*: particle transparency is inversely proportional to its separation distance $maxDev$ from the anchor line.*

scenarios and time-varying sequences of complex fluid flows. Moreover, the overall performance of the extended GPU particle engine is discussed.

6.3.1 Description of the Data Sets Used for Validation

In order to demonstrate the efficiency of the particle engine with anchor lines, the following data sets of 3D flows given on Cartesian grids have been used:

- *Double-vortex flow*: A steady axisymmetric flow with two counter rotating vortices, which was computed using the following analytical expression for velocity $\mathbf{V} = (V_x, V_y, V_z)$:

$$\begin{aligned} V_x &= (-y + 0.5) + (0.5 - 2.0 \cdot x)/10.0 \\ V_y &= (2.0 \cdot x - 0.5) + (0.5 - y)/10.0 \\ V_z &= -z/10.0. \end{aligned}$$

  The computed velocity field corresponds to a spiral-like flow around the Z-axis with the velocity magnitude decreasing towards the main axis of the spiral. The velocity field was mirrored to obtain the two symmetric vortices. An example of visualization of this data set is presented in Figure 6.5.

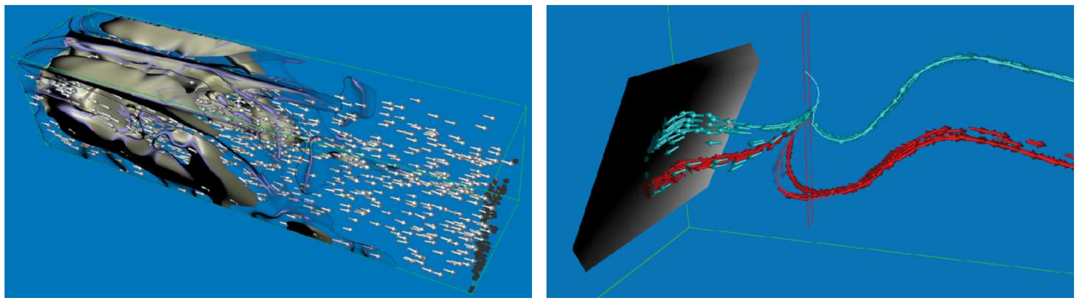
- *Flow around a box*: Result of a 3D time-dependent simulation of an incompressible turbulent flow around a square cylinder at $Re = 22.000$. The simulation was performed using a spectro-consistent discretization of the Navier-Stokes equations [VV98]. The simulation was carried out on a rectilinear grid of size 256×448×64.
- *Flow around a cylinder*: Large eddy simulation of an incompressible unsteady turbulent flow around a wall-mounted finite cylinder at $Re = 200.000$ [FWT05]. 22 time steps were simulated. The size of the data grid is 256×128×128. This data set is visualized in Figure 6.10.
- *Kármán vortex street*: Result of a 3D simulation of an incompressible unsteady flow over an immersed thin rectangular obstacle at $Re = 100$. The simulation was performed via numerical solution of the Navier-Stokes equations according to [GDN98]. The data set contains 30 time steps, each of which is represented by a volume of size 256×64×64.

6.3.2 Results of Visualization and Performance

To validate the effectiveness of the GPU particle engine with anchor lines, we demonstrate additional images of the described data sets rendered using different importance-based visualization techniques in Figures 6.11 and 6.12. With respect to the generated static portrayal, it is worth mentioning that the benefits of particle-based flow visualization can best be perceived in an animation. In a still image, oriented particles can clearly convey the direction of the flow, but in contrast to LIC, for example, coherent particle trajectories can hardly be observed.

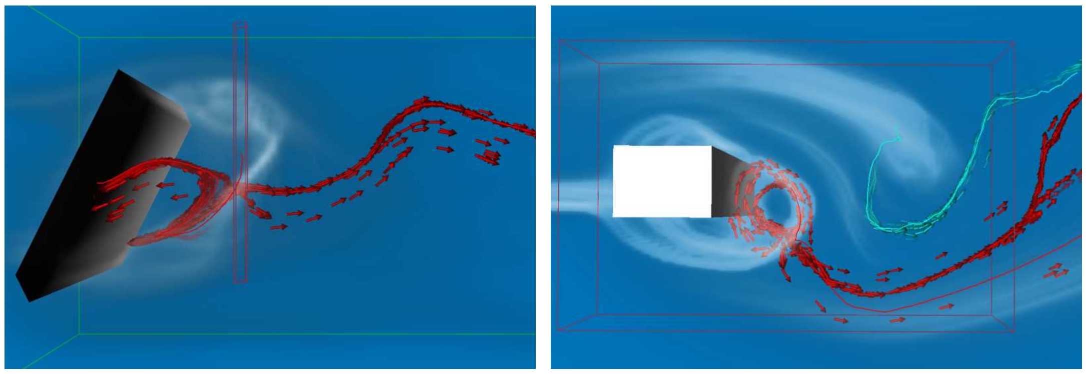
All of our rendering tests were run on a dual core Core2 Duo 6600 equipped with an NVIDIA Geforce 8800 GTX graphics card with 786 MB local video memory. In terms of performance, it can be observed that on recent GPUs the particle advection step consumes only a negligible fraction of the overall time. For instance, in a steady field about 100 millions of particles can be integrated per second using an embedded Runge-Kutta scheme of fourth-order on our target architecture. Although this rate drops significantly in the unsteady case (20%–40%), where streaming the time steps consumes most of the time, we can still trace about 20 millions of particles per second in a time-varying flow field of size 256×256×256. For more detailed timings the reader is referred to [KKKW05] and [BSK$^+$07].

The overall performance of the GPU particle engine strongly depends on the number and the size of the rendered particle primitives. In particular, the more large particles are



(**a**) (**b**)

Figure 6.11: *The visualization of the Kármán vortex street: (**a**) an isosurface of the FTLE, (**b**) two anchor lines placed in the region of high FTLE. From the particle distribution in image (**b**) one can see where particles start to separate from their anchor, and the transparency coding shows how fast they separate.*



(**a**) (**b**)

Figure 6.12: *Anchor lines seeded in the region of high FTLE: (**a**) in the Kármán vortex street, (**b**) in the flow around a box. In both images, the distribution of the FTLE field is visualized using volume rendering (*focus-and-context mode*).*

rendered the faster the application becomes raster bound and the overall performance can decrease considerably. On the other hand, since the importance-driven approaches including the proposed anchor lines technique can effectively reduce the amount of rendered particles, in none of our rendering experiments the performance dropped below 100 fps.

Chapter 7

The Application of Particle Tracing to Tensor Field Visualization

Visual exploration of tensor fields is highly important in many areas of natural sciences. Tensors belong to a special category of multivariate data, incorporating valuable physical information about the underlying phenomenon. The classical examples of quantities represented by tensors include viscous stresses, rate of strain, and velocity gradient describing the flow properties in fluid mechanics; stress and strain expressing the response of material to applied forces in solid mechanics and tectonics; diffusion tensors characterizing Brownian motion of water molecules within the tissue in medicine, etc.

Mathematically, a tensor is described by a number of correlated scalar functions defined over a multi-dimensional domain. For example, the fundamental tensor fields encountered in engineering and the physical sciences consist of second-order tensors, which in the 3D case have nine components. Since it is meaningless to display all these interconnected components independently, the visualization of tensor data is a challenging task. Moreover, the physical interpretation of mathematical tensor features is highly important and it strongly depends on a particular application. This implies that the methods, suitable for the tensor fields arising in one area of natural sciences are not always applicable to the tensor fields in another area. Furthermore, the study of interdependencies between the mathematical and physical properties of the tensors is often complicated by the lack of measurement techniques for experimental investigation of tensor fields in some applications.

A number of approaches for tensor field visualization have been proposed in the last few years. However, due to the intrinsic complexity of the data to be visualized, all of these approaches have disadvantages and need further elaboration. For instance, many visualization methods rely upon the mathematical properties of the tensors and do not give any physical interpretation of the results of visualization. Other approaches incorporate physics but are implemented only for simple 2D cases. In general, most of the presented approaches are showing either local features of the tensor fields, or their global structure, and thus do not allow for a complete understanding of the underlying tensor data. Therefore, the development of more efficient methods for visual exploration of tensor fields is of vital interest for the scientific community.

In this chapter we introduce GPU particle tracing for the visualization of 3D diffusion tensor fields. The developed method provides efficient and intuitive means to show the dynamics in diffusion tensor fields, and it can thus be applied for the exploration of the diffusion properties of biological tissue. For about half a million particles, reconstruction of diffusion directions from the tensor field, time integration and rendering are performed at interactive rates. Different visualization options like oriented particles of diffusion-dependent shape, stream lines or stream tubes facilitate the use of particle tracing for diffusion tensor visualization. The efficiency of the proposed method is demonstrated on real-world data sets of biological tissue.

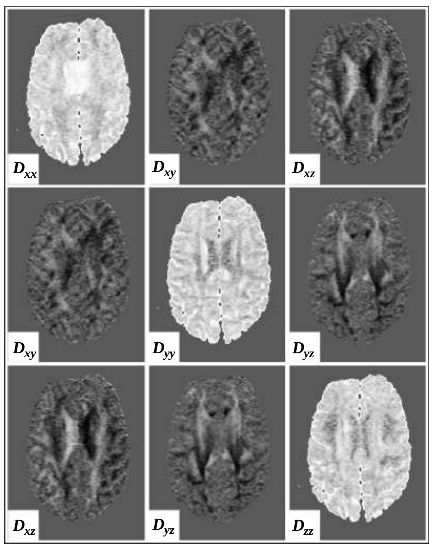
7.1 Visualization of Diffusion Tensor Fields

The diffusion properties of biological tissue can be measured using *diffusion tensor magnetic resonance imaging*[1] (DT-MRI) [BML94]. The imaging process reveals the diffusion of water molecules depending on the shape and orientation of tissue cells, i.e., the diffusion is anisotropic within fibrous material while there is an equal diffusion probability in real matter of other types. The *diffusion probability* is characterized by a second-order tensor, which describes the orientation and mutual alignment of *molecular pathways* as a function of spatial position. Since molecular pathways reveal the connectivity between the biological regions, visualization of diffusion tensor fields helps to investigate the local and global structure of biological tissue.

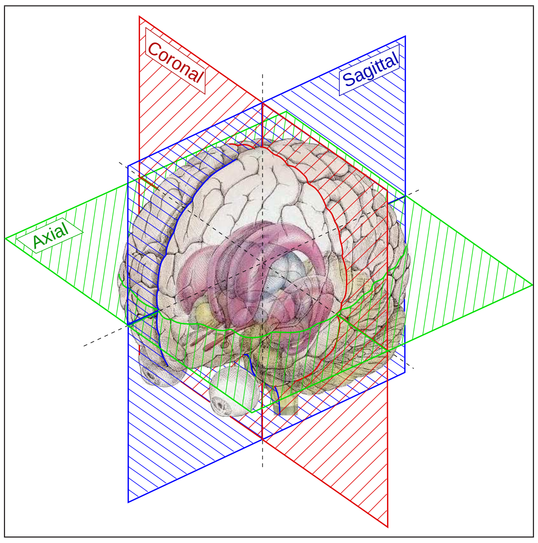
In order to keep the text in this chapter concise and clear we use the standard terminology specific for anatomical description of biological tissue and, in particular, human brain. These terms are compiled in Figures 7.1 and 7.2. Figure 7.1 (a) shows one slice through a DT-MRI scan visualized as a matrix of images, each of which displays a single component of the matrix representation of the diffusion tensor. Figure Figure 7.1 (b) illustrates the notion of three reference planes parallel to the principal axes of the body: *sagittal*, *coronal*, and *axial*. The large scale *neural pathways* within the human brain

[1] A short form of this term is *diffusion tensor imaging* (DTI).

are depicted in Figure 7.2. For more detailed information on neuroanatomy the reader is referred to [AB98, KW98, Hen00].



(**a**)



(**b**)

Figure 7.1: *Diffusion tensor imaging of the human brain: (**a**) DT-MRI scan as a matrix of images by analogy with diffusion tensor matrix itself; (**b**) reference planes through the brain parallel to the principal axes of the body.*

7.1.1 Diffusion Tensors and Their Properties

Diffusion is the transport of one material through another by the action of random molecular motion due to thermal energy (*Brownian motion*). The fundamental equations describing the diffusion process are referred to as *Fick's laws*. According to the first Fick's law, the movement of a material by a diffusion (called *flux* J) is proportional to the gradient in the concentration $\mathcal{C}$ of the material [Cra75]. In three dimensions, Fick's laws generalized to a first-order model (*linear transform*) describing the materials with an arbitrary internal structure can be written as follows:

$$\begin{aligned} \mathbf{J} &= -\mathbb{D}\nabla\mathcal{C}, && 1^{\text{st}} \text{ Fick's law;} \\ \frac{\partial\mathcal{C}}{\partial t} &= \nabla\bullet\left(\mathbb{D}\nabla\mathcal{C}\right), && 2^{\text{nd}} \text{ Fick's law,} \end{aligned} \tag{7.1}$$

where the vector $\mathbf{J}$ is the direction of the overall motion, or *net flux*, of the material with concentration gradient $\nabla\mathcal{C}$. Materials are considered to be *isotropic* if the direction of the net flux is parallel to the concentration gradient. In contrast, there are *anisotropic* materials, which have a directional microstructure enforcing the diffusion in some directions to be faster than in others.

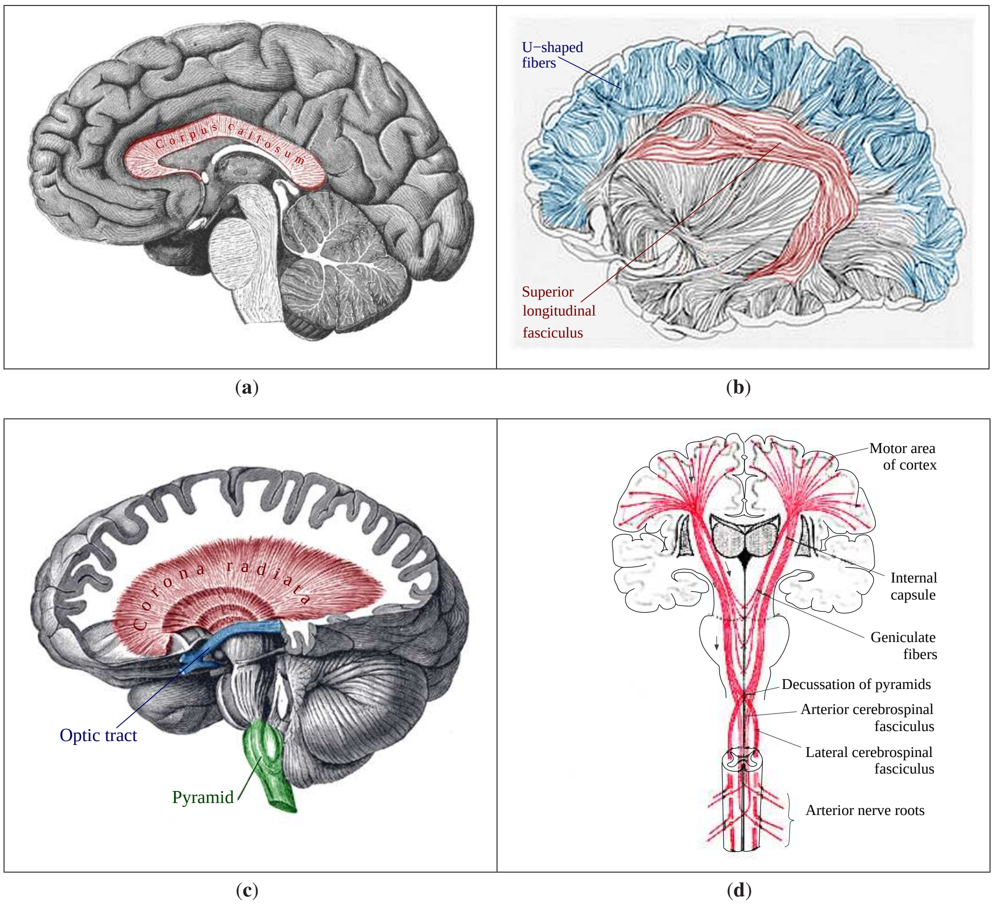


Figure 7.2: *Neural pathways in the human brain: (**a**) corpus callosum, (**b**) superior longitudinal fasciculus and U-shaped fibers, (**c**) corona radiata, optic tract, and pyramid, (**d**) pyramidal tract.*

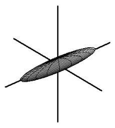
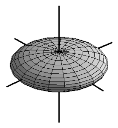
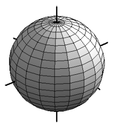
In Equation 7.1 the diffusion coefficient $\mathbb{D}$ is a second-order *diffusion tensor*, which can be expressed mathematically as a 3×3 symmetric semi-positive matrix:

$$\mathbb{D} = \begin{pmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{pmatrix}. \tag{7.2}$$

From a physical point of view, the tensor $\mathbb{D}$ describes the probability density of where a particle's Brownian motion will move it over time. Thus, following the common classification, diffusion tensors can be represented as ellipsoids with main, medium, and minor axes corresponding to the eigenvectors $\mathbf{e_1}$, $\mathbf{e_2}$, $\mathbf{e_3}$ of the tensor – with respective eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The relative differences between the eigenvalues are related to the *anisotropy* of the diffusion. Three basic types of anisotropy are usually considered in the literature (see Table 7.1).

Besides the mapping of local anisotropy to the shape of geometrical icons, a number

Table 7.1: *Classification of diffusion tensor anisotropy.*

Name	linear/prolate	planar/oblate	spherical
Coefficient	$c_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3}$	$c_p = \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3}$	$c_s = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$
Visualization			

of different color encoding schemes have been proposed:

	R	G	B	
Scheme 1:	μ_1	μ_2	A_3	(7.3)
Scheme 2:	J_4	FA	A_3	(7.4)
Scheme 3:	c_l	c_p	c_s	(7.5)
Scheme 4:	$FA\cdot \lvert e_{1_x}\rvert$	$FA\cdot \lvert e_{1_y}\rvert$	$FA\cdot \lvert e_{1_z}\rvert$	(7.6)
Scheme 5:	$RA\cdot \lvert e_{1_x}\rvert$	$RA\cdot \lvert e_{1_y}\rvert$	$RA\cdot \lvert e_{1_z}\rvert$	(7.7)

Here, the coefficients μ_1, μ_2, A_3, J_4, FA, RA are computed as follows:

$$\begin{aligned}
\mu_1 &= \tfrac{1}{3}\sum \lambda_i, && 1^{st} \text{ central moment of eigenvalues,}\\
\mu_2 &= \tfrac{1}{3}\sum(\lambda_i - \mu_1)^2, && 2^{nd} \text{ central moment of eigenvalues,}\\
A_3 &= \tfrac{\sum(\lambda_i - \mu_1)^3}{3\cdot\mu_2\sqrt{\mu_2}}, && \text{skewness of eigenvalues,}\\
J_4 &= \sum \lambda_i^2, && \text{invariant of tensor } \mathbb{D},\\
FA &= \tfrac{3}{\sqrt{2}}\sqrt{\tfrac{\mu_2}{J_4}}, && \text{fractional anisotropy,}\\
RA &= \tfrac{\sqrt{\mu_2}}{\sqrt{2}\mu_1}, && \text{relative anisotropy}
\end{aligned}$$

A thorough discussion of diffusion tensors and quantities derived from them can be found, for example, in [Kin03, MAA$^+$03].

7.1.2 Previous Approaches

The existing approaches to visualize the diffusion in real tissue can be classified into two major categories: glyph-based techniques and fiber tracking[2]. *Glyph-based methods* reveal local variations in diffusion tensor fields by mapping tensor properties like

[2] Fiber tracking is also referred to as *tensor lines* and *directional tracking*.

orientation or anisotropy to the shape or appearance of graphical primitives, i.e., ellipsoids [LAK+98], composite shapes [WMM+02], or superquadrics [Kin03]. In contrast, *fiber tracking* of massless particles along the most probable diffusion directions in tensor field data [DH92] allows for the classification of anatomical structures, e.g., the *white matter fiber tracks*. Different geometric representations like stream lines [WKL99, MCCvZ99] and stream tubes [ZCML00, DZDL01], or stream surfaces [ZCML00, VBvP04] have been employed to visualize these structures. To improve the quality and stability of such tracking techniques, regularization and filtering approaches [CAA01, ZB02, GFP+02] along with heuristics to determine the most probable directions [WKL99, ZB02] have been proposed. Dedicated color and opacity mapping schemes to visually emphasize particular features in diffusion tensor data have been presented in [WKL99, KW99, WMM+02, Kin03]. Examples of visualization of diffusion tensor fields using the existing approaches are shown in Figure 7.3

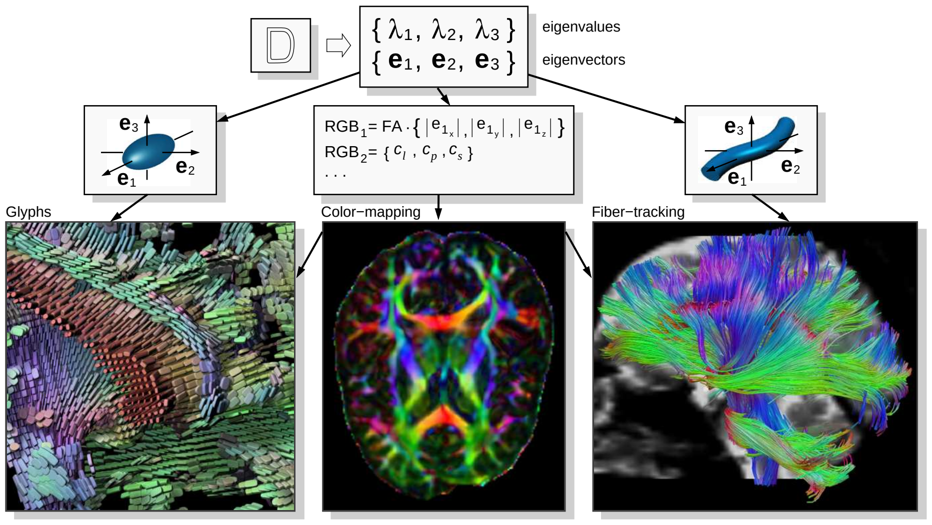

D
$\{\lambda_1, \lambda_2, \lambda_3\}$
$\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$
eigenvalues
eigenvectors
$\mathbf{e}_3$
$\mathbf{e}_2$
$\mathbf{e}_1$
$RGB_1 = FA \cdot \{|e_{1_x}|, |e_{1_y}|, |e_{1_z}|\}$
$RGB_2 = \{c_l, c_p, c_s\}$
...
$\mathbf{e}_3$
$\mathbf{e}_2$
$\mathbf{e}_1$
Glyphs
Color−mapping
Fiber−tracking

Figure 7.3: *Existing approaches for the visualization of diffusion tensor fields.*

While tracking-based techniques can effectively visualize global behavior of tensor fields as well as reveal the connectivity information between the distinctive spatial regions in tissue, glyph-based imaging techniques for visualizing 3D fields can successfully illustrate the local features in such fields. However, when using such methods it is difficult to effectively control glyph density, shape and appearance in a way that depicts both the vectorial information of the diffusion, such as principal directions, *and* the scalar properties of the diffusion, such as anisotropy and magnitude. Neither tracking-based nor glyph-based techniques allow for interactive exploration of large tensor fields, and they typically fail to visualize and analyze the diffusion dynamics in real tissue.

In this chapter, we propose an interactive technique based on the GPU particle tracing for diffusion tensor field visualization. This method can display the dynamics of large particle sets in flow fields (see Section 5.2), and it can thus be applied to investigate the diffusion in biological tissue in real time. A number of visualization options like oriented texture splats, stream lines and stream tubes provide the user with an effective means for the visual analysis of 3D diffusion tensor fields given on a Cartesian grid.

7.2 Diffusion Tensor Field Visualization Using Particle Tracing

To interactively explore the dynamics in 3D diffusion tensor fields, we employ a hardware-accelerated particle system for visualizing steady 3D flow fields on Cartesian grids (GPU particle engine) described in Section 5.2 and proposed by Krüger et al. [KKKW05]. This approach allows for interactive streaming and rendering of millions of particles, and it enables virtual exploration of high resolution fields. In application to 3D diffusion tensor fields, the ability to display the dynamics of large particle sets using visualization options like oriented texture splats, stream lines, and stream tubes provides an intuitive means for the visual analysis of these fields that is far beyond existing solutions.

7.2.1 Algorithm

To employ the GPU particle engine for tensor field visualization, a number of data-specific extensions have been integrated. In particular, the direction vector along which a particle is traced first has to be reconstructed from the tensor field. The basic differences between particle tracing in flow fields and in tensor fields are illustrated in Figure 7.4.

The specific extensions we have integrated to accommodate the use of particle tracing for tensor field visualization are briefly summarized as follows:

- The six distinct entries of the diffusion tensor $D_{xx}, D_{xy}, D_{xz}, D_{yy}, D_{yz}, D_{zz}$ are stored in two 3D RGB texture maps. Since hardware-accelerated tri-linear interpolation of 32 bit floating point textures is not supported on GPUs, we use 16 bit floating point textures in the current implementation. If this precision is not sufficient, hand–coded interpolation of 32 bit floating point values can be performed in a fragment shader. As shown in [KKKW05], this only results in a slight decrease in performance.

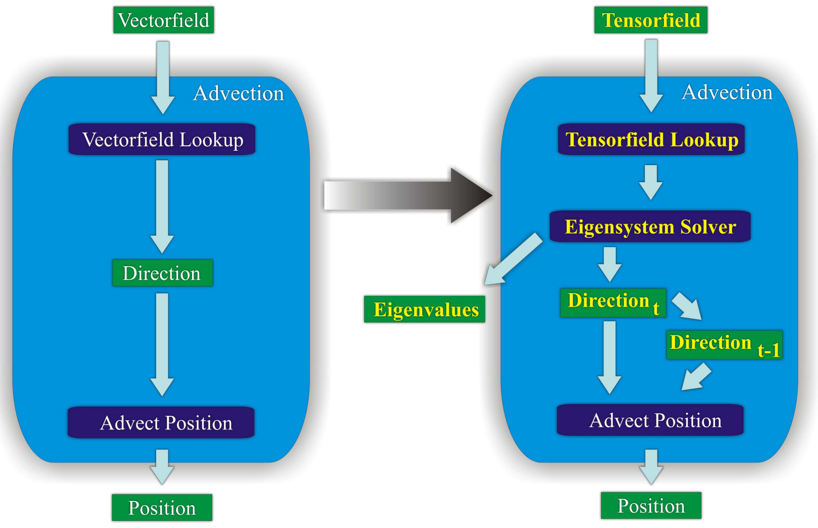


Figure 7.4: *Differences between particle tracing in a vector field and in a tensor field. Note that the advection of a particle in a tensor field as proposed in [WKL99] requires both the current and the last particle direction.*

- To derive a vector field for particle tracing, the eigen-decomposition of the resampled tensor is computed on the flyat every particle position [3]. For this purpose, we have implemented a non-iterative analytical algorithm proposed by Hasan et al. [HBPA01] on the GPU. If the tensor is not *positive-definite*, or if its eigenvalues are degenerate (equal to each other), the propagation process is terminated.

- For the rendering of textured particle sprites, two different texture atlases for tensor visualization have been designed. The first atlas extends the one proposed in [KKKW05] by an additional scaling factor used to emphasize the diffusion anisotropy $c_a = c_l + c_p$. The second one contains precomputed views of short stream tubes at different orientation and size. By texturing particle sprites with the image of the respective view – scaled according to the tensor attributes – the appearance of closed stream tubes can be simulated.

- Sice opposite eigenvector directions are both valid, particle tracking in 3D tensor fields along the largest eigenvector of the tensor can lead to ambiguous results. To avoid it, the outgoing particle direction is computed as a linear combination of the deflected incoming direction and the principal eigenvector [WKL99].

- A number of different criteria to stop particle propagation [Kin03] have been integrated. In particular, if the fractional anisotropy FA is less than a given threshold,

[3] The interpolation of precomputed eigenvectors and eigenvalues does not allow for a consistent computation of diffusion direction [Kin03, ZB02].

a particle trace is stopped. In addition, if FA is above a threshold, it is used to modulate the particle transparency. In this way, particles can be faded out continuously in nearly isotropic regions.

- The user can interactively select and change tensor-specific parameters, such as the color mapping scheme, the threshold of the fractional anisotropy used to fade out particles, and the propagation algorithm (along the largest eigenvector or along the deflected direction).

The theoretical aspects as well as implementation issues related to the application of particle tracing for tensor field visualization are covered in the next sections.

7.2.2 Propagation of Particles and Lines

The basic idea of the algorithm is quite simple. However, in order to make it efficient, on implementation stage many different aspects should be taken into account. In this section we will describe in more detail the particular implementation issues, encountered problems, and their solution.

Efficient Solver for the Eigenvalue Problem

In general, the eigenvalue problems of symmetric matrices of arbitrary size are solved using specialized iterative approaches, for example, the *Jacobi method* [PTVF07]. Since the diffusion tensor $\mathbb{D}$ is represented by a 3×3 matrix, the characteristic polynomial for this matrix is a cubic function. In this case, the eigenvalues of the tensor can be computed by finding the roots of this cubic polynomial, exploiting, for example, Cardano's solver [PTVF07]. However, there is a more efficient method to find the eigenvalues and eigenvectors, using *diffusion tensor invariants* [HBPA01]. As reported by Hasan et al. [HBPA01], this algorithm can increase the speed of eigenvalue/eigenvector calculations by a factor of 5–40 over standard iterative Jacobi or singular-value decomposition techniques.

Diffusion Tensor Invariants

The characteristic equation of the Cartesian diffusion tensor $\mathbb{D}$ is given by:

$$\det\left(\mathbb{D} - \mathbf{\Lambda}\mathbb{E}_{3\times3}\right) = -\left(\lambda^3 - \lambda^2\mathcal{I}_1 + \lambda\mathcal{I}_2 - \mathcal{I}_3\right) = 0, \tag{7.8}$$

where $\{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$ are the principal invariants of $\mathbb{D}$, which are related to the eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$, and $\mathbb{E}_{3\times3}$ is the 3×3 identity matrix.

The three invariants are defined from Equation 7.8 as follows [BT68, Bas97]:

$$\begin{aligned}
\mathcal{I}_1 &= \text{Trace}(\mathbb{D}) = D_{xx} + D_{yy} + D_{zz} = \lambda_1 + \lambda_2 + \lambda_3 && (7.9)\\
\mathcal{I}_2 &= (D_{xx}D_{yy} + D_{xx}D_{zz} + D_{yy}D_{zz}) - \left(D_{xy}^2 + D_{xz}^2 + D_{yz}^2\right) = \\
&= \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 && (7.10)\\
\mathcal{I}_3 &= \det(\mathbb{D}) = D_{xx}D_{yy}D_{zz} + 2D_{xy}D_{xz}D_{yz} - \left(D_{zz}D_{xy}^2 + D_{yy}D_{xz}^2 + D_{xx}D_{yz}^2\right) = \\
&= \lambda_1\lambda_2\lambda_3. && (7.11)
\end{aligned}$$

By using the three principal invariants $\{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$ defined by Equations 7.9-7.10, the eigenvalues and eigenvectors of $\mathbb{D}$ can be found via the analytical diagonalization of $\mathbb{D}$. Note that this diagonalization is specific to the *positive-definite symmetric* Cartesian tensors.

Determination of the Eigenvalues

The following rotationally invariant variables are defined in terms of $\{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$:

$$a = (\mathcal{I}_1/3)^2 - \mathcal{I}_2/3 \quad \text{and} \quad b = (\mathcal{I}_1/3)^3 - \mathcal{I}_1\mathcal{I}_2/6 + \mathcal{I}_3/2. \tag{7.12}$$

Since for real eigenvalues, $a > 0$ and $b^2 < a^3$, the following expression can be computed:

$$\theta = \arccos\left(\frac{b}{a}\sqrt{\frac{1}{a}}\right)/3. \tag{7.13}$$

The sorted eignevalues $(\lambda_1 > \lambda_2 > \lambda_3)$ may now be expressed as follows:

$$\begin{aligned}
\lambda_1 &= \mathcal{I}_1/3 + 2\sqrt{a}\cos(\theta),\\
\lambda_2 &= \mathcal{I}_1/3 - 2\sqrt{a}\cos(\pi/3 + \theta),\\
\lambda_3 &= \mathcal{I}_1/3 - 2\sqrt{a}\cos(\pi/3 - \theta).
\end{aligned} \tag{7.14}$$

Note that the eigenvalues do not require sorting due to the cosine function and the fact that $0 \leq \theta \leq \pi/3$ is assured for a positive-definite diffusion tensor matrix. The third eigenvalue can also be obtained using the trace invariance property, $\lambda_3 = \mathcal{I}_1 - \lambda_1 - \lambda_2$.

Determination of the Eigenvectors

After the sorted eigenvalues have been computed, the i^{th} $(i = 1, 2, 3)$ eigenvector, $\mathbf{e_i} = [e_{ix}, e_{iy}, e_{iz}]^{\text{T}}$, may be calculated by solving the linear system of equations, $\mathbb{D}\mathbf{e_i} = \lambda_i\mathbf{e_i}$. Due to the fact that $(-\mathbf{e_i})$ is also a solution to the eigenvector problem $\mathbb{D}(-\mathbf{e_i}) = \lambda_i(-\mathbf{e_i})$, a sign ambiguity in the vector direction is *unavoidable*.

The calculation of the orthonormalized eigenvectors proceeds as follows:

$$\begin{aligned} A_i &= D_{xx} - \lambda_i; \quad B_i = D_{yy} - \lambda_i; \quad C_i = D_{zz} - \lambda_i; \\ e_{ix} &= \left(D_{xy}D_{yz} - B_iD_{xz}\right)\left(D_{xz}D_{yz} - C_iD_{xy}\right), && (7.15) \\ e_{iy} &= \left(D_{xz}D_{yz} - C_iD_{xy}\right)\left(D_{xz}D_{xy} - A_iD_{yz}\right), && (7.16) \\ e_{iz} &= \left(D_{xy}D_{yz} - B_iD_{xz}\right)\left(D_{xz}D_{xy} - A_iD_{yz}\right), && (7.17) \end{aligned}$$

where $i = 1, 2, 3$. The normalized eigenvector corresponding to λ_i is then computed as:

$$\hat{\mathbf{e}}_\mathbf{i} = \mathbf{e_i}/\sqrt{\mathbf{e_i}^\mathrm{T}\mathbf{e_i}}. \tag{7.18}$$

Note that the third eigenvector can be also obtained via the cross product between the other two orthonormal eigenvectors $\hat{\mathbf{e}}_\mathbf{1}$ and $\hat{\mathbf{e}}_\mathbf{2}$.

Cautions

Prior to obtaining the eigenvalues and eigenvectors, it is strongly suggested [HBPA01] to enforce a nonnegative-definite real eigenvalue mask[4]. This mask includes all voxels satisfying the necessary and sufficient conditions that assure the convexity of the diffusion ellipsoid [Bel60]:

$$\mathcal{I}_3 > 0, \text{ and } \left(D_{ij} \text{ and } D_{ii}D_{jj} - D_{ij}^2\right) \geq 0, \quad \text{for } i, j = x, y, \text{ or } z. \tag{7.19}$$

Since diffusion measurements usually have some random noise that makes the tissue water diffusivities unique and rarely equal, the enforcement of this positive-definite mask additionally assures that:

$$(\lambda_1, \lambda_2, \text{ and } \lambda_3) > 0, \tag{7.20}$$

which will reduce the chance of division by 0 in Equation 7.18. The mask will remove very-low-signal regions and regions outside of the investigated tissue. However, it will not remove the voxels with degenerate eigenvalues, where $a = 0$ (see Equation 7.12). This case rarely occurs in high quality measurements of diffusion processes in biological tissue and corresponds to highly isotropic regions.

Adjusting Particle Tracing for Diffusion Tensor Fields

While classical particle tracing in vector fields is quite simple, it is more computationally involved in diffusion tensor fields and a number of different problems have to be

[4] Interestingly, we have found out that if the matrix is symmetric but *not* positive-definite, the algorithm still returns the correct eigenvalues.

resolved during the computations. Firstly, as it was depicted on the comparative block diagram presented in Figure 7.4, and briefly summarized in Section 7.2.1, the direction of tracing first has to be derived using the eigenvectors of diffusion tensor $\mathbb{D}$. The eigenvector $\hat{\mathbf{e}}_\mathbf{1}$ corresponding to the largest eigenvalue λ_1 is a candidate for the direction of particle advection. The eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ and the eigenvector $\hat{\mathbf{e}}_\mathbf{1}$ are efficiently computed using the non-iterative algorithm described in the previous subsection and implemented on the GPU.

Once the eigenvector $\hat{\mathbf{e}}_\mathbf{1}$ is computed, the next problem arises. As it was shown in the previous subsection, both of the eigenvectors $\hat{\mathbf{e}}_\mathbf{1}$ and $-\hat{\mathbf{e}}_\mathbf{1}$ are valid solutions for the related eigenproblem [DH92]. To resolve this ambiguity, the assumption of continuity and smoothness of tensor fields in general [DH92] and, in particular, in biological tissue under normal conditions can be exploited [WKL99]. Thus, starting with either direction of eigenvector $\hat{\mathbf{e}}_\mathbf{1}(t = 0)$, we proceed by testing whether the currently computed eigenvector $\hat{\mathbf{e}}_\mathbf{1}(t)$ is co-aligned[5] with the eigenvector computed on the previous advection step $\hat{\mathbf{e}}_\mathbf{1}(t - 1)$. If this is not the case, we flip the sign of the current eigenvector $\hat{\mathbf{e}}_\mathbf{1}(t)$.

However, with only flipping of the sign, the problem of correct particle tracing is not yet resolved. In addition, the following observations should be taken into account. Due to the stochastic nature of the diffusion process, the eigenvector $\hat{\mathbf{e}}_\mathbf{1}$ only shows the most probable direction of propagation of water molecules through the tissue. There exist regions within the tissue where the probability of diffusion in all directions is nearly equal. These regions are characterized by the high degree of isotropy (or low degree of anisotropy). Thus, in such regions the direction of the eigenvector does not correspond to the actual direction of diffusion and the classical tracing algorithm becomes unstable. To cure this problem, Weinstein et al. [WKL99] proposed a heuristical algorithm for stabilized propagation of tensor lines, based on the *deflection* principle. The deflection direction $\mathbf{d}_{\text{def}}$ is simply the incoming direction vector $\mathbf{d}_{\text{in}}$ (from the previous advection step), transformed by the tensor:

$$\mathbf{d}_{\text{def}} = \mathbb{D}\,\mathbf{d}_{\text{in}}. \tag{7.21}$$

In order to make the propagation vector properly scaled with respect to the diffusion term, a normalization of the input tensor field should be performed [Kin03]. Thus, in a preprocess step, all the values in the tensor field are scaled by a factor of $(2/\lambda_{\max})$, where $\lambda_{\max}$ is the largest eigenvalue found within the whole data set.

[5] This is done by a simple sign-test of the dot-product between the eigenvectors.

Given vectors $\hat{\mathbf{e}}_1$, $\mathbf{d}_{\text{in}}$, and $\mathbf{d}_{\text{def}}$, the propagation direction is computed as the linear combination of these vectors:

$$\mathbf{d}_{\text{out}} = c_l\,\hat{\mathbf{e}}_1 \;+\; (1-c_l)\Big((1-\omega_{\text{punct}})\,\mathbf{d}_{\text{in}} \;+\; \omega_{\text{punct}}\,\mathbf{d}_{\text{def}}\Big), \tag{7.22}$$

where the coefficient c_l is the anisotropy metric defined in Table 7.1 and ω_{punct} is a user-controlled parameter, which determines the relative contribution of $\mathbf{d}_{\text{in}}$ and $\mathbf{d}_{\text{def}}$. This coefficient can take values from 0 to 1 and it is highly dependent on the data under investigation. Figure 7.5 illustrates the effect of application of (a) the standard method and (b) the heuristical algorithm based on the deflection direction for tensor line propagation.

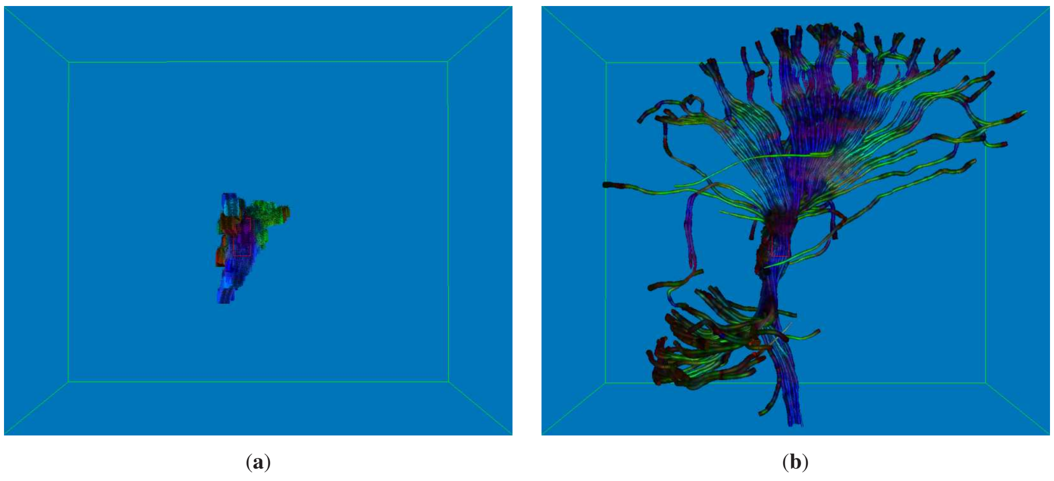


Figure 7.5: *Effect of applying different techniques for tensor line propagation: (**a**) standard line propagation technique, (**b**) heuristical algorithm with deflection.*

In Figure 7.5 (a) the disadvantage of the standard propagation technique can be clearly observed. Due to the unresolved ambiguity related to the choice of direction for eigenvector $\hat{\mathbf{e}}_1$, the line segments in this case are wasted on “reversing” the direction and tracing forwards and backwards multiple times. On the other hand, the application of the heuristical algorithm yields smooth appearance of tensor lines revealing the prominent structures within the human brain data set. The red framebox in Figure 7.5 corresponds to the seeding probe for starting positions of tensor lines.

There is one more problem, untypical for particle tracing in most flow scenarios. The flow fields are usually defined for each voxel within the flow volume (cf. Section 2.1.1 for the definition of the flow field based on the notion of continuum). Thus, the particle tracing can be performed *everywhere* within the flow domain. In contrast, this is not the case for diffusion tensor fields. For instance, in highly isotropic regions, there is no preferred direction of diffusion, the water molecules are diffusing randomly in all directions. Such regions usually do not contain the structures, which could be of interest from the medical or physical point of view. Thus, without the loss of generality, the particles should be stopped in these regions[6]. Moreover, taking into account that data sets of biological tissue usually contain the tissue itself surrounded by an air environment (for example, a DT-MRI scan of a human brain), it is clear that no particle tracing should be performed outside of the biological tissue under investigation. Therefore, for biological data sets, a bit-mask `ValidBitMask` of valid regions within the volume is usually provided with the data set. Sometimes, invalid regions are contained inside of the biological tissue, caused by imperfection of measurement procedures. The cases when the propagation of a particle or a line must be terminated are summarized in Table 7.2.

Table 7.2: *Propagation termination criteria for diffusion tensor fields.*

Termination criteria for $\mathbb{D}$ - fields	Usual for particle tracing	Particle age is expired, Particle position $\mathbf{p}(t)$ is out of flow domain.
	$+$	
	Diffusion tensor specific	$\mathbb{D}(\mathbf{p}(t))$ is degenerate or non–positive–definite (see cautions given by Equations 7.19, 7.20), `ValidBitMask`($\mathbf{p}(t)$) indicates invalid data.

According to [KKKW05], after the propagation is terminated, the particles should reincarnate at their initial positions. Applied to diffusion tensor fields, this approach also requires a modification. It can happen that the particles are initiated close to the regions, where they have to stop rather soon after they started. Thus, reincarnating them at the same starting position will lead to high frequency flickering, which can fatigue the user and distract him from focusing on more important regions in the data set. We found out that a better choice is to reincarnate the particle every time in a new position, using a shift in a random direction from the stored starting position. There is no randomize function available on the current graphics hardware, therefore it is not possible to randomize the initial particle positions on the GPU every time it is necessary[7], as it can be easily done on the CPU. Fortunately, in our application, it is sufficient to provide a single randomized direction and shift all the starting positions of the currently reincarnating particles along this vector.

[6] Otherwise, these particles will chaotically move in a close vicinity of their initial positions that can distract the user from seeing the important structures.

[7] It can be done on the CPU and then uploaded to the GPU, however, this will significantly drop the performance.

Line-Packing Technique for Tensor Lines

While there is no conceptual difference between tracing the lines or particles, the rendering techniques are obviously different for these geometrical objects. While the particles are unconnected discrete objects, the line is a piecewise continuous representation of a particle path. The construction and rendering of both objects for classical particle tracing applications is quite simple. According to the line tracing algorithm proposed by Krüger et al. [KKKW05] and described in Section 5.2.2, *all* the lines have the same number of vertices assigned to them. While this strategy is very efficient for the classical particle tracing, it results in a waste of memory for the particle tracing applied to diffusion tensor fields. Taking into account the additional data-specific termination criteria described in the previous subsection, the problem is as follows. If a line started in a close vicinity to a region marked as invalid, it will be terminated after a few segments. In this case, the rest of the vertices will be unused. Assume that the line length is 1000 segments, and there is a hundred of these lines to be displayed. Termination of 30% of the lines after 100 advection steps leads to wasting 27000 of 100000 segments. The result displayed on the screen will be much sparser than if we would draw the same number of lines with the same length in a standard vector field.

In order to resolve this problem, we suggest to restart the line at some random position in a way similar to particle reincarnation. More precisely, if the line is terminated, we reuse the rest of the memory resources allocated for this line object to trace another line, started at a random position. In this way, several short lines can be packed into the resource of size equal to the default line length. Exploiting this *line-packing* technique, attention should be paid that the "jump" regions between the lines are blended out[8].

The idea behind the line-packing technique is illustrated by example in Figure 7.6. Since one line-restart occurs during the construction process, the vertex buffer is shared between two tensor lines. Similar to particle tracing for vector fields, we use the following notation: `PosTex` for the position texture and `DirTex` for the direction texture. Indices "0" and "1" are assigned to these textures according to a ping-pong technique. The current position and direction are represented by colored rectangles: light blue for $Line_1$ and pink for $Line_2$. The jump-flag is visualized as a thin box attached to a colored rectangle associated with the current position. The empty box corresponds the jump-flag turned off, and the filled black box signalizes that the jump-flag is switched on.

To randomize the restarting positions of the lines we use a similar strategy as for

[8] For this purpose we use the `discard` statement available in HLSL [HLS, Gra03], which allows to discard the current fragment.

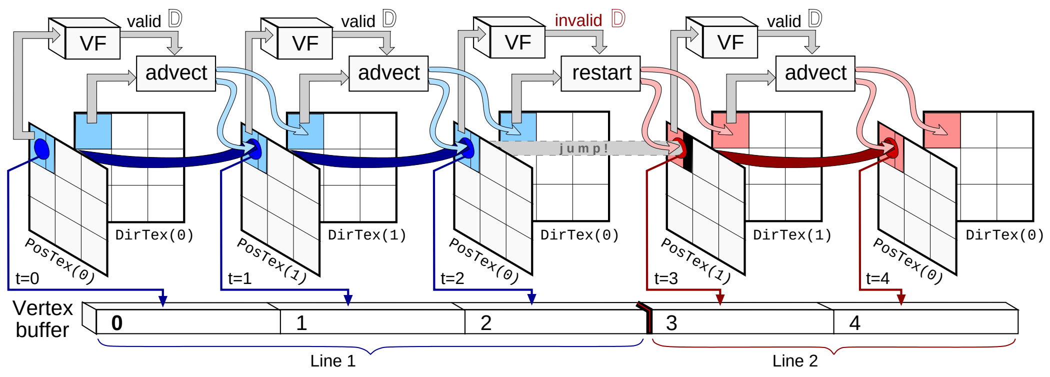


Figure 7.6: *Sketch of the line-packing technique: one line-restart yields two tensor lines packed into one vertex buffer. The current position and direction are emphasized by the colored rectangles. The thin black box corresponds to the jump-flag switched on.*

particles. For each advection step, a random shift vector $\mathbf{RndDir}$ is forwarded to the shader. If the line has to be restarted, $\mathbf{RndDir}$ is added to the position stored in the starting position texture `StartPosTex` for the lines. The special jump-flag (`w`-component of the pixel value in `PosTex`) is used in order to distinguish between the "jump" segment and the segment belonging to a tensor line. The pseudo-code for one advection step of the tensor line propagation algorithm with the line-packing technique is depicted in Figure 7.7. Note that this step is performed as a per-pixel operation for each line in DirectX Pixel Shader [Gra03].

In Figure 7.7, Δt is the advection step size for the Euler integration scheme. The purpose of the other input and output variables (textures) is as follows:

Input variables:

3D textures: `TensorField` represents two textures, which store the tensor values and the validity bit-mask `ValidBitMask` available from the measurements;

2D textures: `StartPosTex` stores the starting positions for the lines; `PosTex`$(t-1)$ and `DirTex`$(t-1)$ are the textures, which store the positions and directions respectively computed in the previous advection step for all lines;

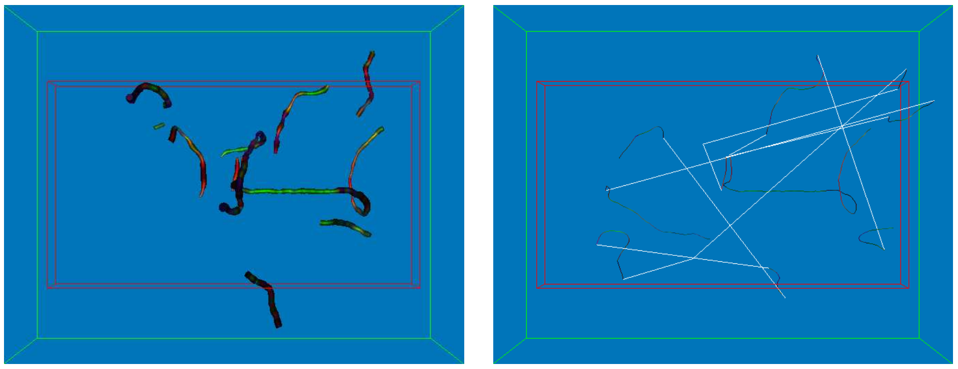
Output variables:

2D textures: `PosTex`(t), `DirTex`(t), `EigenvaluesTex`, `EigenvectorTex` store all line attributes (positions, directions, eigenvalues, and eigenvectors, respectively) computed in the current advection step for all lines.

The effect of the application of the line-packing technique is depicted in Figure 7.8. While the texture resources were allocated for four lines of the same length, line-packing traces ten lines of shorter length (Figure 7.8 (a)). The line "jumps" are explicitly visualized as white line segments in Figure 7.8 (b).

```
LinePropagation(TensorField, StartPosTex, RndDir, $\Delta t$, lineIdx$[i,j]$,
                PosTex$(t-1)$, DirTex$(t-1)$ ) {
1.   RestartPos = StartPosTex$(t-1)$ + RndDir;
     // Get tensor field values via tri-linear interpolation
2.   if(PosTex$(t-1)$.w is true)            // Last position is valid
3.     $\mathbb{D}$ = TensorField(PosTex$(t-1)$);
4.   else
5.     $\mathbb{D}$ = TensorField(RestartPos);
6.   EigenvaluesTex = ComputeEigenvalues($\mathbb{D}$);
     // Line propagation
     //--------------------------------------
7.   if($\mathbb{D}$ is valid ) {           // See cautions given by Equations 7.19 and 7.20
8.     EigenvectorTex  = ComputeEigenvector_E1($\mathbb{D}$, EigenvaluesTex);
       // Compute the propagation direction according to Equation 7.22
9.     DirTex$(t)$       = Deflection(DirTex$(t-1)$, EigenvectorTex, EigenvaluesTex);
10.    PosTex$(t)$       = PosTex$(t-1)$ + DirTex$(t)\cdot\Delta t$;
11.    PosTex$(t)$.w     = true;            // Position is not a jump
12.  } else {
13.    EigenvectorTex  = [1,0,0];
14.    DirTex$(t)$       = EigenvectorTex;
15.    PosTex$(t)$       = RestartPos;
16.    PosTex$(t)$.w     = false;           // Position is a jump
     //--------------------------------------
18.  if( PosTex$(t)$ is out of flow domain )
19.    PosTex$(t)$       = PosTex$(t-1)$;
 }
```

Figure 7.7: *Advection step of the tensor line propagation algorithm with line-packing.*



(**a**) (**b**)

Figure 7.8: *Line-packing technique: (**a**) effect of application: 10 lines of different lengths computed vs. 4 lines of the same length allocated, (**b**) visualization of "jump" segments (white).*

7.2.3 Rendering Aspects

Conceptually, all rendering modes employed for diffusion tensor field visualization are similar to the modes used for classical vector field visualization and presented in

[KKKW05]. However, a number of data-specific modifications and extensions, described in this section, had to be added.

Extension of the Sprite Atlas Techniques for Diffusion Tensor Fields

An efficient rendering technique for aligned glyphs is presented in [KKKW05]. Instead of real geometry this approach displays textured point sprites. The texture attached to the point sprites represents a projection of a real geometry object oriented according to the specified transformation matrix. A number of such object projections captured from different points of view onto the real object are stored into the sprite atlas. For rendering the point sprites, an appropriate sub-image from the whole sprite atlas should be selected. This selection is performed in a fragment shader via the transformation $\mathbb{M}_{\text{dir}}$ of texture coordinates assigned to the point sprite by the GPU (cf. Equation 5.3). A short overview of this technique is given in Section 5.2.1.

With regard to diffusion tensor field visualization, the orientation of the ellipsoid is chosen in the same way as for classical particle tracing: along the direction of the particle propagation. However, the meaning of the shape of the rendered primitive for a particle is of high importance. As depicted in Table 7.1, the elongated cigar-shaped tensor ellipsoids correspond to regions characterized by linear type of anisotropy, typical for fibrous structures within the biological tissue. On the other hand, ellipsoids degenerated into a sphere represent highly isotropic regions within the real matter, for example, cerebral fluid in human brain. Therefore, instead of arbitrary scaling of the aligned sprites, the diffusion tensor specific scaling based on some anisotropy measure is required.

In our implementation we use the overall anisotropy metric $c_a = c_l + c_p$. As denoted in [Kin03, ZB03], the linear c_l and planar c_p metrics both increase in regions with higher anisotropy (see the definitions of c_l and c_p given in Table 7.1), the anisotropy metric c_a can be used as an alternative to the FA or RA metrics described in Section 7.1.1. Note that by construction c_a never takes values beyond the range $[0, \ldots, 1]$[9], and thus can be efficiently employed to adjust the general sprite atlas for the visualization of diffusion tensor fields.

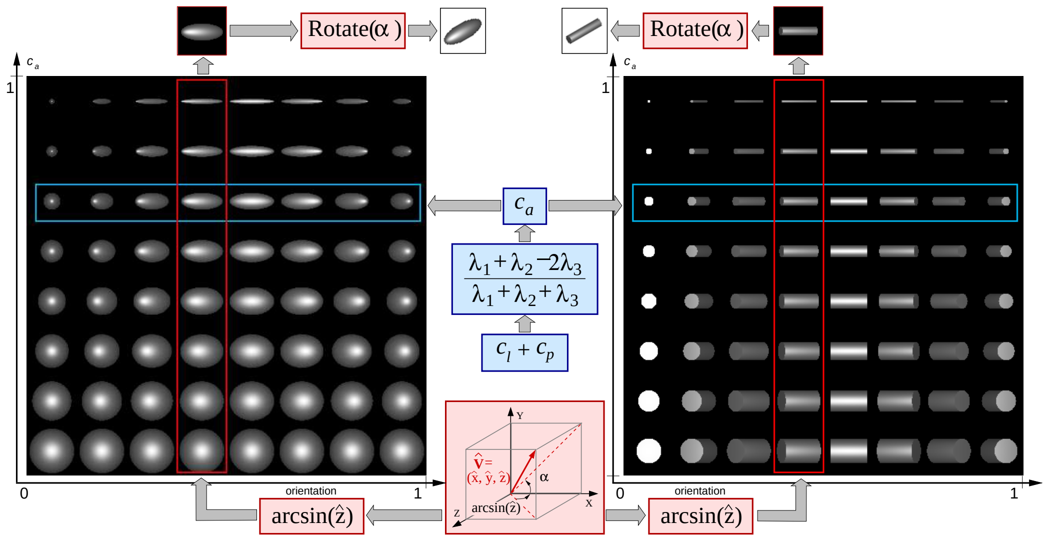
Taking into account the aforementioned observations, the diffusion tensor specific sprite atlas can be constructed as follows. The sphere is chosen to be the starting shape (scale 1.0) for the sprite atlas. The other rows are created by deforming (scaling) the sphere along two principal directions down to scale 0.0. In this way, the sprite atlas,

[9] In the isotropic regions $c_a = 0$ and in highly anisotropic regions $c_a \approx 1$.

involving the full spectrum of ellipsoids for all the discrete values of the c_a anisotropy measure, can be created (see Figure 7.9). For rendering, the appropriate row from the sprite atlas is chosen based on the c_a measure. For instance, if $c_a = 0$, the first row of the atlas will be chosen (spheres), or if $c_a = 1$, the last row of the atlas will be chosen (thin cigar-shaped ellipses). Hence, the transformation matrix $\mathbb{M}_{\text{dir}}$ given by Equation 5.3 can be rewritten as follows:

$$\begin{aligned} \mathbb{M}^{\text{DT}}_{\text{dir}} &= \mathbb{M}^{\text{DT}}_{\text{trans}} \cdot \mathbb{M}_{\text{rot}}, \\ \text{where} \quad \mathbb{M}^{\text{DT}}_{\text{trans}} &= \text{Translate}\big(\arcsin(\hat{z}),\ c_a\big) \\ \mathbb{M}_{\text{rot}} &= \begin{bmatrix} \hat{x} & \hat{y} \\ -\hat{y} & \hat{x} \end{bmatrix}, \end{aligned} \tag{7.23}$$

The process of selection of an appropriate picture from the sprite atlas and its transformation according to $\mathbb{M}^{\text{DT}}_{\text{dir}}$ is illustrated in Figure 7.9.



Rotate(α)
Rotate(α)
c_a
1
1
0
1
0
1
orientation
orientation
c_a
$\frac{\lambda_1+\lambda_2-2\lambda_3}{\lambda_1+\lambda_2+\lambda_3}$
$c_l + c_p$
Y
$\hat{v}=(\hat{x}, \hat{y}, \hat{z})$
α
arcsin(ẑ)
Z
X
arcsin(ẑ)
arcsin(ẑ)

Figure 7.9: *Rendering of aligned glyphs using the sprite atlas. The transformation of texture coordinates of a point sprite is performed according to $\mathbb{M}^{\text{DT}}_{\text{dir}}$ given by Equation 7.23.*

Tensor tubes

For the rendering of tensor tubes we use similar techniques as for particles. In this way, we do not render the real tube geometry, but a number of closely placed point sprites[10]. To make a collection of sprites appear as a smooth tube, a cylinder is used as a prototype shape for the tensor tube sprite atlas. The cylinder, whose section through the main axis is a square, is used for the first row of the atlas corresponding to the

[10] For tensor lines, however, we use the real geometry, i.e., the connected line segments.

isotropic case (see Figure 7.9). For rendering, instead of connecting the line positions with line segments, we display a point sprite with the image of a projected cylinder at these line positions. If the integration step size is chosen to be sufficiently small, the discontinuities between the cylinders are not noticeable in the final image. If the step size is large, then the separate cylinders can be observed. In this case, the tensor tube mode simply converges to the mode of diffusion aligned glyphs of cylindrical shape.

Blending Options and Color Mapping

A number of diffusion tensor based blending options and color mapping schemes have been integrated into the particle system for tensor field visualization. The user has the possibility to vary the corresponding thresholds and choose between the different color mapping schemes. The variety of color mapping schemes implemented is listed in Section 7.1.1. Once the eigenvalues and eigenvectors are computed and stored into the 2D texture objects for particles or for tensor tubes, they can be used to compute the color mapping of the user's choice on the fly. Since the formulae for color mapping schemes are very simple, computing them every time the user has chosen another scheme does not induce a drop of performance. Moreover, if the colors for each particle or line segment are computed on the fly, there is no need to store them in extra texture objects, thus, the valuable video memory can be preserved for other purposes.

One very important aspect is the transparency of the rendered primitives and their size. Typically, the isotropic regions are not of interest in medical applications. Therefore, the particles (represented by spheres) in such regions should not be rendered at all. Since the c_a measure is computed on each advection step for each particle, we check its value, and discard the particle completely if c_a corresponding to its current position is 0. On the other hand, the regions of low non-zero anisotropy could be still of interest or they could be used as a context for other more important regions in biological tissue. Taking into account that the most important regions correspond to fiber tracks, which are characterized by a degree of anisotropy close to 1.0, a smooth blending function between the highly isotropic and highly anisotropic regions can be employed.

In general, the blending function should satisfy the following conditions. The upper-bound case: if the regions are characterized by $c_a > \sigma_{\text{iso\_max}}$, they must be rendered absolutely opaque. The lower-bound case: if the regions are associated with $c_a < \sigma_{\text{iso\_min}}$, they must be completely discarded. The normal case: if regions have $\sigma_{\text{iso\_min}} \leq c_a \leq \sigma_{\text{iso\_max}}$, some smooth blending equation should be employed to render them. Additionally, the validity bit-mask `ValidBitMask`[11] available from the

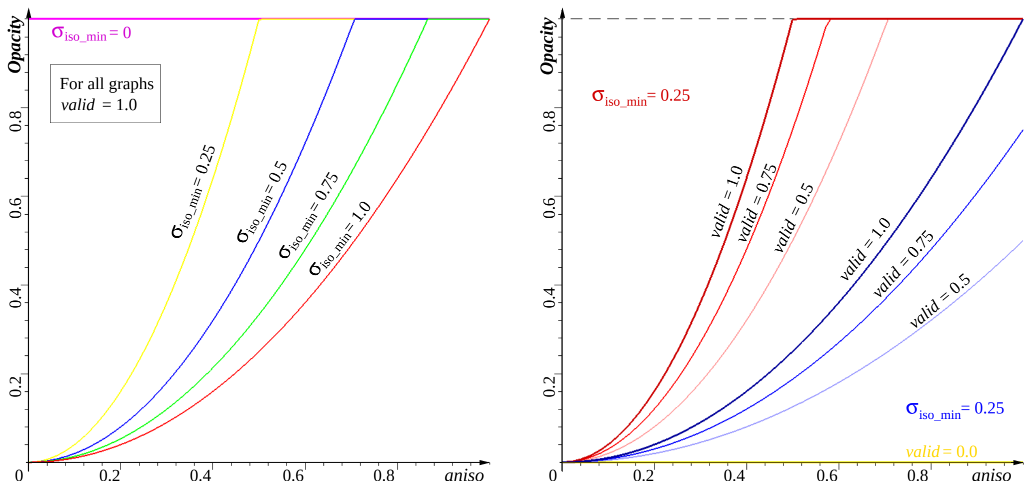
[11] Each "bit" of the validity bit-mask is denoted for further use as $validBit$.

measurement procedure should be applied in order to completely discard the regions marked as "invalid" according to this bit-mask. Note that in general, the bit-mask does not exactly store bits, but rather *floating point* scalar values. In this case, the values contained in such a bit-mask can be interpreted as the probabilities of the corresponding tensor values to be valid (1.0) or invalid (0.0). Therefore, an additional *validity threshold* σ_{valid} can be involved in the transfer function. In the current implementation the following transfer function $\boldsymbol{Opacity}(aniso)$ for opacity values is employed:

$$valid = \min\Big((validBit/\sigma_{\text{valid}}),\ 1.0\Big) \tag{7.24}$$

$$\boldsymbol{Opacity}(aniso) = \begin{cases} \sigma_{\text{iso\_min}} > 0: & \min\left(\dfrac{aniso^2 \cdot valid}{\sigma_{\text{iso\_min}}},\ 1.0\right) \\ \text{Otherwise}: & 0, \end{cases} \tag{7.25}$$

where $aniso$ is some anisotropy metric, for example, c_a or FA. The upper bound of the transfer function is modulated using the scaling by $\sigma_{\text{iso\_min}}$. The families of graphs of the transfer function $\boldsymbol{Opacity}(aniso)$ with parameters $\sigma_{\text{iso\_min}}$ and $valid$ are depicted in Figure 7.10.



(a) (b)

Figure 7.10: *Families of the transfer function for opacity* $\boldsymbol{Opacity}(aniso)$*: (**a**) for* $valid = \text{const}$ *and varying* $\sigma_{\text{iso\_min}}$*, (**b**) for varying* $valid$ *and two fixed* $\sigma_{\text{iso\_min}}$ *values, 1.0 (blue) and 0.25 (red).*

As can be observed from the graphs in Figure 7.10 (a), the smaller $\sigma_{\text{iso\_min}}$, the steeper is the graph of the transfer function. In the boundary case, when $\sigma_{\text{iso\_min}} = 0$, the graph degenerates into the unit step function, which means that all the primitives will be rendered *opaque*, independently of the degree of anisotropy associated with their positions.

The effect of scaling with an additional parameter, $valid$, can be clearly observed in Figure 7.10 (b). If $\sigma_{\text{valid}} = 1$, then only 100% valid data is rendered as absolutely opaque. In this case, $valid$ is solely defined by the probability given in $validBit$. The lower this probability, the less opacity should be assigned to the displayed data. This is illustrated by the graphs unbending towards the $aniso$-axis and approaching it at the boundary case ($valid = 0$). If $valid = 1$, then the opacity of the displayed data is solely modulated by the anisotropy level $aniso$ and the threshold σ_{min}. By construction, $0 \leq valid \leq 1$, thus the situation when $valid > \sigma_{\text{valid}}$ does not make the graph of the transfer function any steeper than for the boundary case $\sigma_{\text{valid}} = 1$.

It is worth mentioning that when semi-transparent objects are present in the scene, correct ordering of these objects in 3D space is a crucial issue. In our application, we can choose between two options: real sorting of particles and corresponding attributes using the bitonic sorting algorithm implemented on the GPU [KKKW05] or a “fake” sorting mode, enabled via additive blending [BKKW08]. When the latter case is chosen, the opaque and almost opaque objects should be rendered with first the depth-write-test enabled, and then the rest of the objects (semi-transparent objects) are rendered using the additive blending mode with the depth-write-test disabled.

7.3 Results and Performance

In this section, we demonstrate the efficiency of the proposed particle tracing method for visualization of diffusion tensor fields in terms of computational costs and rendering performance as well as its effectiveness for the exploration and gaining insight into the real-world data sets of biological tissue. The following two data sets were investigated in our rendering experiments: the human brain data set of resolution $148 \times 190 \times 160$ and the canine heart data set of resolution $256 \times 256 \times 256$. Each entry in both data sets consists of six diffusion tensor components and an additional validity bit $validBit$. Different visualization options and color mappings were selected to emphasize particular diffusion properties and anatomical structures in these data sets. In all examples, visual exploration was performed in real-time, thus allowing for an effective and intuitive analysis of biological tissue.

7.3.1 Performance

All of our experiments have been conducted on a NVIDIA Quadro FX 4400 graphics card equipped with 512 MB video memory. Rendering was performed into a viewport of 1280×1024 pixels. Table 7.3 gives timings for the advection and rendering of particles. Timings in the second column include all operations that are carried out until updated particle positions are available in the current particle container, i.e., tensor interpolation, eigen-decomposition, and particle advection using Euler integration. The third and the fourth columns include timings for the rendering of diffusion-aligned point sprites. To demonstrate the dependency of performance from the number of generated fragments, differently sized sprites are used. The last column shows the time that is required to reconstruct different amounts of stream lines of length 100. In all these tests, a 256^3 tensor data set (canine heart) was visualized.

As can be seen, even for large particle sets the GPU implementation still allows for the interactive visual exploration of tensor fields. It is interesting to note that with increasing fragment size the rendering stage quickly becomes fragment bound. Moreover, the number of generated fragments strongly depends on viewing parameters, such as field of view and the distance of particles to the camera. As it is rather difficult to provide precise and meaningful timing statistics for different amounts of particles in combination with differently sized point sprites, we give specific timings for all the generated images shown in the following. Also, since for fibers the number of advection steps is proportional to their length, the approximate time for tracking and rendering a fiber of an arbitrary length can be computed as the time needed for one segment multiplied by the number of segments.

particles	advection only	oriented splats		*lines*	length 100
		1×1 pixels	7×7 pixels		
64^2	1434	700	470	64	130
128^2	343	185	123	128	100
256^2	83.2	43.7	28.8	256	78
512^2	19.4	8.4	5.9	512	55
1024^2	4.2	2.0	1.6	1024	38

Table 7.3: *Application performance in frames per second, for different amounts of particles and for stream tubes each consisting of 100 segments.*

In contrast to previous approaches, where only the final rendering of precomputed entities can be done interactively, the proposed method provides an even more intuitive means to explore high resolution diffusion tensor fields. In particular, the user can interactively select the seeding density of particles as well as parameters specific to the propagation process. By using these options, pathways can be visualized at almost arbitrary resolution. The advection of oriented particles allows for the simultaneous exploration of both local *and* global diffusion tensor properties. Fiber structures can be observed without that particle positions have to be connected. Even in still images the fibrous structures in the investigated tissue can clearly be seen.

7.3.2 Rendering Examples

In this section, we show screenshots of the dynamic 3D tensor field visualization including various visualization options (Figures 7.11 to 7.17). Although the images already show the functionality of the particle engine, we should note here that the real benefit of the presented approach is a lot more apparent in the animation.

In Figure 7.11, colored point sprites of size 5×5 pixels were rendered to visualize the human brain data set. The FA-based color mapping scheme (Equation 7.6) was used. By fading out particles in regions showing low anisotropy, highly anisotropic brain structures, such as corpus callosum, corona radiata and pyramid, are emphasized and can clearly be distinguished in the generated image (cf. Figure 7.2).

In Figure 7.12, diffusion-dependent oriented sprites were used to render a close up view of the corona radiata. According to the FA-based color mapping scheme, an optic tract can be clearly distinguished by the green color[12]. To generate the image, 64K point sprites of size 45×45 pixels were rendered at 6.1 frames per second.

The left image in Figure 7.13 shows a visualization of the corona radiata using stream tubes. The seeding probe from which particle traces were initially released is shown as wire frame in red. Overall, 512 traces of a maximum length of 600 were traced. Textured point sprites of size 15×15 pixels were rendered using the texture atlas described above. Reconstruction and rendering was performed at 2.4 frames per second. As can be seen, this visualization option helps to track the bunches of fibers with biologically similar properties, and it allows the expert to follow the paths of single fibers. On the right of Figure 7.13 the superior longitudinal fasciculus is rendered by means of colored stream lines. The axial slice along the fasciculus was chosen as seeding region. 512 lines of a maximum length of 600 were computed and displayed at 10 frames per second. Both images show nicely the property of stream tubes and stream lines to effectively reveal spatial relationships between different tracks.

The axial, coronal, and sagittal slices through the human brain are visualized in Figure 7.14 using diffusion-dependent sprites. In combination, shape and color of the rendered ellipsoids give a good impression of the degree of anisotropy and the main diffusion direction within the corresponding brain structures. The brain structures themselves, such as pyramidal tracts, U-shaped fibers, superior longitudinal fasciculus and

[12] The main diffusion direction is along the Y-axis, which corresponds to the green channel of the RGB color scheme.

others, are clearly visible in all slices. The axial and sagittal slices were rendered using 64K particles of size 20×20 pixels at 10.8 frame per second. The coronal slice was produced using 256K particles of size 10×10 pixels at 4.5 frames per second.

Visualizations of the canine heart data set are shown in Figures 7.15–7.17. Figure 7.15 shows the longitudinal and latitudinal heart slices, rendered with textured point sprites. The orientation of the sprites corresponds to the helical construction of the heart muscle, and the spatial positions depict the heart structure consisting of four chambers and several valves in between. The animation of 64K particles of size 25×25 pixels runs at roughly 8 frames per second.

Figure 7.16 shows images of the heart data set rendered from different view points using aligned diffusion-dependent point sprites. In the interactive animation, the clockwise and counter-clockwise motion of particles along the muscle structure can be clearly tracked. Both images were rendered using 64K particles at 25×25 and 45×45 pixels per point sprite, respectively. Accordingly, the application performance dropped from 9.4 frames per second to 7.4 frames per second.

Stream tubes were used in Figure 7.17 to visualize the heart data set. On the right, tubes were colored according to Equation 7.4. Both images were generated using 1K stream tubes of maximum length of 1K in combination with point sprites of size 5×5 pixels to render the texture-based tube segments. Reconstruction and rendering of stream tubes was performed at 1.5 frames per second.

(**a**)

(**b**)

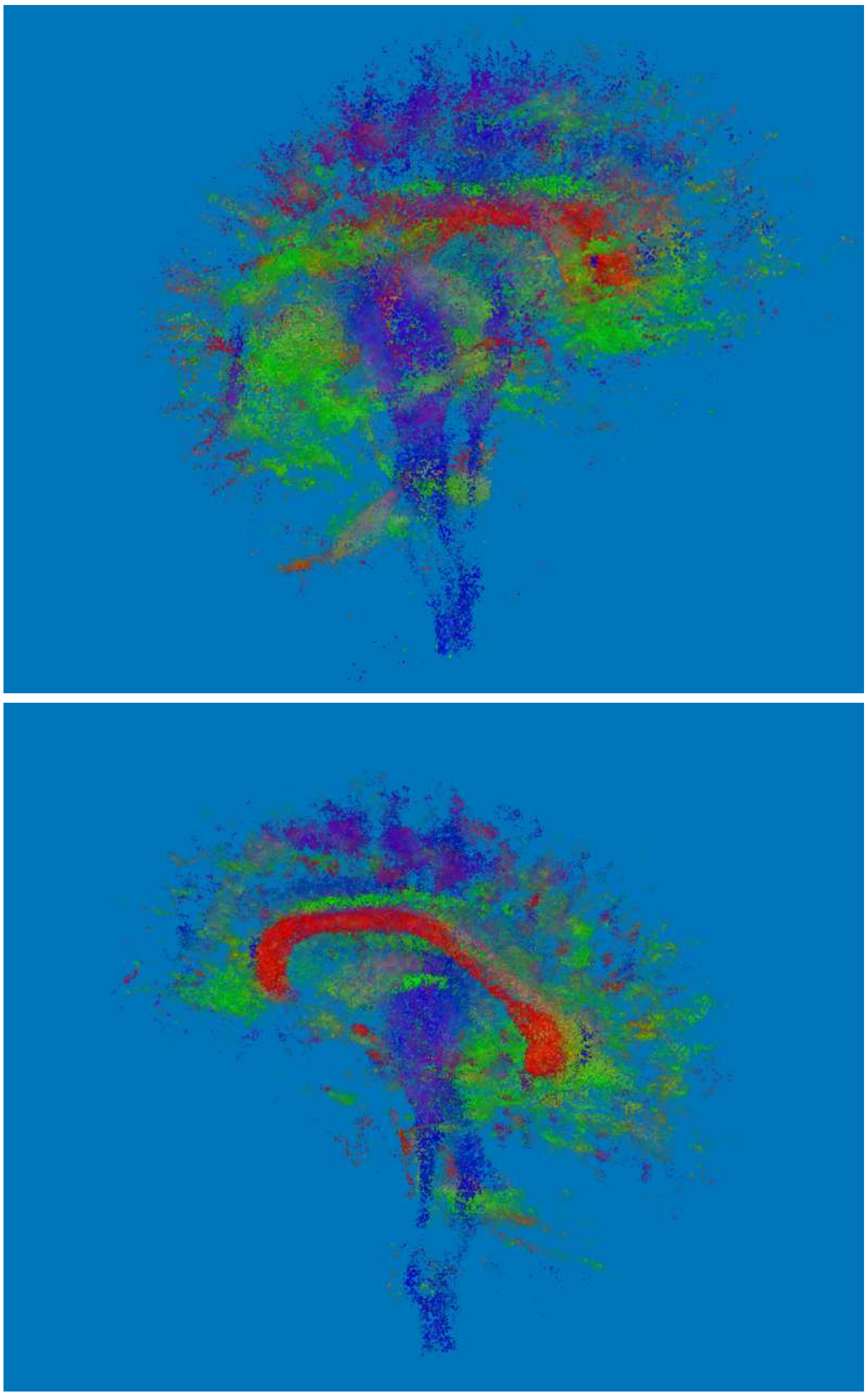


Figure 7.11: *Visualization of the diffusion tensor field measured in a human brain: (**a**) $\sigma_{\mathrm{iso\_min}} = 0.05$, (**b**) $\sigma_{\mathrm{iso\_min}} = 0.2$. Below a given anisotropy threshold $\sigma_{\mathrm{iso\_min}}$ the transparency of rendered particle primitives is inversely proportional to the measured anisotropy. In this way, particles in regions of low anisotropy are continuously faded out. 64K particles of size 5×5 pixels were rendered at roughly 40 frames per second.*

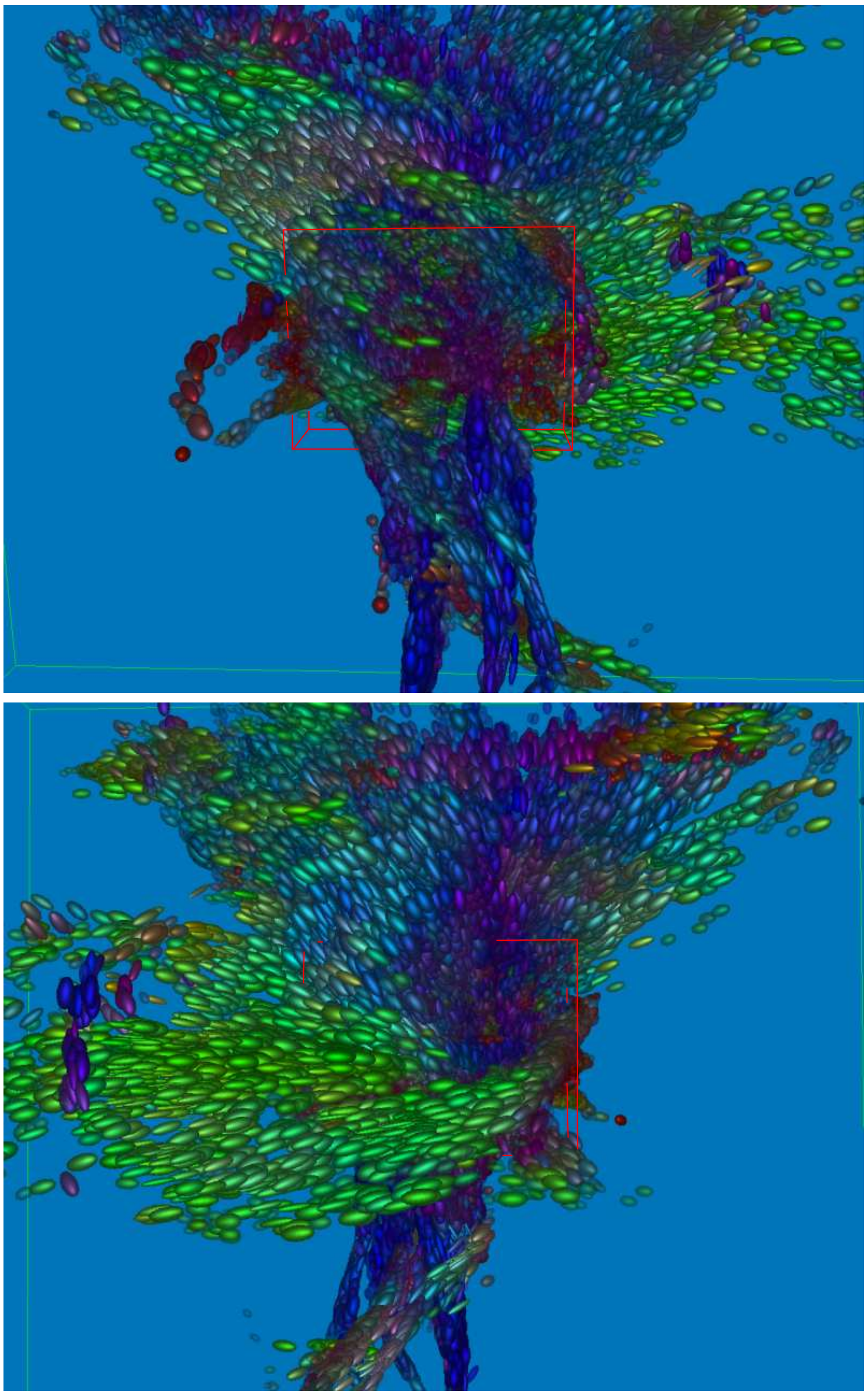
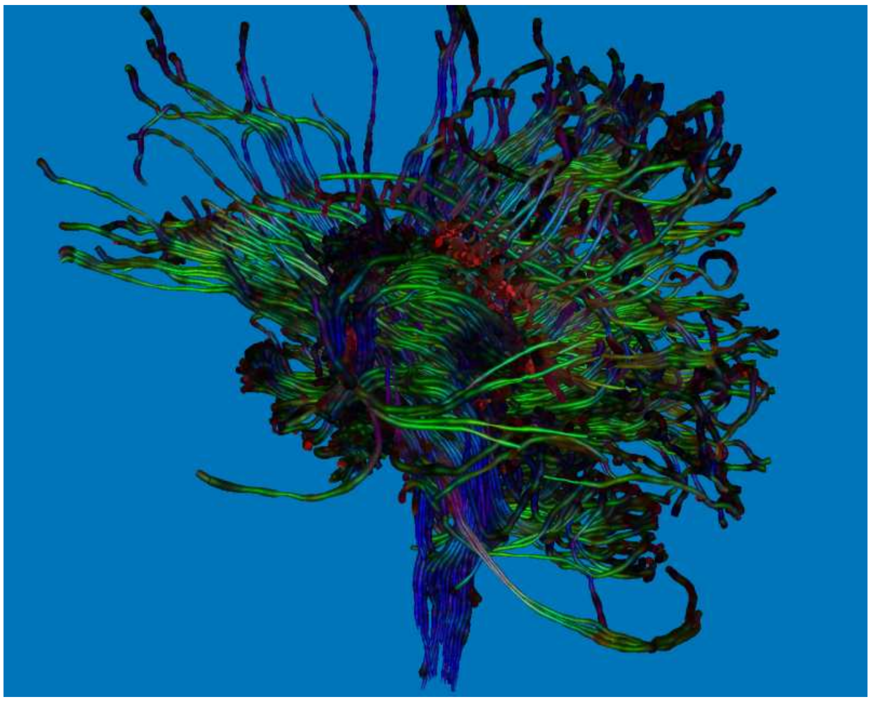


Figure 7.12: *Brain structures: By using the color mapping scheme described in Equation 7.6, the optic tract (green) and corona radiata (blue) fibers can be clearly distinguished. At 6.1 frames per second, 64K particles were advected and rendered using textured sprites of size* 45×45 *pixels.*

(**a**)



(**b**)

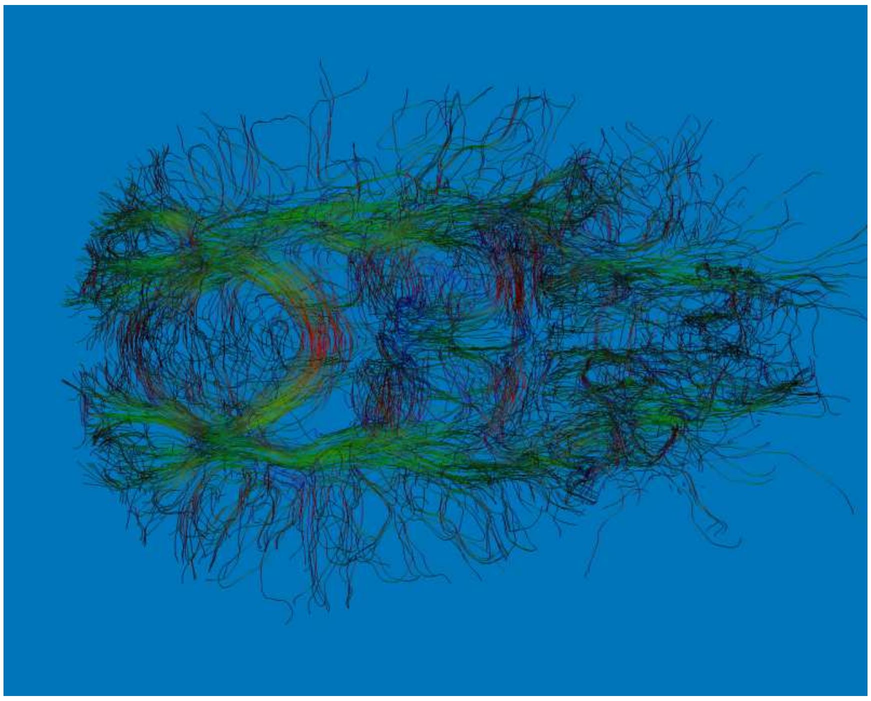
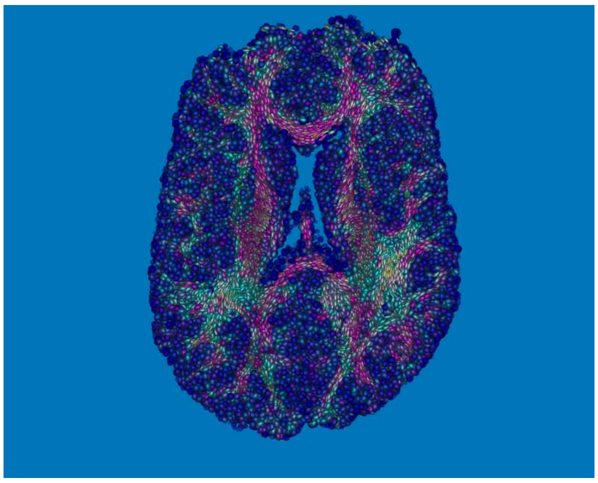
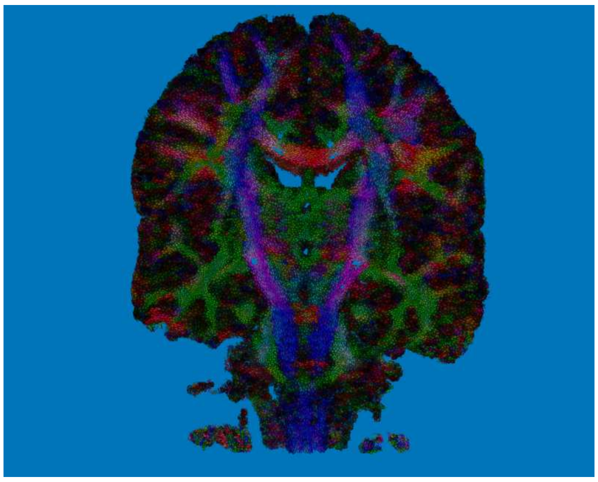


Figure 7.13: *Brain structures: (**a**) corona radiata is visualized using stream tubes. (**b**) the axial slice along the longitudinal fasciculus is rendered by means of simple stream lines. The FA-based color mapping scheme (Equation 7.6) is applied. 512 fibers with a maximum length of 600 were generated and rendered at 2.4 and 10 frames per second, respectively.*

(**a**)



(**b**)



(**c**)

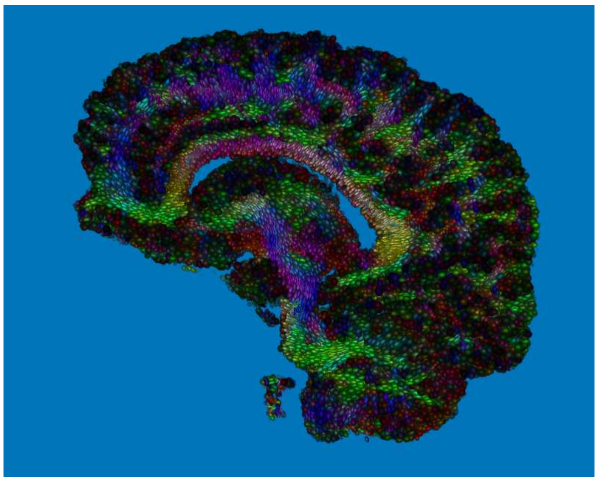
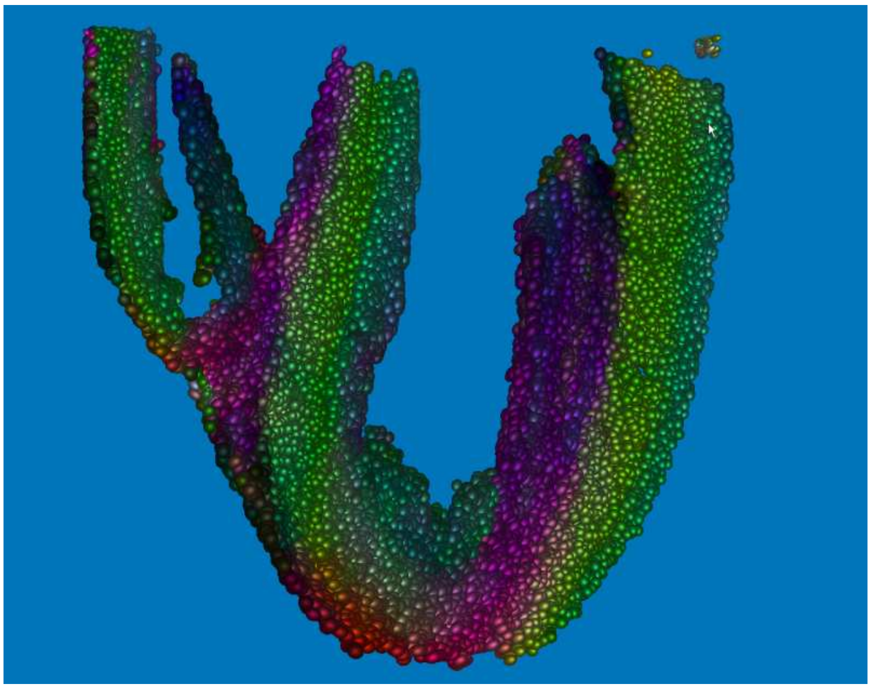


Figure 7.14: *Brain structures:* (**a**) *axial,* (**b**) *coronal, and* (**c**) *sagittal slices through the data set. The coronal and sagittal slices are color coded using Equation 7.6, and Equation 7.5 is used to color the axial slice. The axial and sagittal slices were rendered using 64K particles of size* 20×20 *pixels at 10.8 frames per second. The image of the coronal slice was generated at 4.5 frames per second using 256K particles of size* 10×10 *pixels.*

(**a**)



(**b**)

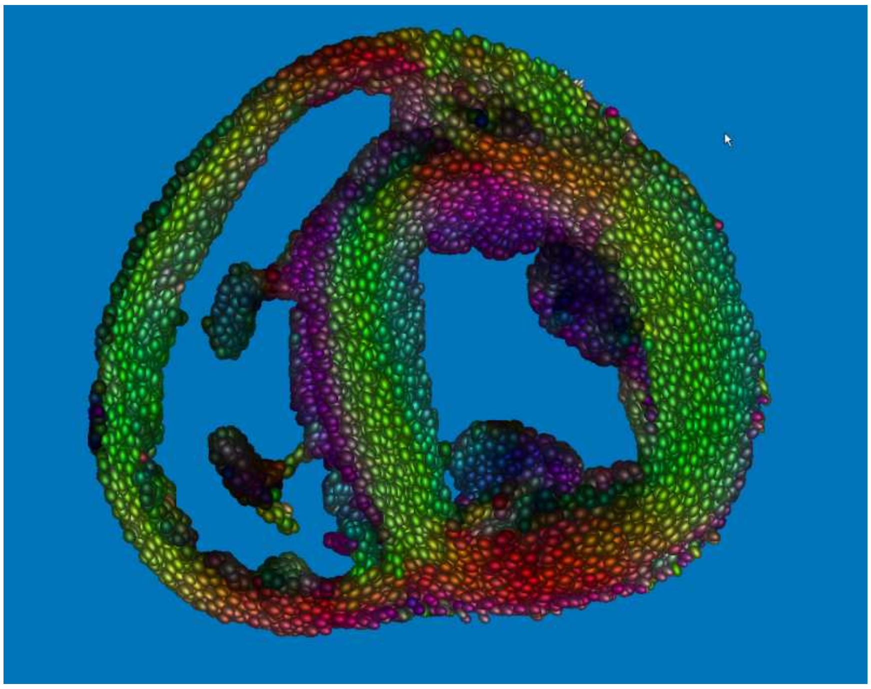


Figure 7.15: *Longitudinal (**a**) and latitudinal (**b**) slices of the heart muscle are reconstructed and rendered at 8 frames per second. The helical orientation of the heart muscle fiber becomes apparent from the latitudinal slice. Both slices also depict the four-chambered structure of the heart. The FA-based color mapping scheme was chosen.*

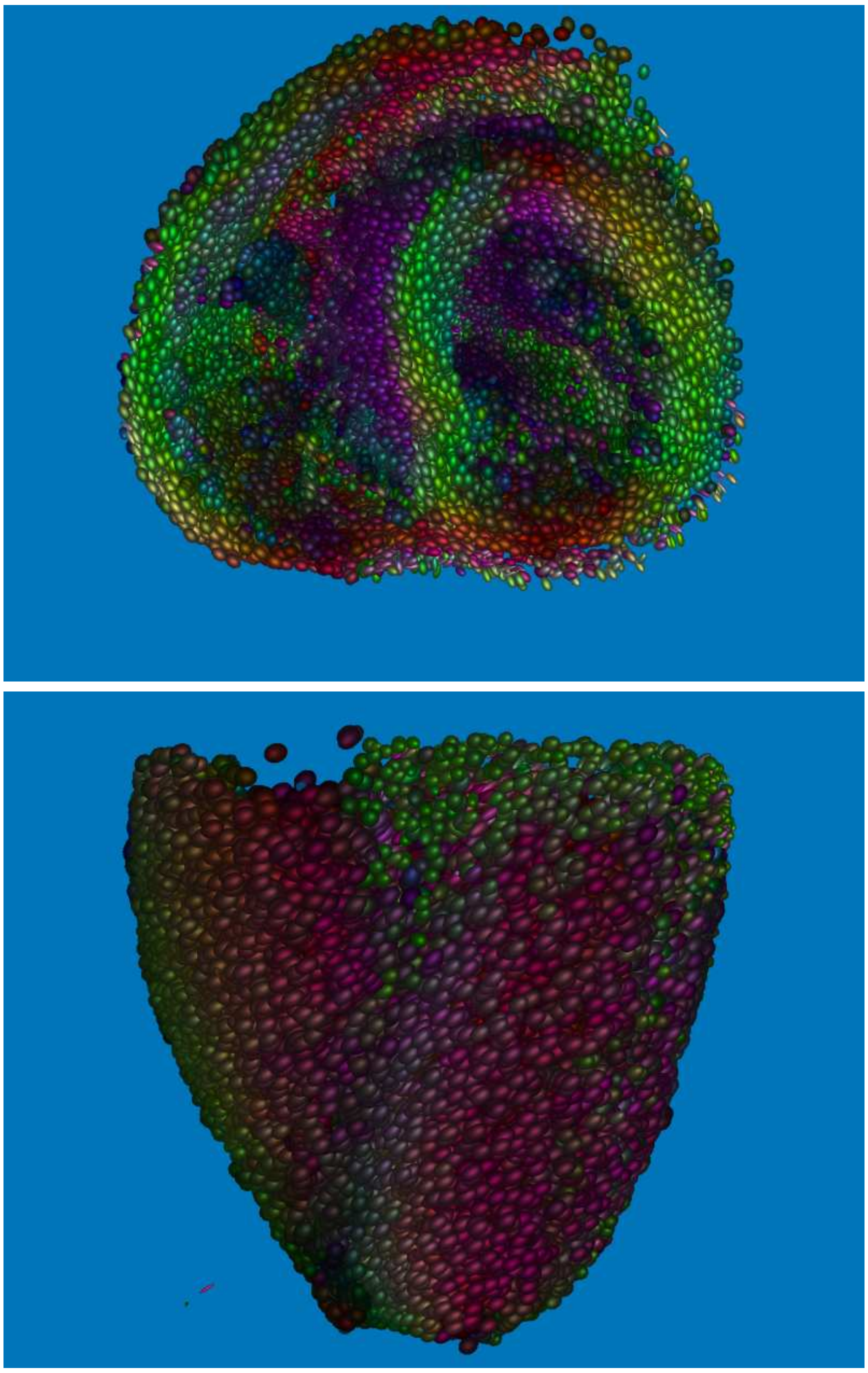


Figure 7.16: *Heart muscle: Two different views of the data set are shown. The helical structure of the heart muscle can be easily tracked on both images. The FA-based color mapping scheme (Equation 7.6) was applied. 64K particles of size 25×25 pixels (left) and 45×45 pixels (right) were animated at 9.4 and 7.4 frames per second, respectively.*

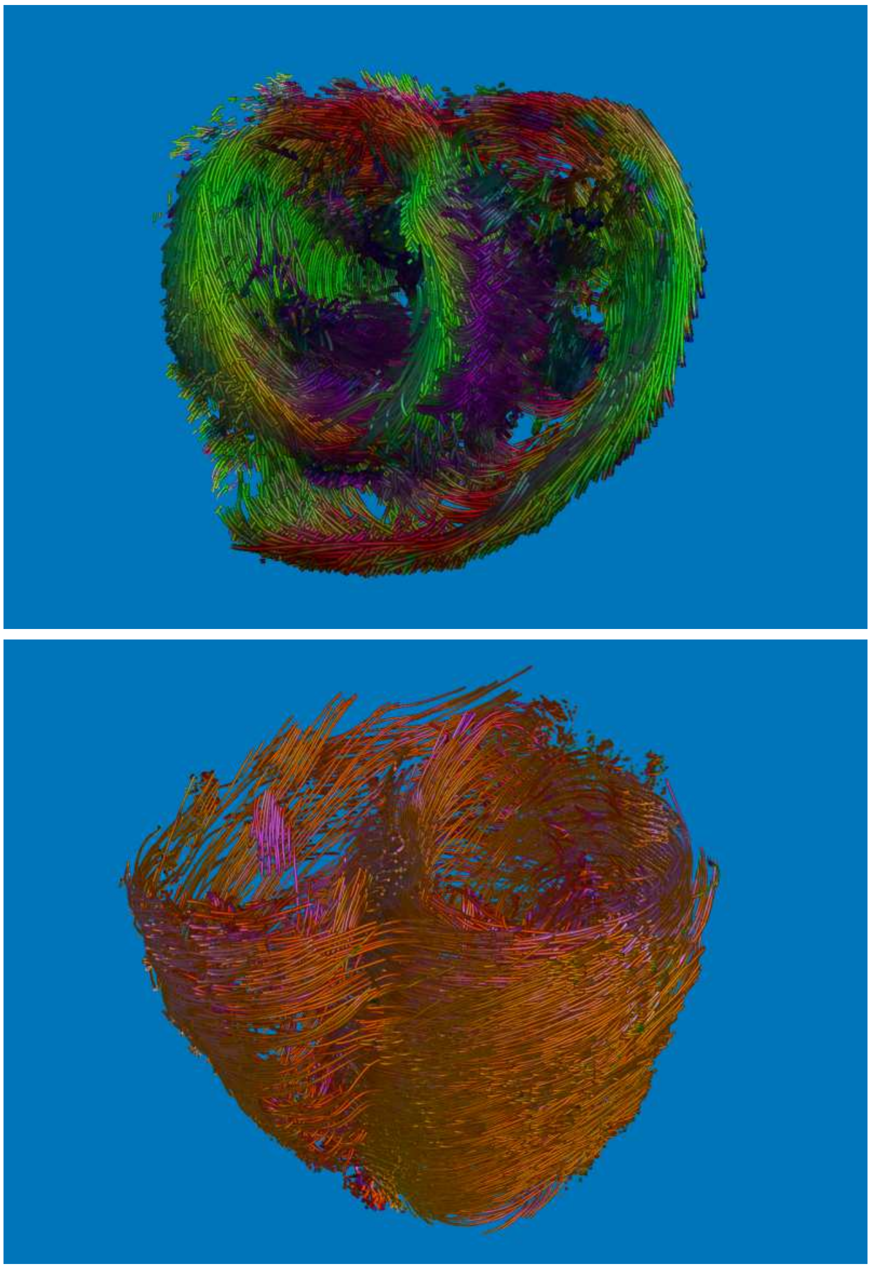


Figure 7.17: *Heart muscle: Two different views of the data set are shown. The color mapping schemes described in Equation 7.6 (left) and Equation 7.4 (right) were used. 1K stream tubes of maximum length of 1K were reconstructed and rendered with point sprites of size* 5×5 *pixels. Both images were generated at 1.5 frames per second.*

Chapter 8

Conclusion and Future Work

In this thesis, several novel methods have been introduced within the context of *flow exploration*, a challenging interdisciplinary area of research, encompassing a wide range of versatile topics. Despite the diversity of these topics, they can be roughly classified into three main groups: flow measurement, reconstruction of flow fields from measured data, and visualization of the results of reconstruction. All of these problems occur in a particular flow situation to be explored and in the ideal case must provide a thorough understanding of the flow phenomena under investigation. This thesis is devoted to the reconstruction of flow fields and their visualization. In this short chapter, I will briefly summarize what has been done in this regard and outline the directions for further research.

8.1 Conclusion

Since *all* of the proposed algorithms are dedicated to fluid flow, it is natural that all of them are built upon one paradigm, which intuitively describes the flow per se: *the particle tracing*. The efficiency of this concept has been proven by a number of papers published by our Chair [Cha] on renown conferences and two international visualization contests [SKKW05, SKB$^+$06] which we have won. The diversity of topics covered in these publications is ranging from numerical simulation of flow fields and visualization of earthquakes, to completely different areas related to rendering of special effects for computer games. In the following, I will briefly summarize my contribution to the *scientific flow exploration* represented by several modules described in this thesis and developed within the framework of the GPU-based particle engine. It is worth mentioning that together with these modules the particle engine has evolved into a *comprehensive tool for interactive flow exploration*.

8.1.1 Reconstruction Algorithm and its Validation

The reconstruction of velocity vector fields from experimental image sequences is a non-trivial problem, especially when a number of restrictions is imposed on the experimental setup in order to satisfy the application-specific requirements. In the first part of this thesis (Chapters 3 and 4), a new model-based prediction-correction technique for the reconstruction of fluid motion from captured images of particles seeded into the flow has been described. Since this technique incorporates a priori knowledge about the induced flows into the reconstruction process, it is able to recover relevant flow features at high fidelity. In contrast to previous approaches, flow boundaries and obstacles in the flow are considered in the correction step.

It have been shown, that exploiting a GPU simulation engine to efficiently predict the flow field as well as to compute the model-based correction of this field, interactive, yet high quality reconstruction is possible. As timings indicate, the proposed technique can be seen as a first step towards the integration of the reconstruction process into real-time scenarios such as high-speed PIV-systems. Furthermore, interactive visual exploration of the reconstructed flow field provides the expert with direct feedback of the changes imposed by the variation of specific model parameters.

By means of the presented technique one additional step towards the use of physical knowledge in a mathematically and numerically sound way for the evaluation of PIV image sequences have been made. Moreover, another important aspect, which is playing an ever increasing role in experimental procedures, has been emphasized: the possibility to gain insight by interactively steering all the specific process parameters involved in the experiment.

Furthermore, a thorough validation of the proposed reconstruction method has been performed and its performance has been compared in terms of quality of the reconstructed vector field and computational time to other algorithms. In particular, the efficiency of this reconstruction method has been shown in application to experimental image pairs captured by biocompatible microscopic PIV-systems. Such systems are characterized by limited illumination power and particle density, and they make it difficult for standard evaluation methods to faithfully reconstruct the fluid motion.

8.1.2 Interactive Visualization of Flow Fields

In the second part of the thesis, two novel approaches for the interactive visualization of flow fields have been presented. Moreover, a result of collaborative research in our Chair – a GPU-based particle engine – is briefly summarized in Chapter 5 in order to provide a high-level description of the implementation foundation for the two novel visualization methods described in Chapters 6 and 7.

Anchor Lines: Feature-Based Importance-Driven Technique for Flow Visualization

In order to solve the problem of an adequate perception of complex flows in 3D space arising in their visual exploration, a new focus strategy for particle tracing has been introduced in Chapter 6. The proposed technique is called anchor lines, which are essentially the integral curves started in regions of high importance and accompanied by the particles seeded in the close vicinity of these regions.

It is a challenging task to decide which regions in the flow are of interest and many different importance metrics can be suggested. After all, the choice may be based solely on a personal opinion of the user or be strictly application-specific. In this thesis, it has been proven that one of the best criteria to guide the classification is the degree of particle separation over time, which is described by the finite-time Lyapunov Exponent (FTLE). In particular, it has been demonstrated that seeding the particles in regions of high degree of particle separation indicated by high FTLE values yields a rich spectrum of various trajectories traced by the accompanied particles.

In Section 6.3.2, it has been shown that rendering only the particles which significantly deviate from their anchors and otherwise displaying solely the anchors, leads to a significant reduction of information, yet highlighting the important flow structures. The effectiveness of the proposed technique has been exhibited by the examples of visual exploration of several complex flow scenarios in 3D space. It is worth mentioning that the benefits of using this technique for real-time flow field visualization are more apparent in the animation and interaction with the developed software application.

Finally, working with the associated software tool, the user can control the visualization by a few parameters such as the size and location of a seeding probe; the threshold of importance related to the minimum degree of particle separation; the radius of particles scattering around the starting position of the anchor line; the size, shape and transparency of particles traced through the flow, as well as the number and type of various rendering modes to be displayed at once, etc. In addition, the FTLE measure is directly derived from the flow and is employed to adaptively modify the particles' visual attributes.

Visualization of Diffusion Tensor Fields

The visualization of tensor fields arising in many areas of natural sciences is a highly important and relatively poorly investigated topic. In Chapter 7, an efficient and effective visualization system for 3D diffusion tensor fields has been presented. This system allows for visual exploration of such fields at interactive rates and at arbitrary level of detail. The user can interactively select regions of interest by positioning a particle probe of adjustable position and size. A number of different visualization options and physically-based parameters can be selected to change the medical interpretation and visual appearance of the displayed information.

In this way, by using the proposed system, local variations in diffusion tensor fields as well as anatomical structures can be visualized at interactive rates. Due to the possibility to simulate the dynamic behavior of massless particles in the derived diffusion field, a very intuitive approach for understanding of diffusion tensor fields has been presented. In contrast to previous methods, real-time advection and rendering of large particle sets produces animations that closely and intuitively mimic the underlying dynamic diffusion process.

8.2 Future Work

Taking into account the versatility of the problems which have been investigated in this thesis, there is a wide range of directions for further research. These directions are related to both theoretical aspects and implementation issues.

8.2.1 Reconstruction Algorithm

The proposed reconstruction algorithm can be extended in several ways. Since the efficiency of the method has been proven for the 2D case, a strong motivation for the re-implementation of this algorithm for the 3D case is provided. Taking into account that all of the real-world experiments are essentially performed in 3D space, it could be of interest to compare the results of the reconstruction from 2D images performed under the assumption that sought for vector field approximately represents a 2D slice of a complete flow situation and the results of the reconstruction from complete 3D measurements of the same flow scenario.

The extension to 3D requires corresponding re-implementation and rearrangement of all the involved numerical procedures. While some of them, for instance, image deformation by a given vector field in 3D, can be implemented straightforward, other procedures involve a lot of work in order to make them computationally efficient. Moreover, the size and complexity of the associated mathematical problems will dramatically increase. This could lead to severe problems related to numerical accuracy and considerably decrease the stability of the involved numerical solvers. Thus, a thorough theoretical analysis of the related stability problems and the development or adaptation of alternative numerical procedures could be required. For instance, in computational fluid dynamics it is a well-known fact, that standard first-order finite element discretizations may result in non-physical pressure oscillations or even in so-called locking effects, where the zero velocity field is the only one satisfying the incompressibility condition. To cope with this problem, mixed finite elements, for example, Taylor-Hood element based on a square reference element with nine nodes can be employed while solving the Navier-Stokes equations [RS07]. Better results in terms of numerical convergence of optical flow solution can be achieved using, e.g., a mimetic finite differencing scheme [YRMS05].

Furthermore, from a theoretical point of view, it is of interest to incorporate alternative physical flow models into the system, in particular, numerical solvers to simulate such models. In this way, the proposed techniques can also be adjusted for scenarios where the flow does not comply with the incompressible viscous flow model. On the other hand, the theoretical research can also be pursued further into the direction of a more accurate model for flows induced by microorganisms. In this sense, the following issues could be of particular interest. The flow model based on Taylor's hypothesis can be incorporated as suggested in [PKD$^+$07] and compared to the currently implemented Stokes model. Another theoretically involved challenging task is the treatment of boundary conditions imposed by the presence of non-solid living organisms in biological application. Moreover, since feeding/discharging cannot be treated using a classical law of energy conservation, related inflow and outflow conditions should be carefully adjusted.

In addition, research could also be directed towards the investigation of the behavior of living organisms and the dependency of exchange characteristics of the induced flow on the presence of external factors. Moreover, the optimality of experimental parameters can be investigated: which time interval between the recordings is sufficient in order to reliably reconstruct the underlying flow field, which size of particles should be used in order to avoid the alternation of the flow pattern, which optical density should particles have in 2D and 3D cases, etc. Yet another challenging and difficult task is related to the investigation of the process of formation of a toroidal eddy and the determination of the critical physical conditions when the organism decides to break the established flow pattern, for example by sudden and very fast contraction of its body.

8.2.2 Anchor Lines

Several directions can be suggested for further research about the anchor line technique presented in this thesis. Since many numerical simulations are performed on unstructured grids, one important extension is related to the re-implementation of the proposed technique for data sets defined on such grids. As none of the procedures involved in the computation of anchor lines inherently depends on the uniform grid structure, this extension is expected to be straightforward, at least from an algorithmic point of view. However, the realization of real-time approaches for GPU-based particle tracing in unstructured grids leaves sufficient room for further research.

Another interesting issue that can be addressed is related to the investigation of topological features as a candidate for importance measure and comparison of its efficiency to the FTLE-based importance measure. I believe that especially the combination of topological features with anchor lines to highlight coherent structures in the flow is an interesting visualization option.

It is worth mentioning that in addition to the FTLE, there also exists a finite-size Lyapunov exponent measure that characterizes how long it takes for particles until they separate to a fixed distance. Such information can be useful to predict the development of a specific flow pattern and even to guide it. The investigations in this direction can be especially valuable for turbulent flow scenarios.

A challenging starting point for further research is related to the investigation of areas of application of anchor lines. For instance, it could be of interest to investigate the efficiency of using this technique for visualization of the results of reconstruction of image sequences obtained in experiments with microorganisms. The behavior of organisms directly affects the flow pattern and, as a consequence, transport and mixing characteristics in the environment surrounding them . Therefore, in such flow scenarios anchor lines could be adjusted for the exploration of transport and mixing characteristics in biological systems.

Taking into account that the FTLE quantity defines the amount of stretching about the trajectory, it could be employed for the visualization of deformations of fluid elements. More suitable rendering modes for anchor lines and accompanying particles can be invented for this purpose. Since deformation is directly related to the deformation tensor, this idea could be elaborated further as a new paradigm for visualization of multivariate fields, in particular, velocity vector fields combined with rate-of-strain tensor fields, which are especially valuable in fluid mechanics application.

Finally, the idea to employ anchor lines in combination with focus and context techniques for the efficient visualization of uncertainty sounds appealing. By simply replacing focus by certainty and context by uncertainty the proposed techniques can be used to distinguish between regions containing reliable and non-reliable information. The questions how to find meaningful seeding positions for anchor lines and how to adjust the accompanying particles for uncertainty visualization suggest the way for further research in this area.

8.2.3 Visualization of Tensor Fields

In regard to the visualization of diffusion tensor fields, there are several directions for improvements and further research. Firstly, in the current implementation, the overall anisotropy measure is used to define the shape of the diffusion ellipsoid. Such treatment does not allow to distinguish between regions of linear and planar anisotropy, which is in many cases unsatisfactory. Therefore, the sprite atlas for diffusion tensor fields can be extended about an additional dimension incorporating the information about the third principal direction of anisotropy.

An additional problem related to the regions of planar isotropy is related to the fact that these regions essentially represent nodes, where fiber tracks are crossing. Since such nodes play a crucial role in the investigation of the connectivity between different regions within biological tissue, tracing a single line in such region is not always sufficient [WLW00]. Furthermore, the second-order diffusion tensor model fails to recover the correct directions of anisotropy in regions, where many fibers are crossing. To eliminate this limitation, a new method for high quality measurement of diffusion in biological tissue – high angular-resolution diffusion imaging (HARDI) – has been recently proposed [TWBW99]. As a consequence, new visualization techniques are awaited. Since HARDI data is described by a higher-order tensor model, rather than by a simple 3×3 second-order tensor model, it is a challenging task to visualize such data.

Second-order tensor fields are fundamental in engineering and physical sciences. In fluid flows, stresses, viscous stresses, rate of strain, turbulent charge, turbulent current, and momentum transfers are all described in terms of tensor data. In fact, the steady-state Navier-Stokes equations describe gas flows with only one quantity – momentum flux density – which is itself a tensor field. Visualization methods using hyperstreamlines were suggested to visualize such fields [DH92]. However, hyperstreamlines in many cases reveal only the mathematical properties of the tensor fields and do not provide intuitive physical interpretation [HFH+04]. It is worth mentioning, that the examination of tensor fields arising in fluid mechanics is relatively new and the physically sound visualizations of tensor quantities in well-known fluid mechanics problems have not been presented so far [KKL04]. Therefore, the development of intuitive visualization methods that enable physical understanding of tensor fields arising in fluid mechanics is a challenging direction for further research.

It is worth mentioning that visualization methods for tensor fields arising in one particular application are not always suited for other applications [HS]. For example, it is quite intuitive to trace particles along the main principal direction of the tensor in diffusion tensor fields, because it mimics the averaged motion of water molecules along the fiber tracks. On the other hand, tracing particles along the principal directions of the velocity gradient tensor is misleading. Furthermore, tensors with the same physical name are treated differently in different fields of natural sciences. For example, strain and stress tensors are present in solid mechanics and fluid mechanics, however their physical interpretation and treatment in these disciplines is different.

Finally, visualization techniques for tensor fields presented so far address symmetric second-order tensors in both 2D and 3D. The methods for asymmetric and mixed tensors as well as for higher-order tensors are still absent. More precisely, there are only a few research papers on this matter [HS, ZYLL07], and due to the complexity of the underlying problem they deal only with 2D tensors or they do not provide a physical interpretation of the visualizations. Therefore, visualization of tensor fields of complex types (asymmetric and mixed, or of higher-order) encountered in different areas of natural sciences is still an unexplored area of research.

Bibliography

[AB78] K. Aderogba and J. R. Blake, *Addendum to: Action of a force near the planar surface between two semi-infinite immiscible liquids at very low reynolds numbers*, Bulletin of Australian Mathematical Society **19** (1978), 309–318.

[AB98] A. K. Afifi and R. A. Bergman, *Functional neuroanatomy*, McGraw-Hill, 1998.

[Adr84] R. J. Adrian, *Scattering particle characteristics and their effect on pulsed laser measurements of fluid flow: speckle velocimetry vs. particle image velocimetry*, Applied Optics **23** (1984), 1690–1691.

[Adr91] R.J. Adrian, *Particle-imaging techniques for experimental fluid mechanics*, Annual Review of Fluid Mechanics **23** (1991), 261–304.

[Adr97] ______, *Dynamic ranges of velocity and spatial resolution of particle image velocimetry*, Measurement Science and Technology **8** (1997), no. 12, 1393–1398.

[Adr05] ______, *Twenty years of particle image velocimetry*, Experiments in Fluids **39** (2005), 159–169.

[Arn92] V. I. Arnold, *Ordinary differential equations*, Springer-Verlag, Berlin, 1992.

[AS87] M. S. Acarlar and C. R. Smith, *A study of hairpin vortices in a laminar boundary layer. part2. hairpin vortices generated by fluid ejection*, Journal of Fluid Mechanics **175** (1987), 43–83.

[BA83] J. Burt and E. H. Adelson, *The Laplacian pyramid as a compact image code*, IEEE Transactions on communication **COM-31** (1983), no. 4, 532–540.

[BAHH92] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani, *Hierarchical model-based motion estimation*, The 2nd European Conference on Computer Vision (Santa Margherita, Italy) (G. Sandini, ed.), LNCS, vol. 558, 1992, pp. 237–252.

[Bas97] P.J. Basser, *New histological and physiological stains derived from diffusion-tensor MR images*, Annals of the New York Academy of Sciences **820** (1997), 123–138.

[Bel60] R. E. Bellman, *Introduction to matrix analysis*, McGraw-Hill, New York, 1960.

[BFB94] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, *Performance of optical flow techniques*, International Journal of Computer Vision **12** (1994), no. 3, 43–77.

[BFH$^+$02] S. M. Bartram, G. A. Fleming, W. M. Humphreys, L. N. Jenkins, J. D. Jordan, J. W. Lee, B. D. Leighty, J. F. Meyers, and B. W. South, *Unified instrumentation: Examining the simultaneous application of advanced measurement techniques increased wind tunnel testing capability*, Aerodynamic Measurement Technology and Ground Testing Conference (St. Louis, Missouri, USA) (G. A. Fleming, ed.), vol. 22, American Institute of Aeronautics and Astronautics, June 2002.

[BG88] J. Bigün and G. H. Granlund, *Optical flow based on the inertia matrix of the frequency domain*, Proceedings from SSAB Symposium on Picture Processing (Lund University, Sweden), SSAB, March 1988.

[BHM00] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, 2nd edition ed., SIAM, Philadelphia, 2000.

[BK02] J. Barron and R. Klette, *Quantitative color optical flow*, The 16th International Conference on Pattern Recognition, vol. 4, 2002, pp. 251–255.

[BKHJ01] R. Bruckschen, F. Kuester, B. Hamann, and K. I. Joy, *Real-time out-of-core visualization of particle traces*, IEEE Symposium on parallel and large-data visualization and graphics, 2001, pp. 45–50.

[BKKW08] K. Bürger, P. Kondratieva, J. Krüger, and R. Westermann, *Importance-driven particle techniques for flow visualization*, IEEE VGTC Pacific Visualization Symposium, 2008.

[BL91] S. Bryson and C. Levit, *The virtual windtunnel: an environment for the exploration of three-dimensional unsteady flows*, IEEE Visualization, 1991, pp. 17–24.

[BML94] P.J. Basser, J. Mattiello, and D. LeBihan, *MR diffusion tensor spectroscopy and imaging*, Biophysical Journal (1994), 259–267.

[BO96] J.R. Blake and S. R. Otto, *Ciliary propulsion, chaotic filtration and a 'blinking' stokeslet*, Journal of Engineering Mathematics **30** (1996), 151–168.

[BOB98] J. R. Blake, S. R. Otto, and D. A. Blake, *Filter feeding, chaotic filtration and a blinking stokeslet*, Theoretical and Computational Fluid Dynamics **10** (1998), no. 1, 23–36.

[Bol99] J. Bolinder, *On the accuracy of a digital particle image velocimetry system*, Tech. report, Lund Institute of Technology, Lund, Sweden, 1999.

[BP02] L. Barreira and Y. Pesin, *Lyapunov exponents and smooth ergodic theory*, University Lecture Series (Providence, USA), vol. 23, American Mathematical Society, 2002.

[BSK$^+$07] K. Bürger, J. Schneider, P. Kondratieva, J. Krüger, and R. Westermann, *Interactive visual exploration of instationary 3D-flows*, Eurographics/IEEE VGTC Symposium on Visualization (EuroVis), 2007.

[BT68] A. I. Borisenko and I. E. Tarapov, *Vector and tensor analysis with applications*, Dover, New York, 1968.

[BT05] J. L. Barron and N. A. Thacker, *Tutorial: Computing 2D and 3D optical flow*, Tina Memo No. 2004-012, 2005.

[BWF$^+$05] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr, *Variational optical flow computation in real-time*, IEEE Transactions on Image Processing **14** (2005), no. 5, 608–615.

[CAA01] O. Coulon, D. Alexander, and S. R. Arridge, *A regularization scheme for diffusion tensor magnetic resonance images*, Lecture Notes in Computer Science **2082** (2001), 92–105.

[CE97] M. Cox and D. Ellsworth, *Application-controlled demand paging for out-of-core visualization*, IEEE Visualization, 1997, pp. 235–244.

[Cha] *Chair for Computer Graphics and Visualization, TU München, Germany*, http://wwwcg.in.tum.de/.

[CHA$^+$06] T. Corpetti, D. Heitz, G. Arroyo, É. Mémin, and A. Santa Cruz, *Fluid experimental flow estimation based on an optical-flow scheme*, Experiments in Fluids **40** (2006), no. 1, 80–97.

[CK05] J. Chen and J. Katz, *Elimination of peak-locking error in PIV analysis using the correlation mapping method*, Measurement Science and Technology **16** (2005), no. 8, 1605–1618.

[CMP02] T. Corpetti, E. Mémin, and P. Pérez, *Dense estimation of fluid flows*, IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002), no. 3, 365–380.

[Cra75] J. Crank, *The mathematics of diffusion*, Oxford University Press, Oxford, England, 1975.

[CW05] J. Carlier and B. Wieneke, *Report 1 on production and diffusion of fluid mechanics images and data, Fluid project deliverable 1.2*, European Project "Fluid image analysis and description" (FLUID) – http://www.fluid.irisa.fr/, 2005.

[DH92] Thierry Delmarcelle and Lambertus Hesselink, *Visualization of second order tensor fields and matrix data*, IEEE Visualization, 1992, pp. 316–323.

[DH02] H. Doleisch and H. Hauser, *Smooth brushing for focus+context visualization of simulation data in 3D*, The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media 2002 (WSCG 2002), 2002, pp. 147–154.

[DX9] *Reference for Direct3D 9*, http://msdn2.microsoft.com/.

[Dyk82] M. V. Dyke, *An album of fluid motion*, Parabolic Press, 1982.

[DZDL01] M. DaSilva, S. Zhang, C. Demiralp, and D. H. Laidlaw, *Visualizing diffusion tensor volume differences*, Visualization '01 Work in Progress Proceedings, October 2001, pp. 16–17.

[FD00] A. Fincham and G. Delerce, *Advanced optimization of correlation imaging velocimetry algorithms*, Experiments in Fluids **29** (2000), no. 7, 13–22.

[FG98] A. L. Fuhrmann and E. Gröller, *Real-time techniques for 3D flow visualization*, IEEE Visualization (D. Ebert, H. Hagen, and H. Rushmeier, eds.), 1998, pp. 305–312.

[FL03] J. Fried and H. Lemmer, *On the dynamics and function of ciliates in sequencing batch biofilm reactors*, Water Science and Technology **47** (2003), no. 5, 189–196.

[FP02] J. H. Ferziger and M. Peric, *Computational methods for fluid dynamics*, 3d edition ed., ch. Basic Concepts of Fluid Flow, pp. 1–2, Springer-Verlag, Berlin, 2002.

[FS97] A. M. Fincham and G. R. Spedding, *Low cost, high resolution DPIV for measurement of turbulent fluid flow*, Experiments in Fluids **23** (1997), no. 6, 449–462.

[FT32] A. Fage and H. C. H. Townend, *Turbulent motion near a wall*, Royal Society of London, A, vol. 135, 1932, pp. 656–677.

[FWT05] O. Frederich, E. Wassen, and F. Thiele, *Flow simulation around a finite cylinder on massively parallel computer architecture*, International Conference on Parallel Computational Fluid Dynamics, 2005, pp. 85–93.

[GDC00] R. B. Green, C. J. Doolan, and R. M. Cannon, *Measurements of the orthogonal blade-vortex interaction using a particle image velocimetry technique*, Experiments in Fluids **29** (2000), no. 4, 369–379.

[GDN98] M. Griebel, T. Dornseifer, and T. Neunhoeffer, *Numerical simulation in fluid dynamics: a practical introduction*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.

[GFP$^+$02] C. Gossl, L. Fahrmeir, B. Putz, L.M. Auer, and D.P. Auer, *Fiber tracking from DTI using linear state space models: Detectability of the pyramidal tract*, NeuroImage, vol. 16, 2002, pp. 378–388.

[GJ03] R. Gilbert and D. A. Johnson, *Evaluation of FFT-based cross-correlation algorithms for PIV in a periodic grooved channel*, Experiments in Fluids **34** (2003), no. 4, 473–483.

[GLS01] L. Gui, J. Longo, and F. Stern, *Biases of PIV measurement of turbulent flow and the masked correlation-based interrogation algorithm*, Experiments in Fluids **30** (2001), no. 1, 27–35.

[GMN$^+$98] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills, *Recovering motion fields: An evaluation of eight optical flow algorithms*, British Machine Vision Conference (D. Ebert, H. Hagen, and H. Rushmeier, eds.), vol. 1, 1998, pp. 195–204.

[Gol98] D. Goldstein, *Inverted microscopes*, Micscape (1998), on-line monthly magazine of the Microscopy UK web site.

[GP96] S. Gupta and J. Prince, *Stochastic models for div-curl optical flow methods*, IEEE Signal Processing Letters **3** (1996), no. 2, 32–35.

[GP97] I. Grant and X. Pan, *The use of neural techniques in PIV and PTV*, Measurement Science and Technology **8** (1997), 1399–1405.

[Gra03] K. Gray, *Microsoft DirectX 9 programmable graphics pipeline*, Microsoft Press, Redmond, WA, USA, 2003.

[GSAW05] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, *Random walks for interactive organ segmentation in two and three dimensions: Implementation and validation*, MICCAI (Palm Springs, Canada) (J. Duncan and G. Gerig, eds.), LNCS 3750, no. 2, MICCAI Society, Springer, October 2005, pp. 773–780.

[Hac94] W. Hackbusch, *Iterative solutions of large sparse systems of equations*, Springer-Verlag, New York, 1994.

[Hal88] N. A. Halliwell, *Progress in Particle Image Velocimetry*, Optics and Lasers in Engineering **9** (1988), 161–162.

[Hal01] G. Haller, *Distinguished material surfaces and coherent structures in three-dimensional fluid flows*, Physica D: Nonlinear Phenomena **149** (2001), no. 4, 248–277.

[Har00a] D. P. Hart, *PIV error correction*, Experiments in Fluids **29** (2000), no. 1, 13–22.

[Har00b] ______, *Super-resolution PIV by recursive local-correlation*, Journal of Visualization **3** (2000), no. 2, 187–194.

[HBPA01] K. M. Hasan, P. J. Basser, D. L. Parker, and A. L. Alexander, *Analytical computation of the eigenvalues and eigenvectors in DT–MRI*, Journal of Magnetic Resonance **152** (2001), no. 1, 41–47.

[HDG97] H. T. Huang, D. Dabiri, and M. Gharib, *On errors of digital particle image velocimetry*, Measurement Science and Technology **8** (1997), no. 12, 1427–1440.

[Hen00] W. J. Hendelman, *Atlas of functional neuroanatomy*, CRC Press, 2000.

[HFH$^+$04] I. Hotz, L. Feng, H. Hagen, B. Hamann, K. Joy, and B. Jeremic, *Physically based methods for tensor field visualization*, IEEE Visualization, 2004, pp. 123–130.

[HFW93] H. T. Huang, H. E. Fiedler, and J. J. Wang, *Limitation and improvement of PIV. Part II: particle image distortion, a novel technique*, Experiments in Fluids **15** (1993), no. 4, 263–273.

[HHN$^+$07] D. Heitz, P. Héas, V. Navaza, J. Carlier, and E. Mémin, *Spatio-temporal correlation-variational approach for robust optical flow estimation*, The 7th International Symposium on Particle Image Velocimetry, 2007.

[HJ04] C. D. Hansen and C. R. Johnson, *The visualization handbook*, Academic Press Inc., US, 2004.

[HLS] *HLSL: DirectX Software Development Kit: HLSL Shader Reference (Direct3D 9 HLSL)*, http://msdn.microsoft.com.

[HOP$^+$07] C. Hartmann, Ö. Özmutlu, H. Petermeier, J. Fried, and A. Delgado, *Analysis of the flow field induced by the sessile peritrichous ciliate Opercularia asymmetrica*, Journal Biomechanics **40** (2007), no. 1, 137–148.

[Hor89] P. Horowitz, *The art of electronics*, 2nd edition ed., ch. How to draw schematic diagrams, pp. 1056–1059, Cambridge University Press, 1989.

[HPvW94] L. Hesselink, F.H. Post, and J.J. van Wijk, *Scientific visualization: Advances and challenges*, ch. Research issues in vector and tensor field visualization, pp. 488–495, Academic Press, New York, 1994.

[HS] M. Hlawitschka and G. Scheuermann, *HOT-lines: tracking lines in higher order tensor fields*, IEEE Visualization, pp. 27–34.

[HS81] B. Horn and B. Schunck, *Determining optical flow*, Artificial Intelligence **17** (1981), 185–203.

[HWHJ99] B. Heckel, G. H. Weber, B. Hamann, and K. I. Joy, *Construction of vector field hierarchies*, IEEE Visualization, 1999, pp. 19–26.

[ILA] ILA, *Intelligent laser applications GmbH: Particle image velocimetry*, http://www.ila.de/.

[ISM05] T. Inanc, S.C. Shadden, and J.E. Marsden, *Optimal trajectory generation in ocean flows*, The 24th American Control Conference, 2005.

[JCT01] M.-C. Jullien, P. Castiglione, and P. Tabeling, *Intermittency of a passive tracer in the inverse energy cascade*, Physical Review **64** (2001), no. 3.

[JH95] J. Jeong and F. Hussain, *On the identification of a vortex*, Journal of Fluid Mechanics **285** (1995), 69–94.

[Jir03] T. Jirka, *Multidimensional data visualization*, Tech. report, University of West Bohemia, Pilsen, Czech Republic, 2003.

[JJDAF95] K. Jambunathan, X. Y. Ju, B. N. Dobbins, and S. Ashforth-Frost, *An improved cross correlation technique for particle image velocimetry*, Measurement Science and Technology **6** (1995), no. 5, 507–514.

[KA90] R. D. Keane and R. J. Adrian, *Optimization of particle image velocimeters. Part 1: double pulsed system*, Measurement Science and Technology **1** (1990), 1202–1215.

[KAZ95] R. D. Keane, R. J. Adrian, and Y. Zhang, *Super-resolution particle imaging velocimetry*, Measurement Science and Technology **6** (1995), no. 6, 754–768.

[KGW06] P. Kondratieva, J. Georgii, and R. Westermann, *Echtzeitverfahren zur modellbasierten rekonstruktion von strömungsfeldern aus experimentell bestimmten partikelsequenzen*, 14. GALA Fachtagung, Lasermethoden in der Strömungsmesstechnik, 2006.

[KGW$^+$08] P. Kondratieva, J. Georgii, R. Westermann, H. Petermeier, W. Kowalczyk, and A. Delgado, *A real-time model-based approach for the reconstruction of fluid flows induced by microorganisms*, Experiments in Fluids (2008), 25–+, Online First.

[Kin03] G. Kindlmann, *DTI visualization and analysis of diffusion tensor fields*, Ph.D. thesis, University of Utah, 2003.

[Kir04] T. Kirmse, *Image pattern correlation technique (IPCT). high precision optical measurement of surface shape and surface deformation*, Deutsches Zentrum für Luft- und Raumfahrt DLR, 2004.

[KKKW05] J. Krüger, P. Kiefer, P. Kondratieva, and R. Westermann, *A particle system for interactive visualization of 3D flows*, IEEE Transactions on Visualization and Computer Graphics **11** (2005), no. 6, 744–756.

[KKL04] M. Kirby, D. Keefe, and D. H. Laidlaw, *Visualization handbook*, ch. Painting and Visualization, Academic Press, 2004.

[KKW05] P. Kondratieva, J. Kruger, and R. Westermann, *The application of GPU particle tracing to diffusion tensor field visualization*, IEEE Visualization, 2005, pp. 73–78.

[KLRS04] A. Kolb, L. Latta, and C. Rezk-Salama, *Hardware-based simulation and collision detection for large particle systems*, IEEE Proceedings Eurographics Graphics Hardware Conference (T. Akenine-Möller and M. McCool, eds.), 2004, pp. 123–131.

[Kom07] J. Kompenhaus, *The 12th international symposium in flow visualization*, Journal of Visualization **10** (2007), no. 1, 123–128.

[KR03] E. M. Kalmoun and U. Rüde, *A variational multigrid for computing the optical flow*, Vision, Modeling, and Visualization Conference, 2003, pp. 577–584.

[KSE04] T. Klein, S. Stegmaier, and T. Ertl, *Hardware-accelerated Reconstruction of Polygonal Isosurface Representations on Unstructured Grids*, Pacific Graphics, 2004, pp. 186–195.

[KSK$^+$02] I. Kimura, Y. Susaki, R. Kiyohara, A. Kaga, and Y. Kuroe, *Gradient-based PIV using neural networks*, Journal of Visualization **5** (2002), no. 4.

[KSW04] P. Kipfer, M. Segal, and R. Westermann, *UberFlow: A GPU-based particle engine*, IEEE Eurographics Graphics Hardware Conference (T. Akenine-Möller and M. McCool, eds.), 2004, pp. 115–122.

[KW98] H. J. Kretschmann and W. Weinrich, *Neurofunctional systems. 3d reconstructions with correlated neuroimaging*, Thieme, New-York, 1998.

[KW99] G. Kindlmann and D. Weinstein, *Hue–balls and lit–tensors for direct volume rendering of diffusion tensor field*, IEEE Visualization, 1999, pp. 183–189.

[KW03] J. Krüger and R. Westermann, *Linear algebra operators for GPU implementation of numerical algorithms*, ACM Transactions on Graphics **22** (2003), no. 3, 908–916.

[KYKI98] A. Kaga, K. Yamaguchi, A. Kondo, and Y. Inoue, *Combination of PIV data with CFD using cost function method*, The 8th international symposium on flow visualization, 1998.

[KZD07] W. Kowalczyk, B. E. Zima, and A. Delgado, *A biological seeding particle approach for μ-PIV measurements of a fluid flow provoked by microorganisms*, Experiments in Fluids **43** (2007), no. 1, 147–150.

[Lab01] G. Labonté, *Neural network reconstruction of fluid flows from tracer-particle displacements*, Experiments in Fluids **30** (2001), no. 4, 399–409.

[LAK$^+$98] D. H. Laidlaw, E. T. Ahrens, D. Kremers, M. J. Avalos, C. Readhead, and R. E. Jacobs, *Visualizing diffusion tensor images of the mouse spinal cord*, IEEE Visualization, 1998, pp. 127–134.

[Lar08] R. S. Laramee, *Flow visualization, the state-of-the-art*, The 19th Conference of Simulation and Visualization, February 2008, Invited talk, pp. 127–134.

[LB81] N. Liron and J. R. Blake, *Existence of viscous eddies near boundaries*, Experiments in Fluids **107** (1981), 109–129.

[LBS03] G.-S. Li, U. Bordoloi, and H.-W. Shen, *Chameleon: An interactive texture-based rendering framework for visualizing three-dimensional vector fields*, IEEE Visualization, 2003, pp. 241 – 248.

[LCBS97] T. Liu, B. T. Campbell, S. P. Burns, and J. P. Sullivan, *Temperature- and pressure-sensitive luminescent paints in aerodynamics*, Applied Mechanics Reviews **50** (1997), no. 4, 227–246.

[LCM$^+$05] F. Lekien, C. Coulliette, A.J. Mariano, E.H. Ryan, L.K. Shay, G. Haller, and J.E. Marsden, *Pollution release tied to invariant manifolds: A case study for the coast of Florida*, Physica D: Nonlinear Phenomena **210** (2005), 1–20.

[LG98] H. Löffelmann and M. E. Gröller, *Enhancing the visualization of characteristic structures in dynamical systems*, The 9th Eurographics Workshop on Visualization in Scientific Computing, 1998, pp. 35–46.

[LHD$^+$04] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf, *The state of the art in flow visualization: Dense and texture-based techniques*, Computer Graphics Forum, vol. 23, 2004, pp. 203–221.

[LHZP05] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post, *Topology-based flow visualization, the state of the art*, Topology-Based Methods in Visualization, 2005, pp. 1–19.

[Lia66] A.M. Liapunov, *Stability of motion*, Academic Press, New York, 1966.

[Lig75] Sir J. Lighthill, *Mathematical biofluiddynamics*, ch. Mathematics of Aquatic Animal Locomotion at Low Reynolds Number, pp. 48–49, Society for Industrial and Applied Mathematics, Philadelphia, 1975.

[Lir82] N. Liron, *Ciliary fluid transport: theory and experiment*, Cell Motility and the Cytoskeleton **2** (1982), no. S1, 47–51.

[LJL03] D. F. Liang, C. B. Jiang, and Y. L. Li, *Cellular neural network to detect spurious vectors in PIV data*, Experiments in Fluids **34** (2003), no. 1, 52–62.

[LK81] B. D. Lucas and T. Kanade, *An iterative image registration technique with an application to stereo vision*, International Joint Conference on Artificial Intelligence, 1981, pp. 674–679.

[LK95] L. Lourenco and A. Krothapalli, *On the accuracy of velocity and vorticity measurements with PIV*, Experiments in Fluids **18** (1995), no. 6, 421–428.

[LK00] ______, *True resolution PIV: a mesh-free second-order accurate algorithm*, The 10th International Symposium on Applications of Laser Techniques in Fluid Mechanics, 2000, Paper 13.5.

[LL04] F. Lekien and N. Leonard, *Dynamically consistent lagrangian coherent structures*, The 8th Experimental Chaos Conference, vol. 742, American Institute of Physics, 2004, pp. 132–139.

[LM05] F. Lekien and J. E. Marsden, *Tricubic interpolation in three dimensions*, International Journal for Numerical Methods in Engineering **63** (2005), no. 3, 455–471.

[Löf98] H. Löffelmann, *Visualizing local properties and characteristic structures of dynamical systems*, Ph.D. thesis, Technischen Universität Wien, Vienna, Aurstria, 1998.

[LPW03] R. Lindken, C. Poelma, and J. Westerweel, *Compensation for spatial effects for non-uniform seeding in PIV interrogation by signal relocation*, The 5th International Symposium on Particle Image Velocimetry (Busan, Korea), September 2003, Paper 3302.

[LSM06] F. Lekien, S. C. Shadden, and J. E. Marsden, *Lagrangian coherent structures in n-dimensional systems*, Journal of Mathematical Physics **48** (2006), no. 6.

[LV84] W. Lauterborn and A. Vogel, *Modern optical techniques in fluid mechanics*, Annual Reviews in Fluid Mechanics **16** (1984), 223–244.

[MAA$^+$03] Y. Masutani, S. Aoki, O. Abe, N. Hayashi, and K. Otomo, *MR diffusion tensor imaging: recent advance and new techniques for diffusion tensor visualization*, European Journal of Radiology **1** (2003), no. 46, 53–66.

[May00] S. Mayer, *Numerical simulation of flow fields and particle trajectories in ciliary suspension feeding*, Bulletin of Mathematical Biology **62** (2000), no. 6, 1035–1059.

[MCCvZ99] S. Mori, B.J. Crain, V.P. Chacko, and P.C. van Zijl, *Three dimensional tracking of axonal projections in the brain by magnetic resonance imaging*, Annual Neurology **45** (1999), no. 2, 265–269.

[MDB87] B. H. McCormick, T. A. DeFanti, and M. D. Brown, *Visualization in scientific computing*, Computer Graphics **21** (1987), no. 6.

[Mer87] W. Merzkirch, *Flow visualization*, Academic Press, Orlando, Florida, 1987.

[Mod04] J. Modersitzki, *Numerical methods for image registration*, Oxford university press, New York, 2004.

[MP98] É. Mémin and P. Pérez, *A multigrid approach to hierarchical motion estimation*, International Conference on Computer Vision (Bombay, India), January 1998, pp. 933–938.

[MP99] É. Mémim and P. Pérez, *Fluid motion recovery by coupling dense and parametric vector fields*, The 7th IEEE International Conference on Computer Vision, September 1999.

[MSLJ00] M. Marxen, P. E. Sullivan, M. R. Loewen, and B. Jähne, *Comparison of Gaussian particle center estimators and the achievable measurement density for particle tracking velocimetry*, Experiments in Fluids **29** (2000), no. 2, 145–153.

[MT95] J. E. Marsden and A. J. Tromba, *Vector calculus*, 4th edition ed., W.H. Freeman & Company, New York, 1995.

[MTHG03] O. Mattausch, T. Theußl, H. Hauser, and M. E. Gröller, *Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines*, Proceedings of Spring Conference on Computer Graphics (K. Joy, ed.), SCCG, April 2003, pp. 213–222.

[Mur01] D. Murphy, *Fundamentals of light microscopy and digital imaging*, ch. Differential interference contrast (DIC) microscopy and modulation contrast microscopy, pp. 153–168, Wiley-Liss, New-York, 2001.

[NIN$^+$03] Y. Nakajima, H. Inomata, H. Nogawa, Y. Sato, S. Tamura, K. Okazaki, and S. Torii, *Physics-based flow estimation of fluids*, Pattern Recognition **36** (2003), no. 5, 1203–1212.

[NLRa] J. Nogueira, A. Lecuona, and P. A. Rodríguez, *Identification of a new source of peak locking, analysis and its removal in conventional and super-resolution PIV techniques*, Experiments in Fluids **30**, no. 3.

[NLR$^+$b] J. Nogueira, A. Lecuona, P. A. Rodríguez, J. A. Alfaro, and A. Acosta, *Limits on the resolution of correlation PIV iterative methods. practical implementation and design of weighting functions*, Experiments in Fluids **39**, no. 2.

[NLR97] J. Nogueira, A. Lecuona, and P. A. Rodríguez, *Data validation, false vectors correction and derived magnitudes calculation on PIV data*, Measurement Science and Technology **8** (1997), no. 12, 1493–1501.

[NLR01] ———, *Local field correction PIV, implemented by means of simple algorithms, and multigrid versions*, Measurement Science and Technology **12** (2001), 1911–1921.

[NOBT05] H. Nobach, N. T. Ouellette, E. Bodenschatz, and C. Tropea, *Full-field correlation-based image processing for PIV*, The 6th International Symposium on Particle Image Velocimetry, 2005.

[OLG$^+$07] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, *A survey of general-purpose computation on graphics hardware*, Computer Graphics Forum, vol. 26, 2007, pp. 80–113.

[ONI97] K. Ogawara, T. Noborizato, and S. Iida, *A moving least square PIV algorithm coupled with Navier-Stokes flow solver*, The 2nd international Workshop on PIV, 1997.

[ONSK00] K. Okamoto, S. Nishio, T. Saga, and T. Kobayashi, *Standard images for particle-image velocimetry*, Measurement Science and Technology **11** (2000), 685–691.

[OYB01] S. R. Otto, A. N. Yannacopoulos, and J. R. Blake, *Transport and mixing in Stokes flow: The effect of chaotic dynamics on the blinking stokeslet*, Journal of Fluid Mechanics **430** (2001), 1–26.

[OYN00] T. Okuno, S. Yasuhiko, and S. Nishio, *Image measurement of flow field using physics-based dynamic model*, Measurement Science and Technology **11** (2000), 667–676.

[PALO] A. K. Prasad, R. J. Adrian, C. C. Landreth, and P. W Offutt, *Effect of resolution on the speed and accuracy of particle image velocimetry interrogations*, Experiments in Fluids **13**, no. 2.

[PDK$^+$06] H. Petermeier, A. Delgado, P. Kondratieva, R. Westermann, F. Holtmann, V. Krishnamachari, and C. Denz, *A hybrid approach between experiment and evaluation for artefact detection and flow field reconstruction*, The 12th International Symposium on Flow Visualization, 2006.

[PG92] D. Pnueli and C. Gutfinger, *Fluid mechanics*, Cambridge Unviversity Press, New York, 1992.

[PH84] J. D. Pickering and N. A. Halliwell, *Speckle photography in fluid flows: signal recovery with two step processing*, Applied Optics **23** (1984), no. 8, 1128–1129.

[PKD$^+$07] H. Petermeier, W. Kowalczyk, A. Delgado, C. Denz, and F. Holtmann, *Detection of microorganismic flows by linear and nonlinear optical methods and automatic correction of erroneous image artefacts and moving boundaries in image generating methods by a neuronumerical hybrid implementing the Taylor's hypothesis as a priori knowledge*, Experiments in Fluids **42** (2007), no. 4, 611–623.

[PTVF07] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes: The art of scientific computing*, 3d edition ed., ch. Eigensystems, Cambridge University Press, 2007.

[PVH$^+$02] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch, *Feature extraction and visualization of flow fields*, Eurographics, September 2002, State of the Art Reports, pp. 69–100.

[PVH$^+$03] ______, *The state of the art in flow visualization: Feature extraction and tracking*, Computer Graphics Forum, vol. 22, 2003, pp. 775–792.

[PvW93] F. Post and T. van Walsum, *Focus on scientific visualization*, ch. Fluid Flow Visualization, pp. 1–40, Springer-Verlag, Berlin, 1993.

[QPK98] G. M. Quenot, J. D. Pakleza, and T. A. Kowalewski, *Particle image velocimetry with optical flow*, Experiments in Fluids **25** (1998), no. 3, 177–189.

[RFH02] J. Rohály, F. Frigerio, and D. P. Hart, *Reverse hierarchical PIV processing*, Measurement Science and Technology **13** (2002), 984–996.

[RKSN05] P. Ruhnau, T. Kohlberger, C. Schnörr, and H. Nobach, *Variational optical flow estimation for particle image velocimetry*, Experiments in Fluids **38** (2005), no. 1, 21–32.

[RNI00] J. Rohály, T. Nakajima, and Y. Ikeda, *Identification of true particle image displacement based on false correlation symmetry at poor signal peak detectability*, Experiments in Fluids **29** (2000), S23–S33.

[Roe] T. Roesgen, *Optimal sub-pixel interpolation in particle image velocimetry*, Experiments in Fluids **35**, no. 3.

[RR01] H. Richard and M. Raffel, *Principle and applications of the background oriented schlieren (BOS) method*, Measurement Science and Technology **12** (2001), 1576–1585.

[RRK98] O. Ronneberger, M. Raffel, and J. Kompenhans, *Advanced evaluation algorithms for standard and dual plane particle image velocimetry*, The 9th International Symposium on Applications of Laser Techniques in Fluid Mechanics, 1998.

[RS07] P. Ruhnau and C. Schnörr, *Optical stokes flow estimation: an imaging-based control approach*, Experiments in Fluids **42** (2007), no. 1, 61–78.

[RSY01] X. Ruan, X. Song, and F. Yamamoto, *Direct measurement of the vorticity field in digital particle images*, Experiments in Fluids **30** (2001), no. 6, 696–704.

[RWK01] M. Raffel, C. E. Willert, and J. Kompenhans, *Particle image velocimetry: A practical guide*, 2nd edition ed., Springer-Verlag, Berlin Heidelberg New York, 2001.

[SBB04] A. M. Shinneeb, J. D. Bugg, and R. Balachandar, *Variable threshold outlier identification in PIV data*, Measurement Science and Technology **15** (2004), no. 9, 1722–1732.

[Sca02] F. Scarano, *Iterative image deformation methods in PIV*, Measurement Science and Technology **13** (2002), no. 1, R1–R19.

[Sca04] ______, *A super-resolution particle image velocimetry interrogation approach by means of velocity second derivatives correlation*, Measurement Science and Technology **15** (2004), no. 2, 475–486.

[SKB$^+$06] J. Schneider, J. Krüger, K. Bürger, P. Kondratieva, and R. Westermann, *California Streaming*, IEEE Visualization Contest 2006 winning entry – http://wwwcg.in.tum.de/vis-contest06, 2006.

[SKKW05] J. Schneider, P. Kondratieva, J. Krüger, and R. Westermann, *All you need is – Particles!*, IEEE Visualization Contest 2005 winning entry – http://wwwcg.in.tum.de/Research/Projects/VisContest05, 2005.

[Slá81] V. Sládecek, *Indicator value of the genus Opercularia (Ciliata)*, Hydrobiologia **79** (1981), no. 3, 229–232.

[SLBS03] M. Samimy, L. G. Leal, K. S. Breuer, and P.H. Steen, *A gallery of fluid motion*, Cambridge Univeristy Press, 2003.

[SLM05] S. C. Shadden, F. Lekien, and J. E. Marsden, *Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows*, Physica D: Nonlinear Phenomena **212** (2005), no. 3–4, 271–304.

[SP07] F. Sadlo and R. Peikert, *Visualizing Lagrangian coherent structures and comparison to vector field topology*, Topology-based Methods in Mathematics + Visualization, 2007.

[SPS99] D. Silver, F. Post, and I. Sadarjoen, *The encyclopedia of electrical and electronics engineering*, vol. 7, ch. Flow Visualization, pp. 640–652, John Wiley & Sons, New York, 1999.

[SR99] F. Scarano and M. L. Riethmuller, *Iterative multigrid approach in PIV image processing*, Experiments in Fluids **26** (1999), 513523.

[SR00] ______, *Advances in iterative multigrid PIV image processing*, Experiments in Fluids **29** (2000), no. 7, 51–60.

[Sta99] J. Stam, *Stable fluids*, The 26th annual conference on computer graphics and interactive techniques, 1999, pp. 121–128.

[Sut94] D. Suter, *Motion estimation and vector splines*, Computer Vision and Pattern Recognition, 1994, pp. 939–942.

[TD95] P. T. Tokumaru and P. E. Dimotakis, *Image correlation velocimetry*, Experiments in Fluids **19** (1995), no. 1, 1–15.

[TMKK05] M. Thomas, S. Misra, C. Kambhamettu, and J. T. Kirby, *A robust motion estimation algorithm for PIV*, Measurement Science and Technology **16** (2005), no. 3, 865–877.

[TvW99] A. Telea and J. J. van Wijk, *Simplified representation of vector fields*, IEEE Visualization, 1999, pp. 35–42.

[TvW03] ______, *3D IBFV: Hardware-Accelerated 3D Flow Visualization*, IEEE Conference on Visualization 2003, 2003, pp. 233–240.

[TWBW99] D. S. Tuch, R. M. Weisskoff, J. W. Belliveau, and V. J. Wedeen, *High angular resolution diffusion imaging of the human brain*, The 7th Annual Meeting of ISMRM, 1999, p. 321.

[USM96] S. Ueng, K. Sikorski, and K. Ma, *Efficient streamline, streamribbon, and streamtube constructions on unstructured grids*, Transactions on Visualization and Computer Graphics, IEEE, 1996, pp. 2:100–110.

[Utz03] L. R. P. Utz, *Identification, life history, and ecology of peritrich ciliates as epibionts on calanoid copepods in the Chesapeake bay*, Ph.D. thesis, University of Maryland, 2003.

[van95] T. van Walsum, *Selective visualization on curvilinear grids*, Ph.D. thesis, Delft University of Technology, 1995.

[VBvP04] A. Vilanova, G. Berenschot, and C. van Pul, *DTI visualization with streamsurfaces and evenly–spaced volume seeding*, VisSym '04 Joint EG –IEEE TCVG Symposium on Visualization, 2004, pp. 173–182.

[VGB$^+$05] I. Viola, M. E. Gröller, K. Bühler, M. Hadwiger, B. Preim, D. Ebert, M. C. Sousa, and D. Stredney, *Illustrative visualization*, IEEE Visualization Tutorial on Illustrative Visualization, 2005.

[VHG02] G. A. Voth, G. Haller, and J.P. Gollub, *Experimental measurements of stretching fields in fluid mixing*, Physical Review Letters **88** (2002), no. 25.

[VV98] R. Verstappen and A. Veldman, *Spectro-consistent discretization of Navier-Stokes: a challenge to RANS and LES*, Journal of Engineering Mathematics **34** (1998), no. 1, 163–179.

[WALL97] R. P. Wildes, M. J. Amabile, A.-M. Lanzillotto, and T.-S. Leu, *Physically based fluid flow recovery from image sequences*, Computer Vision and Pattern Recognition, 1997, pp. 969–975.

[WDG97] J. Westerweel, D. Dabiri, and M. Gharib, *The effect of a discrete window offset on the accuracy of cross-correlation analysis of digital PIV recordings*, Experiments in Fluids **23** (1997), 20–28.

[WE04] D. Weiskopf and G. Erlebacher, *The visualization handbook*, ch. Flow Visualization Overview, Academic Press Inc., US, 2004.

[Wes93] J. Westerweel, *Digital particle image velocimetry: Theory and application*, Ph.D. thesis, Delft University of Technology, 1993.

[Wes94] ______, *Efficient detection of spurious vectors in particle image velocimetry data*, Experiments in Fluids **16** (1994), no. 3, 236–247.

[Wes97] ______, *Fundamentals of digital particle image velocimetry*, Measurement Science and Technology **8** (1997), 1379–1392.

[Wes98] ______, *Effect of sensor geometry on the performance of PIV interrogation*, The 9th International Symposium on Applications of Laser Techniques in Fluid Mechanics, 1998, Paper 1.2.

[Wes00] ______, *Theoretical analysis of the measurement precision in particle image velocimetry*, Experiments in Fluids **29** (2000), no. 7, 03–12.

[WG91] C. Willert and M. Gharib, *Digital particle image velocimetry*, Experiments in Fluids **10** (1991), no. 4, 181–193.

[WG03] S. T. Wereley and L. Gui, *A correlation-based central difference image correction (CDIC) method and application in a four-roll-mill flow PIV measurement*, Experiments in Fluids **34** (2003), 42–51.

[Wig92] S. Wiggins, *Chaotic transport in dynamical systems*, Springer-Verlag, New York, 1992.

[Wil00] C. E. Willert, *Evaluation of digital PIV recordings*, Application of particle image velocimetrytheory and practice, course notes, 2000.

[WKL99] D. M. Weinstein, G. L. Kindlmann, and E. C. Lundberg, *Tensorlines: Advection-diffusion based propagation through diffusion tensor fields*, IEEE Visualization, 1999, pp. 249–254.

[WLW00] M. R. Wiegell, H. B. W. Larsson, and V. J. Wedeen, *Fiber crossing in human brain depicted with diffusion tensor MR imaging*, Radiology **217** (2000), no. 3, 897903.

[WM00] S. T. Wereley and C. D. Meinhart, *Accuracy improvements in particle image velocimetry algorithms*, The 10th International Symposium on Applications of Laser Techniques in Fluid Mechanics, 2000, Paper 13.4.

[WM01] ______, *Second-order accurate particle image velocimetry*, Experiments in Fluids **31** (2001), no. 3, 258–268.

[WMM$^+$02] C.-F. Westin, S.E. Maier, H. Mamata, A. Nabavi, F.A. Jolesz, and R. Kikinis, *Processing and visualization for diffusion tensor MRI*, Medical Image Analysis **6** (2002), 93–108.

[WS05] J. Westerweel and F. Scarano, *Universal outlier detection for PIV data*, Experiments in Fluids **39** (2005), no. 6, 1096–1100.

[YC04] X. Yuan and B. Chen, *Illustrating Surfaces in Volume*, Joint IEEE/EG Symposium on Visualization, May 2004, pp. 9–16, 337.

[YJW04] C. N. Young, D. A. Johnson, and E. J. Weckman, *A model-based validation framework for PIV and PTV*, Experiments in Fluids **36** (2004), no. 1, 23–35.

[YRMS05] J. Yuan, P. Ruhnau, E. Mémin, and C. Schnörr, *Discrete orthogonal decomposition and variational fluid flow estimation*, The 5th international conference on scale space and PDE methods in computer vision, vol. 3459, 2005, pp. 267–278.

[ZB02] L. Zhukov and A. Barr, *Oriented tensor reconstruction: tracing neural pathways from diffusion tensor MRI*, IEEE Visualization, 2002, pp. 387–394.

[ZB03] ———, *Heart-muscle fiber reconstruction*, 2003, pp. 597–602.

[ZC01] M. Zaleski and C. Claps, *First record of some peritrichous ciliates for San Miguel del Monte pond (Buenos Aires, Argentina)*, Gayana (Concepc.) **65** (2001), no. 1, 27–36.

[ZCML00] S. Zhang, C. T. Curry, D. S. Morris, and D. H. Laidlaw, *Streamtubes and streamsurfaces for visualizing diffusion tensor MRI volume images*, Visualization '00 Work in Progress, October 2000.

[ZH97] M. Zöckler and D. Stalling H.-C. Hege, *Parallel line integral convolution*, Parallel Computing **23** (1997), no. 7, 975–989.

[ZYLL07] E. Zhang, H. Yeh, Z. Lin, and R. S. Laramee, *Asymmetric tensor analysis for flow visualization*, under review, 2007.