Technische Universität München
Fakultät für Informatik
Professur für Physik-basierte Simulation

Optimization for Fluid Simulation and Reconstruction of Real-World Flow Phenomena

Marie-Lena Katharina Noemi Eckert

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.

Technische Universität München
Fakultät für Informatik
Professur für Physik-basierte Simulation

Optimization for Fluid Simulation and Reconstruction of Real-World Flow Phenomena

Marie-Lena Katharina Noemi Eckert

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.

Vorsitzender:	Prof. Dr.-Ing. Matthias Althoff
Prüfende der Dissertation:	1. Prof. Dr.-Ing. Nils Thürey
	2. Prof. Dr.-Ing. Matthias Teschner, Albert-Ludwigs-Universität Freiburg

Die Dissertation wurde am 06.08.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 25.08.2019 angenommen.

To my family and friends

Abstract

Fluid effects like spilling water, coiling honey, dancing flames, or smoke plumes are part of our daily life. As such, they certainly play an important role in manifold computer animations for the entertainment industry, e.g., in feature films. To facilitate the creation of convincing effects, numerical simulation is commonly applied to model the physical behavior of fluids like water or air. However, the resulting motions and shapes are not easily predicted by human artists. Furthermore, to produce realistic flows properly with turbulent structures, typically high and expensive simulation resolutions are required. An alternative approach to forward fluid simulation is the reconstruction of three-dimensional real-world fluid phenomena from two-dimensional video captures. Reconstructions yield visually plausible and realistic effects with behavior known beforehand, independent of resolution.

In this thesis, we introduce a powerful convex optimization technique, the *fast Primal-Dual method* known from the image processing and machine learning areas, to the area of fluid simulation. We employ it to develop techniques targeting the improvement of control and realism of fluid simulations for animation. With the Primal-Dual method, we are able to split complex multi-objective functions into separate sub-problems, which are often easier to solve.

First, we present an efficient, physics-based method to reconstruct volumetric density and motion accurately from sparse captures of real-world single-phase fluid phenomena, i.e., smoke. In order to reconstruct smoke plumes precisely from such a small set of views, we constrain a forward fluid simulation based on the incompressible Navier-Stokes equations to meet the two-dimensional input videos. As matching boundary conditions are crucial for numeric simulations, we provide a novel estimation method for the unseen density inflow. By definition, our reconstructions reproduce real-world fluid phenomena and exhibit known shapes and motions. We gather those reconstructions of turbulent, natural, and buoyancy-driven scalar transport flows to create our large-scale data set *ScalarFlow*. This data set is the first of its kind and has many potential applications. For instance, it is now possible to evaluate and compare different simulation methods based on real-world phenomena. A first perceptual evaluation study demonstrates that our accurate reconstructions with moderate resolutions capture the natural complexity of real-world smoke plumes, which would require large simulation resolutions for regular fluid solvers. Furthermore, *ScalarFlow* enables training of neural networks with real-world fluid data, which could lead to more insights about fluid simulation and to faster simulation techniques in the future.

To create reconstructions that match reality closely, multiple camera views are required as input.

However, minimizing the number of input views to one drastically reduces the complexity of the capturing setup. With only a single view, much more data sources are eligible as input, e.g., visual effects companies' or regular online video databases, or even smartphone and consumer camera recordings. Therefore, we additionally propose a single-view reconstruction technique, which is highly challenging since with only a single sequence of two-dimensional input images, the three-dimensional reconstruction problem becomes heavily under-constrained, especially in the depth direction. Yet, by using the Primal-Dual method, we present a powerful single-view reconstruction technique that incorporates several physical constraints and additional physical priors, e.g., a depth-based regularizer. Our method finds suitable motion solutions within the null-space of the under-determined reconstruction problem that feature natural and realistic fluid behavior from all angles.

When reconstructing real-world phenomena, we guide a forward simulation to match the given input images. Such concept of constraining a simulation's resulting motion and shape towards a desired state is called fluid guiding, which additionally allows for artistic modeling. We present a guiding approach through prescribing a target velocity field by either an artist or the motion of a low-resolution simulation. The challenge for fluid guiding is to exert not only control but also to preserve appealing fluid characteristics that are not present in the target velocity. With the Primal-Dual method, we are able to achieve both: we obtain explicit control over large-scale motion and small-scale details without prohibiting the creation of complex fluid simulation features.

In order to achieve realistic fluid simulations, it is often required to solve inequality constraints, e.g., for separating solid-wall boundary conditions. For example, when simulating liquids with the *Fluid Implicit Particle* method, unrealistic artifacts of fluid crawling up solid walls and sticking to ceilings are frequently present. These artifacts can be prevented by allowing the normal component of the velocity at fluid-solid faces to be greater or equal to zero. Solving such inequality constraints typically leads to expensive *Quadratic Programming* solvers. By employing the Primal-Dual method, we adapt a regular pressure solver to account for separating solid-wall boundary conditions requiring simply a few changes to existing implementations. Our presented solver provides much more realistic large-scale liquid behavior with only additional 12 % of run time costs.

In summary, our multi- and single-view reconstruction techniques, fluid guiding scheme, and separating boundary conditions solver reduce present artifacts, increase control, and emphasize real-world features in the resulting visual effects. With our reference data set *ScalarFlow*, we enable the evaluation and development of simulation or neural network methods in order to further improve efficiency and realism of fluid effects.

Zusammenfassung

Fluideffekte, wie überlaufendes Wasser, zähflüssiger Honig, züngelnde Flammen oder Rauchwolken, sind Teil unseres täglichen Lebens und spielen deshalb in der Computeranimation für die Unterhaltungsindustrie eine wichtige Rolle, beispielsweise in Spielfilmen. Um die Erschaffung überzeugender Effekte zu ermöglichen, wird häufig numerische Simulation zur Modellierung des physikalischen Verhaltens von Fluiden, wie Wasser oder Luft, eingesetzt. Künstler haben jedoch Schwierigkeiten, die daraus resultierenden Bewegungen und Formen im Voraus abzuschätzen. Zudem wird typischerweise eine hoch aufgelöste und teure Simulation benötigt, um realistisches Verhalten mit turbulenten Strukturen zu erzeugen. Als Alternative zur Vorwärtssimulation kann die Rekonstruktion realer dreidimensionaler Fluidphänomene anhand von zweidimensionalen Videoaufnahmen eingesetzt werden. Die Rekonstruktion liefert unabhängig von der Auflösung realistische Effekte, die dem Verhalten aus den Aufnahmen folgen.

In dieser Arbeit stellen wir eine mächtige, aus den Bereichen der Bildverarbeitung und des maschinellen Lernens bekannte, konvexe Optimierungstechnik, die *schnelle Primal-Dual-Methode* vor. Wir setzen diese zur Entwicklung verschiedener Algorithmen ein, welche die Kontrolle und den Realismus von Strömungssimulationen in der Animation verbessern. Mit der Primal-Dual-Methode ist es möglich, eine komplexe mehrkriterielle Funktion in Unterprobleme aufzuteilen, die oft einfacher zu lösen sind.

Zuerst stellen wir eine effiziente, Physik-basierte Methode zur genauen Rekonstruktion realer einphasiger Fluidphänomene, z.B. Rauch, vor. Unsere Methode nutzt als Eingabe wenige zweidimensionale Kameraaufnahmen und liefert die volumetrische Dichte und das Bewegungsfeld des Fluides. Um Rauchwolken aus einer geringen Anzahl von Ansichten akkurat zu rekonstruieren, beschränken wir eine Vorwärtssimulation auf der Grundlage der inkompressiblen Navier-Stokes-Gleichungen, sodass sie mit den Eingabevideos übereinstimmt. Da passende Randbedingungen für numerische Simulationen wesentlich sind, stellen wir eine neuartige Schätzung des ungesehenen Rauchzuflusses vor. Per Definition geben unsere Rekonstruktionen reale Fluidphänomene wieder und weisen zuvor bekannte Formen und Bewegungen auf. Diese Rekonstruktionen turbulenter, natürlicher und auftriebsgetriebener skalarer Transportströme fassen wir in einem großen Datensatz *ScalarFlow* zusammen. Dieser ist der erste seiner Art und findet Einsatz in zahlreichen Anwendungen. Beispielsweise ist es nun möglich, verschiedene Simulationsmethoden basierend auf realen Phänomenen zu bewerten und zu vergleichen. Eine erste Wahrnehmungsstudie zeigt, dass unsere präzisen Rekonstruktionen mit moderater Auflösung die natürliche Komplexität von realen Rauchwolken erfassen, wohingegen klassische Fluidlöser dafür hohe Simulationsauflösungen benötigen würden. Darüber hinaus ermöglicht *ScalarFlow* das Trainieren neuronaler Netze mit realen Fluiddaten, welche zu neuen Erkenntnissen in der Fluidsimulation und in Zukunft auch zu schnelleren Simulationstechniken führen könnten.

Um genaue Rekonstruktionen zu erstellen, sind mehrere Kameraaufnahmen als Eingabe erforderlich. Reduziert man jedoch die Anzahl der Kameraansichten auf eine einzige, so verringert sich die Komplexität der Aufnahmeeinrichtung drastisch. Dieser Schritt ermöglicht verschiedenste Aufnahmequellen als Eingabe, z.B. online Videodatenbanken, Datenbanken von Spezialeffektunternehmen und sogar Videos von Smartphones oder handelsüblichen Handkameras. Daher stellen wir zusätzlich eine Rekonstruktionstechnik vor, die nur eine Ansicht als Eingabe erhält. Dies stellt eine besondere Herausforderung dar, da mit nur einer Sequenz von zweidimensionalen Eingabebildern das Problem der dreidimensionalen Rekonstruktion stark unterbestimmt wird, insbesondere in der Tiefenrichtung. Dennoch präsentieren wir unter der Verwendung der Primal-Dual-Methode eine leistungsstarke Rekonstruktionstechnik, die mehrere physikalische Bedingungen und Annahmen beinhaltet, z.B. einen tiefenbasierten Regularisierer. Unsere Methode findet geeignete Lösungen im Nullraum des unterbestimmten Rekonstruktionsproblems, die aus allen Sichten ein natürliches und realistisches Strömungsverhalten aufweisen.

Beim Rekonstruieren realer Fluidphänomene wird im Grunde eine Vorwärtssimulation anhand der Eingabebilder gelenkt. Dieses Konzept wird Fluidführung genannt, welches außerdem künstlerische Modellierungen des Fluides erlaubt. Wir präsentieren eine Methode der Fluidführung, bei der durch einen Künstler oder durch eine niedrigaufgelöste Simulation ein Sollgeschwindigkeitsfeld vorgegeben wird. Die Herausforderung der Fluidführung besteht darin, nicht nur Kontrolle auszuüben, sondern auch ansprechende, durch die Simulation kreierte Fluideigenschaften zu erhalten, die im Sollgeschwindigkeitsfeld nicht enthalten sind. Mit der Primal-Dual-Methode erreichen wir beides: Wir erhalten explizite Kontrolle über großflächige Bewegungen und kleinere Details, ohne die Erstellung komplexer Fluidcharakteristika zu unterbinden.

Um realistische Strömungssimulationen zu erstellen, ist es oft erforderlich Ungleichheitsbedingungen zu erfüllen, z.B. für separierende Randbedingungen zwischen Fluiden und Festkörpern. Zum Beispiel entstehen bei der Simulation von Flüssigkeiten mit der *Fluid-Implicit-Particle*-Methode häufig unrealistische Artefakte: Flüssigkeit kriecht an Wänden hoch und haftet an Decken. Diese können verhindert werden, indem die Geschwindigkeit orthogonal zu Flächen zwischen Fluid und Festkörpern auf größer oder gleich Null beschränkt wird. Die Lösung solcher Ungleichheitsbedingungen führt in der Regel zu teuren Lösern der *Quadratischen Optimierung*. Durch den Einsatz der Primal-Dual-Methode passen wir einen regulären Drucklöser von Simulationen so an, dass durch wenige Änderungen an bestehenden Implementierungen mit nur 12 % zusätzlichen Laufzeitkosten separierende Festkörper-Randbedingungen erfüllt werden, die ein realistischeres großflächiges Flüssigkeitsverhalten ermöglichen.

Unsere Techniken zur Multi- und Einzelansicht-Rekonstruktion, der Fluidführung und der separierenden Randbedingungen reduzieren vorhandene Artefakte, erhöhen die Kontrolle und verstärken realistisches Verhalten der resultierenden visuellen Effekte. Mit der Bereitstellung unseres Referenzdatensatzes *ScalarFlow* ermöglichen wir die Bewertung und Entwicklung von Simulationsmethoden oder Neuronalen Netzen zur weiteren Verbesserung der Effizienz und des Realismus von simulierten Fluideffekten.

Acknowledgments

First of all, I want to thank my supervisor Nils Thürey, who enabled and accompanied my journey of getting started with physics-based numerical simulation as a master's student in 2013 and finishing my PhD in 2019. Throughout these years, I was fortunate to be inspired by his ideas, to learn from him how to approach research problems, and to be encouraged when research seemed too challenging. He always took his time for meetings and ensured to answer questions promptly. Furthermore, I thank Matthias Teschner for inspirational conversations, for the possibility to get to know his research team and highly interesting research approaches in Freiburg, for being my session chair twice, and his kind support during my PhD. I am also thankful to Rüdiger Westermann for his helpful encouragement and wise guidance.

I am grateful to all my co-workers for highly competent research discussions and brainstorming as well as for delightful social interactions, which were not limited to lunch breaks. I especially grew through the helpful communication with my experienced postdoc co-workers Sebastian Eberhardt, Tiffany Inglis, and Kiwon Um. Special thanks goes to my fellow PhD candidates Mathias Kanzler, Michael Kern, Alexander Kumpf, Lukas Prantl, and Steffen Wiewel for proof reading and for so much more. I am also grateful for my indispensable co-workers Rachel Chu, Seyedbehdad Ghaffari, Kevin Höhlein, Philipp Holl, Henrik Masbruch, Christian Reinbold, Sebastian Weiss, and You Xie as well as to Susanne Weitz and Sebastian Wohner. In addition, it was a pleasure to meet Matthias Niessner and his research group, who brought more liveliness and inspiration to south of the wall. Thank you so much for your company!

Moreover, I greatly appreciate the presence and assistance of my dear friends Natascha Abrek and Matthias Gollwitzer – my adventure in computer science wouldn't have been the same without you. Last but not least, I thank all my friends and my family Waltraud, Thomas, Heike, Werner, Ilonka, Aranka, and Christoph for their ongoing support and encouragement during my PhD and their belief in my abilities.

Contents

Nomenclature

Fluid Simulation

t	time
p	pressure
$\mathbf{u}$	velocity
Φ	density
ρ	physical density in the context of NSE, which is set to 1 in this thesis
ν	kinematic viscosity
$\mathbf{f}_{\text{ext}}$	external forces
Γ	whole simulation and reconstruction domain

Convex Optimization

$\mathbf{prox}$	proximal operator
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	general optimization variables with iterative updates
ξ	general input variable
ρ	parameter to control convergence in the context of ADMM
σ, τ, θ	parameters to control convergence for PD
α, β	weights for smoothness and kinetic energy regularizers
r, s	primal and dual residuals
$\epsilon^{\text{pri}}, \epsilon^{\text{dual}}$	primal and dual thresholds
$\epsilon^{\text{abs}}, \epsilon^{\text{rel}}$	absolute and relative thresholds
ϵ_{CG}	threshold denoting the accuracy of a CG solver
n_{dim}	number of dimensions (three in the 3D case)

Reconstruction

$\tilde{\mathbf{u}}$	predicted velocity
$\Delta\mathbf{u}$	residual velocity
$\Delta\mathbf{u}_{\mathrm{tmp}}$	temporary residual velocity
$\Delta\mathbf{u}_{\mathrm{C}}$	coarse residual velocity (from a lower resolution)
Φ_{tar}	target density
$\Delta\Phi_{\mathrm{tar}}$	residual target density
$\tilde{\Phi}$	predicted density
$\Delta\Phi$	residual density
$\Delta\Phi_{\mathrm{tmp}}$	temporary residual density
$\Delta\Phi_{cor}$	residual density correction
Φ_{I}	density inflow
$\tilde{\Phi}_{\mathrm{I}}$	predicted density inflow
$\tilde{\Phi}_{\mathrm{I,new}}$	new predicted density inflow
i	input images
$\tilde{i}$	predicted images (projected predicted density)
Δi	residual images (projected residual density)
Δi_{cor}	correction images (projected residual density correction)
Γ_I	invisible inflow area
Γ_R	area reachable from the inflow area through advection with a given velocity
Γ_{RI}	part of inflow area ending up in visible domain after advection with a given velocity
N	total number of cells in the simulation or reconstruction domain
γ	camera viewing angle
P	matrix projecting 3D density to 2D images
R	symmetric matrix with regularizers
S	matrix summing up depth velocities along one voxel row
$\mathbb{R}^N_{\geq 0}$	space of non-negative N-dimensional real vectors

Π_{NN}	projection onto the space of non-negative density field
C_{DIV}	space of divergence-free velocity fields
Π_{DIV}	projection onto C_{DIV}
$C_{\text{NN normal}}$	space of velocity fields with non-negative normal velocity components at fluid-solid faces
$\Pi_{\text{NN normal}}$	projection onto $C_{\text{NN normal}}$

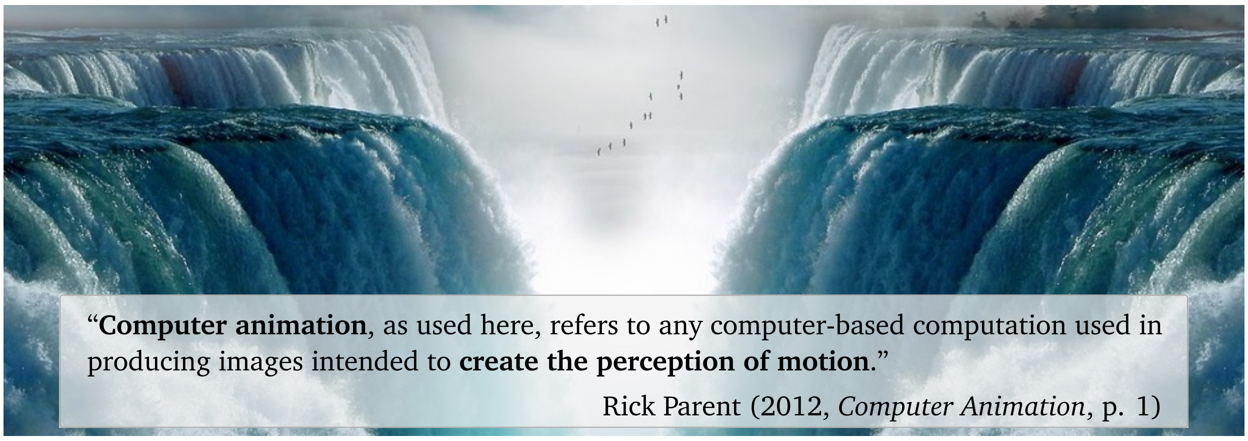
Guiding and Boundary Conditions

$\mathbf{u}_{\text{cur}}$	current velocity
$\mathbf{u}_{\text{tar}}$	target velocity
r_β	blur radius controlling the influence of $\mathbf{u}_{\text{cur}}$ in fluid guiding
G	convolution matrix of Gaussian blur
W	matrix of guiding strength parameters
r_{blend}	ratio for *ratio blending*
$\mathbf{m}$	memory velocity field for separating boundary conditions
$\hat{\mathbf{n}}$	normal vector at a fluid-solid face
S_{nsep}	set of non-separating fluid-solid faces

Abbreviations

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
ADMM	Alternating Direction Methods of Multipliers
BC	Boundary Condition
CFD	Computational Fluid Dynamics
CG	Conjugate Gradient
CGLS	Conjugate Gradient Least Squares
CT	Computed Tomography
IOP	Iterated Orthogonal Projection
FLIP	Fluid-Implicit Particle
LCP	Linear Complementarity Problem
NSE	Navier-Stokes equations
OF	Optical Flow
PD	*fast first-order Primal-Dual method* proposed by Pock et al. [PCBC09]
PSNR	peak signal-to-noise ratio
QP	Quadratic Programming

1 Introduction



"**Computer animation**, as used here, refers to any computer-based computation used in producing images intended to **create the perception of motion**."

Rick Parent (2012, *Computer Animation*, p. 1)

Nowadays, most images we see on displays and screens are no longer authentic pictures of reality. With today's computational possibilities, imaginary worlds are created with their own environments, objects, and creatures such that viewers are able to fully immerse into virtual worlds, indistinguishable from reality. Computer animation is the art of breathing life into characters, animals, objects, or substances. For example, human characters are moved with their skeletons, muscles, hair, and cloth, and dynamics are added to substances, such as fluids.

Computer animation plays a major role in the real-time and offline entertainment industry, such as in movie production, television, video games, or commercials. Especially the photorealistic creation of the most beautiful, impressive, and breathtaking spectacles of nature observed in fluid effects are indispensable for intense, exciting, and convincing scenes.

Such effects certainly substantiate storytelling and embrace the following materials: water (like the separating ocean to let *Moana (2016)* pass through to raging Te Ka), snow (to illustrate the magical world of princess Elsa in *Frozen (2013)*), fire (as for Drogon's dragon fire in *Game of Thrones (2019)*), lava (for the volcano's eruption in *Jurassic World: Fallen Kingdom (2018)*), explosions (as in various battles in *The Avengers (2011-2019)* movies), or smoke (as for the rising of the creatures in *Godzilla II: King of the Monsters (2019)*). Animation is also commonly applied for scientific applications in medicine, forensics, accident reconstruction, or architecture, and for training pilots, SWAT teams, or nuclear reactor operators [Par12].

In the beginning of the animation era, artistic techniques like stop-motion or keyframing were popular where the animator specifies still frames by hand, e.g., in *King Kong (1933)* or *Gertie the Dinosaur (1912)*. The current demand for realistic, complex, and detailed fluid effects exceeds the abilities of a human animator to create manually. Instead, computational models are used to generate a fluid's motion, e.g., in form of physics-based or behavioral simulation. Besides from animation, numerical simulation is an established technique in various research fields to replace expensive or dangerous real-world experiments with computer simulations. Real-world processes cannot always be experimentally examined, e.g., due to extremely long or short duration of the experiments. Exemplary application areas are mechanical engineering for evaluating elastic solids, chemistry for modeling combustion, melting, or coating processes, nuclear physics for investigating atomic collisions, or traffic engineering for planning street networks. Concerning fluids, numerical simulation usually solves the Navier-Stokes equations (NSE), a set of unstable partial differential equations from the area of Computational Fluid Dynamics (CFD). Fluid simulation is employed for weather or climate forecasts, for optimizing the shape of a vehicle's geometry in wind tunnels, to improve the understanding of blood flow in the human vascular system, and in many other application areas.

Here, we focus on physics-based fluid simulation and real-world fluid reconstructions for animation purposes. Contrary to engineering applications, which are primarily concerned with accuracy, the focus in computer graphics is to generate realistic fluid effects that can be controlled to tell a story. Visual plausibility is essential as humans spot artificial motions easily due to their daily interactions with a wide range of fluid phenomena, e.g., pouring water, stirring milk into coffee, burning candles, or smoke rising from a chimney. However, it is challenging to create complex and visually convincing fluid motion. To enable computationally feasible fluid simulation such that it can be integrated into modern visual effects pipelines, the NSE are commonly simplified, e.g., reduced to the incompressible Euler equations for inviscid and divergence-free flows. There exist two different viewpoints for solving the NSE: the Eulerian viewpoint, where quantities are discretized on a spatial grid, and the Lagrangian viewpoint, where particles carrying several quantities are traced. Besides subsequent various extensions, a significant step for practical fluid simulation was the introduction of the grid-based and first unconditionally stable solver [Sta99], through which simulation time steps can be increased leading to shorter run times.

Physically-based fluid simulation enables animators to generate realistic and manifold fluid behaviors. However, fluids are typically chaotic at observable scales such that small perturbations, e.g., changing a simulation parameter, strongly affect large-scale behavior. Parameters additionally influence each other and are difficult to be determined upfront. Hence, controlling and predicting a simulation's behavior is challenging. Moreover, higher simulation resolutions are required in order to create detailed, realistic, and turbulent fluid flows, which result in rapidly increasing computational costs. Thus, artists usually fine-tune faster low-resolution simulations until the desired motion and visual shapes are met. Unfortunately, increasing the simulation's resolution for final production leads to very different behavior after all. Furthermore, unrealistic artifacts, such as artificial numeric viscosity introducing excessive dissipation or fluid sticking unnaturally long to solid objects, are possible side effects from practical low-resolution simulations and simplified physical models.

An alternative approach to forward fluid simulation is compositing existing fluid videos. Composition of videos makes use of large collections of footage available at visual effects companies, where the videos need to be modified to fit into the targeted surroundings. Such compositing suffers from static, two-dimensional (2D) limitations, which prohibit the consideration of illumination effects or the production of realistic three-dimensional (3D) interactions with a possibly changing environment. However, image-based modeling, where 2D input sequences are used to reconstruct 3D shape and motion, eliminates the 2D limitations and further makes use of available fluid footage. Reconstructing real-world fluid phenomena yields realistic effects with visual shapes and motions known beforehand, independent of resolution and other parameters. Having both density, i.e., material distribution, and motion, we can either directly use the density for rendering an effect from arbitrary viewpoints or easily re-simulate the real phenomenon and adapt it to fit desired artistic purposes. Reconstruction of fluid phenomena belongs to the class of data-driven animation and is similar to motion capture for humans or objects, where live motion is mapped onto graphical objects through models wearing special devices that allow their motion to be tracked by sensors.

In summary, it is highly challenging to create predictable and fast simulations that match real-world fluid phenomena closely. For instance, Fluid-Implicit Particle (FLIP) simulations for liquids feature artifacts of fluid being stuck on solids, leading to fluid crawling up walls and sticking to ceilings. Furthermore, flexible comparison and evaluation of different simulation techniques regarding their realism is infeasible, as only rare real-world fluid reference data exists. Different techniques lead to varying results making it difficult to draw conclusions about superior realism of either technique.

In this thesis, we propose

- a low-cost and simple hardware setup for generating and capturing real fluids (Section 4.1),
- an accurate, physics-based, sparse reconstruction method for real-world fluid flows (Section 4.2),
- a large-scale data set, called *ScalarFlow*, of real-world scalar transport flows (Chapter 5),
- reconstructions from single-views mitigating the effort of camera calibration or synchronization and allowing for much more readily available video data as input (Chapter 6),
- a technique for flexible fluid guiding through prescribing a target velocity field (Chapter 7), and
- a realistic separating solid-wall Boundary Condition (BC) for liquid simulations (Chapter 8).

In order to realize our ideas towards more realistic, controllable, and assessable fluid effects, we introduce and utilize a powerful convex optimization technique for fluid optimization, the *fast first-order Primal-Dual method* proposed by Pock et al. [PCBC09] (PD). This technique, known from the image processing and machine learning areas, is presented in the following section and explained in more detail in Section 3.2.

1.1. Convex Optimization with a *fast Primal-Dual Method*

PD is an established optimization scheme from the computer vision area with an optimal convergence rate for non-smooth convex problems. While many primal-dual methods exist, we refer to this particular instance with PD. Convex optimization problems inherit only one global optimum, which is typically found in a reasonable time, good precision, and independent of initialization. With PD, we are able to split difficult and complex multi-objective functions into separate, modular sub-problems. Those sub-problems are often easier to solve, e.g., with specific and highly optimized solvers. Such divide-and-conquer approaches avoid the need for a monolithic, holistic framework and combine local sub-problem solutions to find a global solution to the complex problem. We utilize PD for each of our multi-objective minimization problems.

Gregson et al. [GITH14] established the connection between the fluid pressure solver and convex optimization while utilizing of the Alternating Direction Methods of Multipliers (ADMM). ADMM is a variant of the augmented Lagrangian scheme, an algorithm for solving constrained optimization problems, and has been proven successful in the fields of statistics and machine learning. ADMM and PD are strongly related, as we will show in Section 3.2.2. However, for the simplest setting of convergence parameters, PD reduces to ADMM while PD exhibits superior convergence rate for more sophisticated parameter settings. An alternative to both PD and ADMM is the Iterated Orthogonal Projection (IOP) [MCPN08] where all sub-problems need to be orthogonal projections, which is not always given. Those orthogonal projections are applied iteratively in order to meet multiple conditions after convergence. In Chapters 7 and 8, we demonstrate that PD outperforms ADMM and IOP for both fluid guiding and separating solid-wall BCs.

1.2. Reconstruction of Real-World Fluid Phenomena

An alternative to purely simulating costly fluid effects is to capture and reconstruct natural fluid phenomena. The resulting outcome can be estimated beforehand and dynamics are realistic by definition. Reconstruction of 3D density and motion from 2D fluid recordings enables the automatic and flexible use of real-world phenomena as visual effects, and can be interpreted as the inverse problem to forward fluid simulation. The input images provide control over the resulting fluid behavior, i.e., parameter fine-tuning is not required and changing the resolution will not affect the reconstructed large-scale behavior. The density can directly be used as visual effect, e.g., by re-rendering it from arbitrary viewpoints. The reconstructed velocity field preserves temporal coherence and allows for editing the captured data once it is reconstructed, e.g., through re-simulation, level-of-detail adjustment, fluid guiding, or domain modification.

The underlying theory of reconstructing 3D data from 2D images originates from Radon [Rad86], who proved that any mathematical function could be recovered from an infinite set of projections. In the late 1960s, this knowledge was used to build the first Computed Tomography (CT) machine, which approximates a 3D object based on a set of X-ray images [Cie11]. Medical applications typically employ complex setups with a large number of different viewpoints, which are often obtained by rotating the reconstruction target itself or by moving a sensor around the target. It is inherently more difficult to capture the volumetric and temporal features of fluids compared to static or slower moving objects, as they are chaotic and highly deformable phenomena. Furthermore, their complex image formation model possibly including transparency, reflection, refraction, absorption, or scattering, such as for smoke or water, increases the challenge of accurately capturing the full properties of fluids based on images. Compared to medical applications, traditional image-based CT methods in the graphics area, e.g., [IM04], [AIH*08], or [GKHH12], are designed for less but still a multitude of input viewing angles. They usually require projections from rather expensive and high-quality capturing sensors, or make use of sophisticated capturing setups. Their reconstruction quality rapidly drops when decreasing the number of sensors, as the corresponding inverse problem becomes under-constrained.

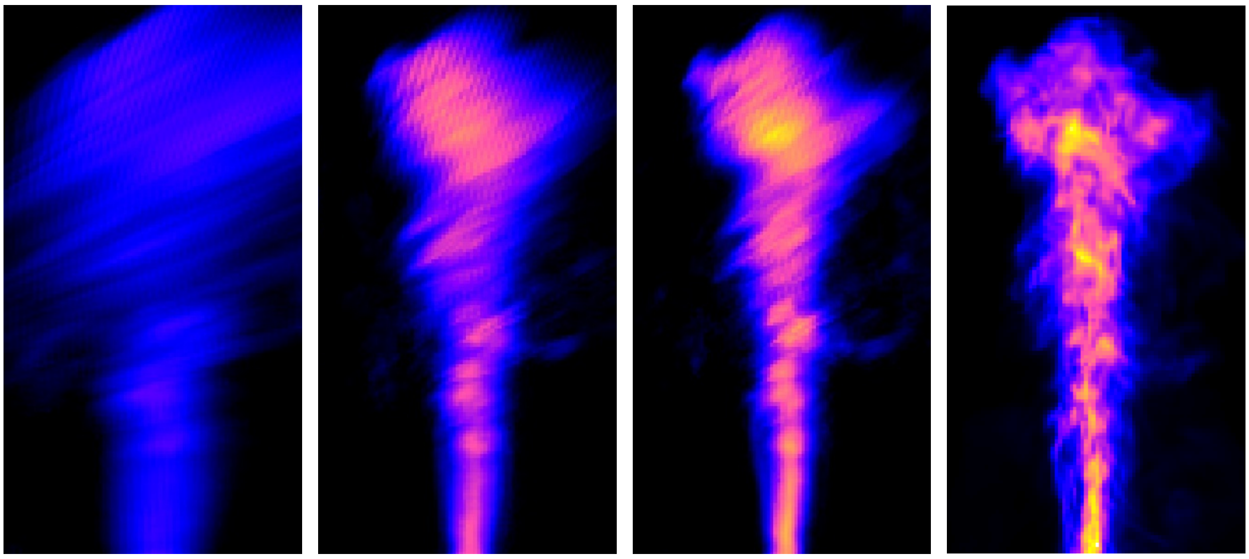


Figure 1.1.: Examples of traditional computed tomography for smoke plumes with two, three, and five input images and our powerful, accurate reconstruction technique based on the same five input images. The quality of the reconstructed volume increases with number of input images and even more when using our powerful reconstruction technique.

Our objective is to use a very sparse number of input images and to make use of a simple capturing setup with inexpensive optical sensors, namely one or five cameras mounted on a Raspberry Pi. Reducing the number of cameras mitigates the efforts for camera calibration and synchronization and allows for flexible capturing setups as input source. With only few, inexpensive sensors, our capturing setup is practical and accessible to a wide range of people. As shown in Figure 1.1, the quality of the reconstructed volume is heavily related to the number of input views. Our powerful reconstruction method overcomes these difficulties and produces a realistic density field, even with only five input views. In this thesis, we target the reconstruction of single-phase flows and, as such, use a common smoke machine producing buoyancy-driven plumes. Technically, our smoke machine diffuses water with propylene glycol, i.e., produces *fog*. However, the underlying physics of smoke and fog are equivalent such that our setup is representative for popular hot smoke simulations [FSJ01]. As publications in graphics typically refer to dense, passively advected tracers as *smoke*, we refer to our tracers as smoke as well.

Capturing the velocity field of a fluid, i.e., velocimetry, is highly challenging, as motion is usually only observed indirectly by tracking temporal changes in observable quantities, e.g., smoke density. However, smoke has no definite features but exhibits non-rigid motion with dynamically changing patterns. For velocity reconstruction, we make use of incompressible 3D Optical Flow (OF) as presented in [GITH14]. An incompressible velocity field is divergence-free, meaning it is mass-preserving without possessing any source or sink. We are solving a heavily under-constrained tomography problem for non-stationary motives, i.e., fluids, and therefore, velocity reconstruction gets even more challenging than in previous work. To counteract the ill-posedness, we constrain a forward fluid simulation based on the incompressible Navier-Stokes equations to meet the 2D input videos. As such, we couple the reconstruction of density and velocity tightly through a novel optimization-based formulation and propose an efficient implementation based on primal-dual optimization including additional physical priors. Our reconstructed quantities are strongly temporal coherent.

Accurate Multi-View Reconstruction for *ScalarFlow* With our physical setup and our reconstruction procedure, we capture a multitude of complex, natural, buoyancy-driven, and turbulent real-world fluid phenomena. We gather them in our novel data set *ScalarFlow*, first of its kind. Such volumetric flow data sets represent a large class of fluid effects in computer animation [FSJ01, SDKN18] and are also important for engineering [MSCL91, YRK18] or medical applications [MNvTK*16], as the observation of transported quantities yields important insights about the underlying flow. Our data set includes volumetric reconstructions (i.e., velocity and density fields), input video sequences, and camera calibration data. For our accurate reconstructions, we use five cameras, and as matching boundary conditions are crucial for numeric simulations, we additionally provide a novel and advanced estimation technique of unseen density inflow. We furthermore incorporate smoothness and kinetic energy regularizers in an efficient tomography scheme. In order to provide accuracy measurements, we reconstruct multiple and varying synthetic forward simulations where we have access to the 3D ground truth quantities. Our reconstructions exhibit a good peak signal-to-noise ratio (PSNR) for density, velocity, and rendered images comparisons.

While typically only a small number of experts visually evaluates results of fluid simulation for animations, other research areas, such as computational photography, geometry processing, and machine learning applications, rely on model and image databases for evaluation. The lack of ground truth reference data for fluid simulation can be explained by the difficulty of capturing the dynamically changing and often semi-transparent 3D fluid density. A fluid's motion can only be indirectly observed by its effect on visible tracers, e.g., smoke particles. However, having a reference and benchmark data set could inspire novel research, lead to new insights and knowledge about internal processes in fluid dynamics, and to analyses and comparisons that are more objective. One example for such data set from the computer vision community is the ImageNet data set [DDS*09], which has been used in thousands of studies, and has encouraged vast progress in image classification research.

ScalarFlow comes with many potential applications. For example, fundamental and established simulation methods for single-phase flows can be evaluated and compared concerning their ability of reproducing real-world fluid phenomena. In order to evaluate the realism of fluid simulations perceptually, it is crucial to have full control over the setup of user studies. Our data set provides this flexibility with a large number of volumetric reconstructions that allow for background modifications, varying smoke colors, different rendering styles, and novel viewing angles. Two-dimensional video recordings do not come with this flexibility. As we will show in Chapter 5, a first perceptual evaluation study demonstrates that our reconstructions are able to reproduce most of the natural complexity from real-world smoke plumes with moderate resolution, which would require large simulation resolutions for regular fluid solvers. Furthermore, deep learning for physics-based simulation is a thriving field. Our data set connects numerical simulations with real-world fluid phenomena, and enables training of neural networks with real-world fluid data, which could lead to better understanding, to novel and improved realistic data-driven approaches, and to faster simulation techniques in the future.

Single-View Reconstruction While previous fluid capture techniques typically require multiple cameras or at least a complex setup; we develop a reconstruction method from a single and simple camera only. Minimizing the number of input views to one drastically reduces the complexity of the capturing setup, reduces costs, and eliminates tedious procedures like camera calibration and synchronization. Single-view reconstruction techniques allow for simpler generation of novel fluid captures and much more data sources as input, e.g., smartphones, consumer cameras, or visual effects companies and regular online video databases. However, single-view reconstruction is highly challenging, as with only a single sequence of 2D input images, the 3D reconstruction problem becomes heavily under-constrained, especially in the depth direction. We aim for plausible reconstructions from arbitrary viewing angles through a tailored combination of several physical constraints and additional geometric priors, e.g., a depth-based regularizer, to resolve motion uncertainty along lines of sight in the single camera view. Our method is able to find suitable motion solutions within the null-space of the under-determined reconstruction problem that feature natural and realistic fluid behavior. In order to demonstrate the capabilities of our method, we first reconstruct a variety of complex synthetic flows generated with fluid simulation and reconstruct smoke volumes captured with our hardware setup afterwards.

1.3. Fluid Guiding

As fluids behave chaotic at human scales, it is impossible to add small-scale details to low-resolution simulations, to modify the simulation domain, or to re-run a simulation with different fluid parameters without influencing the overall large-scale fluid behavior. For storytelling and artistic modeling, however, it is often necessary to drive a simulation towards a desired visual shape or behavior, or to simply add interesting details to existing coarse simulations. For example, creating fictitious but visually plausible motions is very valuable for the entertainment industry as demonstrated by the ocean horse and the smoke dancer in Figure 1.2.

Figure 1.2.: Examples for guiding water to form a horse shape and smoke to model a dancing ballerina. Both fluids preserve a realistic look while assisting fictitious storytelling.

Our goal is to provide a flexible guiding technique with explicit control over large-scale motion and small-scale details without prohibiting the creation of complex fluid simulation features. In particular, we propose fluid guiding through a target velocity field with possibly spatially varying guiding parameters. An artist or a low-resolution input simulation prescribes a velocity field, and the guided fluid is forced to be arbitrarily close to the current simulation's and the target velocity field. At the same time, the velocity field is ensured to be divergence-free. As such, the fluid's motion follows the targeted input velocity and physical laws, i.e., the NSE, creating appealing fluid characteristics not present in the target velocity. We furthermore introduce an efficient method to invert the corresponding system of linear equations approximately. Our technique is applicable for artistic guiding of simulations and level-of-detail adjustment of coarse input simulations. In Chapter 7, we show examples of natural effects where smoke is following a star-shape or moving in an upward spiral motion similar to a tornado. Furthermore, we conduct performance studies of alternative convex optimization techniques such as ADMM and IOP, where PD clearly features superior convergence.

1.4. Separating Solid-Wall Boundary Conditions

In general, BCs play a crucial role in determining the numerical solution of differential equations. Interesting fluid effects include interactions with possibly moving obstacles, a fluid's container like a glass or streambed, or with another fluid, such as the surface between water and air. BCs occur where the fluid volume meets a volume of some other material. There are two types of BC: the solid-wall and the free-surface BCs. In fluid simulation, BCs as well as the constraint of divergence-free motion are enforced in the pressure equations.

Free-surface BCs occur, e.g., at the surface of liquids where they typically meet air. For simplicity, air is usually not explicitly simulated. Instead, the pressure in air is assumed to be zero such that the liquid can move around freely. Enforcing zero pressure in air is a Dirichlet BCs on pressure. Considering solid-wall BCs, the fluid is not allowed to flow into or out of a solid. Therefore, the full velocity or only the normal component of the velocity at fluid-solid faces are set to the corresponding velocity of the solid itself, which is zero for stationary objects. BCs constraining the full velocity are called no-slip or stick BCs. Slip BCs occur when only the normal velocity component is restricted while tangential motion is allowed. The velocity is set to the desired value and the pressure solver does not alter these velocities. As velocities are updated with the pressure gradient, we need to enforce a Neumann BC on the pressure, i.e., set the pressure gradient to zero.

A zero normal velocity component, however, prevents the fluid from separation. While it is physically correct for a very thin film of fluid being left after contact, the thickness of such films in numerical simulation is proportional to the grid resolution, which is generally unnaturally large when simulating large-scale motions. For example, when simulating liquids with the FLIP method, artifacts of fluid sticking to solid obstacles are frequently observed. Fluid crawls up walls and sticks to ceilings leading to behavior perceived as unnatural. Batty et al. [BBB07] allow the normal velocity to be greater or equal to zero, which is expressed as an inequality constraint. Incorporating an inequality constraint into the pressure solver transforms the previously linear system of equations into a Linear Complementarity Problem (LCP), which drastically increases the problem's complexity and typically requires expensive Quadratic Programming (QP) solvers.

Our objective is to adapt the popular preconditioned Conjugate Gradient (CG) method commonly employed in pressure solvers to allow for separating solid-wall BCs with only a few implementation changes. We first classify the interfaces between the fluid and solids into separating and non-separating faces. Then, we set the normal velocity at non-separating faces, e.g., when fluid pushes towards a wall, to zero. In the regular CG pressure solver, all faces are treated as free surface allowing the fluid to move freely. We iterate between classification of fluid-solid faces, setting normal velocities at non-separating faces to zero, and solving the pressure equations until convergence is reached. Therefore, we simultaneously enforce incompressibility and separating solid-wall BCs while maintaining a system of linear equations in the pressure solver. With only an increase of 12 % run time, we are able to produce much more realistic large-scale liquid behavior as shown in multiple 2D and 3D fluid simulations in Chapter 8. PD outperforms both ADMM and IOP, leading to similar performance evaluations as for fluid guiding. Note that in contrast to guiding, the two constraints for separating solid-wall BCs can be expressed as orthogonal projections. We furthermore present an accelerated separating solid-wall BCs solver where the pressure solver is involved in treating non-separating faces regularly, which drastically reduces the number or iterations required for convergence. With such few iterations, the advanced update scheme of PD is not fully exploited and as such, IOP converges slightly faster.

1.5. Outline

First, we give an overview over related works from the areas of fluid simulation, fluid reconstruction, fluid guiding, boundary conditions, and convex optimization in Chapter 2. Then, in Chapter 3, we discuss fundamental methods of fluid simulation, convex optimization, OF, and CT. Next, we present our inexpensive hardware setup for capturing real fluid phenomena and our powerful reconstruction technique in Chapter 4. The obtained accurate reconstructions and the data set *ScalarFlow* are demonstrated and evaluated in Chapter 5. The single-view reconstructions are presented in Chapter 6, while fluid guiding and separating solid-wall BCs are discussed in Chapters 7 and 8. Each of the main chapters describes specifics of the developed method and the corresponding evaluation including results. Moreover, we show that PD outperforms ADMM and IOP for the optimization problems of fluid guiding and separating solid-wall BCs, see Chapters 7 and 8. When evaluating run times of our proposed algorithms, we typically use a single CPU, namely an Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz. In the last Chapter 9, we summarize and discuss each contribution, conclude this thesis, and outline future work.

1.6. List of Publications

Most of the research in this thesis has been originally presented in the following papers, which are published or accepted in peer-reviewed journal articles:

IEGT17 T. Inglis*, M.-L. Eckert*, J. Gregson, N. Thuerey: Primal-Dual Optimization for Fluids. *Computer Graphics Forum*, 36(8):354-368, 2017.
doi:10.1111/cgf.13084
Accompanying video https://youtu.be/Pgbat5MXo8Q

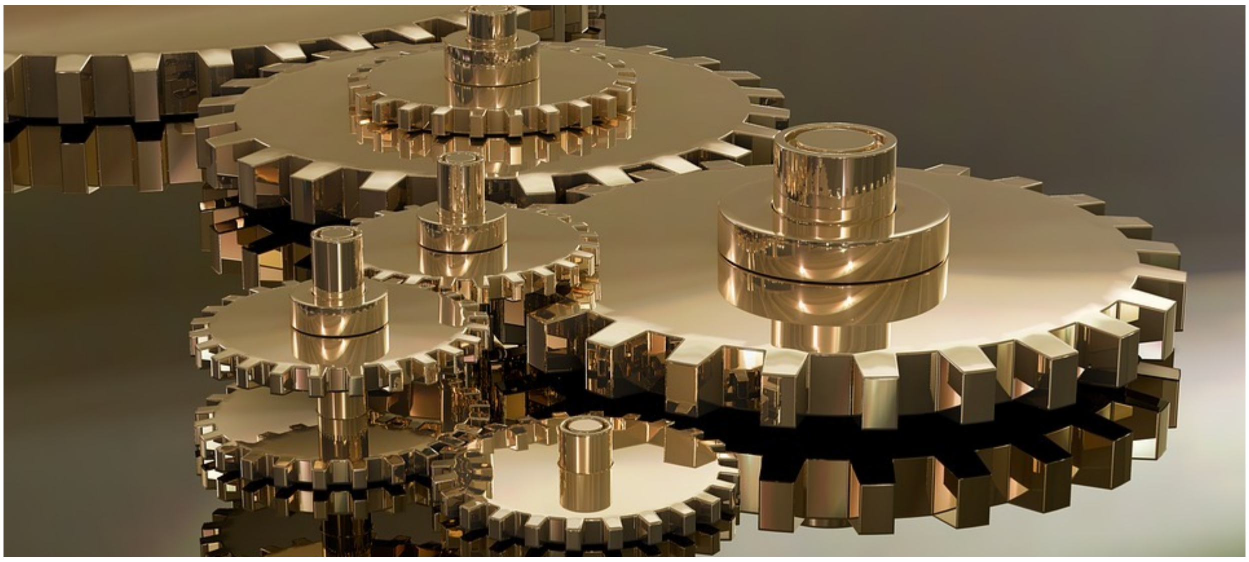
EHT18 M.-L. Eckert, W. Heidrich, N. Thuerey: Coupled Fluid Density and Motion from Single Views. *Computer Graphics Forum*, 37(8):47-58, 2018; **best paper award from SCA'18.**
doi:10.1111/cgf.13511
Accompanying video https://youtu.be/J2wkPNBJLaI

EUT19 M.-L. Eckert, K. Um, N. Thuerey: ScalarFlow: A Large-Scale Volumetric Data Set of Real-world Scalar Transport Flows for Computer Animation and Machine Learning. *ACM Transactions on Graphics*, 38(6):239, 2019.
doi:10.1145/3355089.3356545
Accompanying video https://youtu.be/J0y8zakWQ4Y
Data set https://ge.in.tum.de/publications/2019-scalarflow-eckert/

More precisely, the fluid capturing setup and algorithms in Chapter 4 as well as the results from multi- and single-view reconstructions in Chapters 5 and 6 were published in [EUT19, EHT18]. The Primal-Dual optimizations for fluid guiding and separating solid-wall BCs from Chapters 7 and 8 were published in [IEGT17].

* These authors contributed equally to the paper.

2 Related Work



In this chapter, we give an overview over related research in fluid simulation, fluid reconstruction including methods for density and velocity reconstruction and scientific data sets, fluid guiding, boundary conditions for simulations in graphics, and convex optimization.

2.1. Fluid Simulation

Since Kass and Miller [KM90] introduced a linear wave equation for the simulation of water surfaces in 1990, fluid simulation has become more and more popular in computer graphics. Harlow and his team [Har62, HW65] further presented the Particle-in-cell method (PIC) and the Marker-and-Cell (MAC) grid, a staggered grid for more accurate finite-differences approximations, which is still widely used. Foster and Fedkiw [FM96] were the first to use the MAC grid as discretization scheme for 3D Eulerian liquid simulations. There are particle- and grid-based approaches for simulating fluids: the Lagrangian approach tracing particles and the Eulerian approach using finite grids. Some methods use a hybrid approach where both particles and grids are used. With the Eulerian unconditionally stable solver [Sta99] solving the NSE, real-time fluid interactions became feasible for the first time. Though the presented semi-Lagrangian advection scheme introduces diffusion, simulations can be run much faster, as time steps can be increased due to the unconditional stability. Various extensions have been proposed since then, e.g., for more natural and swirling motion: a hybrid solver for vorticity confinement [FSJ01] and a vortex particle approach [SRF05]; for increasing the perceived resolution of a fluid simulation and therefore saving costs compared to a full NSE approximation [KTJG08, NSCL08, PTC*10]; improvements of the advection scheme in terms of accuracy: the Back and Forth Error Compensation and Correction (BFECC) [KLLR05], the MacCormack advection method reducing mass and momentum loss [SFK*08], the advection-reflection scheme [ZNT18], and most recently BiMocq^2 [QZG*19]; for reducing run time of the pressure solver [MST10, SABS14, Sta01]; or for improved exploitation of grid resolution with dynamically adapted octrees [LGF04].

Smoothed-particle hydrodynamics (SPH) is a purely particle-based method, which can be very beneficial for liquid simulations especially, as mass conservation and convection are dispensable. A study of different SPH methods is presented by Ihmsen et al. [IOS*14]. Based on [DG96] where the authors animate highly deformable bodies, Müller et al. [MCG03] extend SPH to create an interactive method to simulate free-surface flows. The particle level set method (PLS) [FF01] is also commonly used to simulate liquids where the surface is tracked and numerical errors and mass-loss are reduced by using Lagrangian marker particles. [ANSN06, WP10, GNS*12] are popular vorticity-based methods for volumetric flows that use Lagrangian elements as vortex basis. A popular grid-particle hybrid for simulating liquids is FLIP [ZB05] where particles are used for accurate surface tracking and advection. The particles are mapped to an auxiliary grid in order to handle BCs efficiently. FLIP and its variants originate from PIC [Har62] and result in only minor numerical diffusion during the advection stage. Ferstl et al. [FAW*16] reduce computational costs for FLIP by using particles only near the surface of the fluid volume. The now prevalent material point method (MPM) is an extension of FLIP and presents a very useful alternative for simulating complex and fluid-like materials [SSC*13, JST*16, TGK*17].

Robert Bridson gives a comprehensive overview over common fluid simulation techniques in his book [Bri15]. In this thesis, we mainly focus on Eulerian fluid solvers, as they are typically applied for single-phase smoke simulations. However, our separating solid-wall BCs solver is integrated into a FLIP simulation.

2.2. Fluid Reconstruction

Capturing rigid bodies, human shapes, human motions, faces, and flowing material like garment has been investigated by [GW99, DATSS07, DAST*08, ARL*09, BBA*07, BPS*08, WCF07] with a modified Radon transform, marker-based, or markerless methods. Fluids, however, are chaotic and highly deformable phenomena and we are interested in a fluid's inner structure, not only in its surface. Often, high-quality optical sensors or other expensive setups are used to capture their appearances accurately. For example, Kavandi et al. [KCG*90] conduct localized measurements with photoluminescence and imaging techniques such that they are able to determine the pressure of aerodynamic surfaces in wind tunnels. A complex and expensive laser scanning setup is used by [HED05] for capturing smoke where a single high-speed camera records the scattering of a laser sheet swept through the observed volume.

Zang et al. [ZIT*19] reconstruct the inner structure of objects that rapidly deform, such as copper foam crumpling under a compressive force.

Capturing the velocity field of a fluid, i.e., velocimetry, is highly challenging, as motion is only observed indirectly by tracking temporal changes in observable quantities, e.g., smoke density. Particle image velocimetry (PIV) [ESWvO06] is a popular and advanced technique for reconstructing the velocity field of fluids. Carefully chosen, visible particles are introduced to the flow and tracked over time with complex camera setups requiring great spatial and temporal resolution, The particles' positions are then used to reconstruct the flow field. The insertion of particles, however, comes with the risk of perturbing the measurements and are not applicable to all substances, e.g., smoke could obscure the marker particles. Furthermore, PIV is typically restricted to significantly smaller volumes than the volumes required for visual effects. A specialized variant is *Rainbow PIV* by Xiong et al. [XIAP*17] where they illuminate the particles with different colors depending on their depth. Their setup requires even more specialized hardware than traditional PIV methods. Schneiders et al. [SS16, SSE17] present vortex-in-cell-plus (VIC+), which uses volume particle tracking velocimetry (PTV) in order to reconstruct instantaneous velocity fields for turbulent boundary layer experiments in the aerospace engineering area. Their reconstruction method incorporates the temporal derivatives of the velocity, and is able to increase spatial resolution.

In this thesis, we focus on reconstructing smoke based on a sparse number of images taken within a simple setup with inexpensive and easily accessible optical sensors. In order to achieve high quality reconstructions, we propose a strong method for coupled tomographic density updates and velocities utilizing physical constraints. Previous work also employed physical simulations or parts of them to capture liquids [WLZ*09] and divergence-free motions [GITH14]. In the following, we present related works regarding both density and velocity reconstruction.

2.2.1. Density Reconstruction

Shadowgraph and Schlieren methods reveal disturbances in transparent media based on variations in the refractive index, which is influenced by temperature and density. These non-uniformities can be observed as shadows (Shadowgraph) or changes in light refraction (Schlieren) in a background image. For instance, Atcheson et al. [AIH*08] successfully used Schlieren imaging for capturing non-stationary gas flows from sixteen rolling shutter cameras. In contrast, we aim to reconstruct shape and motion of visible phenomena. Hasinoff and Kutulakos [HK03, HK07] reconstruct fire density fields from images based on a decomposition where they divide flames into epipolar slices, i.e., flame sheets, and compute the 3D density field as combination of these 2D sheets. Two photographs of a semi-transparent scene define one flame sheet. In medicine, a CT scan is created by shooting thousands of X-ray beams, usually at least 160 beams from each of 180 directions [Fee10]. Those beams with known intensity pass through the medium where photons are absorbed or scattered related to the electron density of the medium. The number of photons passing through are measured by detector images. Medical CT typically makes use of direct or iterative algebraic reconstruction methods, e.g., the classical filtered backprojection

(FBP) algorithm based on the Radon transform [Rad86], or the simultaneous algebraic reconstruction technique (SART) [AK84, KSW02] and its extensions. Direct, analytical methods struggle with noisy and incomplete data. Iterative methods, on the other hand, work with sparse input data, allow for incorporating additional constraints, and minimize the error term of the correspondence between input images and rendered reconstructed volume.

Driven by medical applications, image-based CT approaches have been developed and are established in graphics [KSW02]. Multiple 2D projections of 3D fluids are created through visible light [Cie11]. In many cases, fewer input images are available compared to medical CT. In order to reconstruct astronomical structures, specialized and powerful algorithms assuming rotational symmetry were proposed [LLM*07, WLM13]. In this thesis, we focus on the reconstruction of single-phase flows as smoke and briefly outline methods for capturing liquids. Morris and Kutulakos [MK11] reconstruct the position and normal of surface points of a liquid from two viewpoints and by using submerged checkerboard patterns. However, internal processes are not investigated. Wang et al. [WLZ*09] make use of structured light and physics-based simulation in combination with image-based reconstruction to capture a dynamic, diffuse liquid surfaces with a single stereo camera. The shape of flowing water is reconstructed by Ihrke et al. [IGM05]. Water is dyed with fluorescent substances and 8-24 video streams are used as input. Slowly deforming objects have also been captured in detail with higher-level optimization tomographies [ZIT*18, ZAI*18].

Considering single-phase flows, Ihrke and Magnor [IM04] present a realistic, iterative, image-based reconstruction of flames using 4-8 input images. They assume a linear image formation model with constant or trilinear basis functions. The projection matrix forms a sparse, ill-conditioned linear system of equations solved with a least squares CG solver. An adaptive algorithm is proposed subsequently to reduce the high computational costs of the reconstruction [IM06]. As the matrix containing pixel-voxel correspondences grows rapidly with the resolution of both images and volume, a matrix-free and gridless approach with small memory footprint is proposed by Gregson et al. [GKHH12], the so-called *Stochastic Tomography*. The authors reconstruct the mixing behavior of two-phase flows, namely mixing ink with water, based on 8-16 high-quality camera input videos. Such liquid mixing is closely related to our single-phase flows of rising hot smoke phenomena. The authors do not build a linear system of equations or impose a discretization beforehand but work with samples created based on stochastic random walks and splat them to residual images. General convex regularizers are easily incorporated such that they achieve high reconstruction quality while requiring less computation time and memory. However, artifacts increase with reducing the number of input views and with calibration errors.

Our goal on the other hand is to reconstruct real-world fluid phenomena with a very sparse number of input views, namely one or five, and to use inexpensive consumer cameras in a simple setup with reduced synchronization efforts. Okabe et al. [ODAO15] present their method *Appearance Transfer* that allows for sparse input views, i.e., one or two, but relies on additional virtual views generated by internal calculations. Based on the original input images, they reconstruct a rough fluid volume, re-render it from novel virtual cameras, transfer the appearance from the original to the novel images, and iterate between reconstructing the volume and adapting the novel image views until a satisfactory result is achieved. Appearance transfer is conducted by using steerable pyramids and histogram matching. Using two input views orthogonal to each other significantly improves their reconstruction quality. The reconstructed densities look realistic but are shaped with the artificially created, virtual input images that do not necessarily match reality. Fuchs et al. [FCG*06] use only a single view to reconstruct scattering media but rely on a set of laser lines sampling the reconstruction volume sparsely with limited spatial resolution.

2.2.2. Velocity Reconstruction

In order to estimate the motion between two 2D images or two 3D density fields, OF methods are typically employed. While there exist local approaches for velocity estimation, e.g., the Lucas-Kanade window technique, we only consider global approaches as fluids behave in global patterns. However, for some applications, it might be beneficial to combine both local and global approaches [BWS05]. Global OF estimation presented by Horn and Schunck [HS81] is very popular and widely used for dense motion reconstructions. They propose a variational formulation of the brightness constancy assumption, which relates the flow to the derivatives of the input densities. However, as infinitely many velocity fields comply with the brightness constancy assumption, the minimization problem is ill-posed and a smoothness regularizer is introduced to alleviate under-determination. Large displacements are often difficult to handle, such that Meinhardt-Llopis [MLPK13] propose using a multi-scale strategy for handling larger motions. In order to account for different illumination or lighting, Radke [Rad13] use a gradient constancy assumption instead of the brightness constancy assumption. Suter [Sut94] decompose the commonly used smoothness regularizer into parts related to the divergence and the curl of the velocity field. Corpetti et al. [CMP02] extend this idea and regularize highly deformable 2D cloud motions with a divergence term and a divergence-curl-smoothness prior. Such regularizer is useful for fluid flows that are known to exhibit concentrations of either divergence or vorticity at some locations. In our case of rising smoke plumes, we assume zero divergence and do not make any assumption about the present vorticity.

Chen et al. [CLH16] reconstruct a sparse, skeletal flow of smoke, and interpolate it to obtain a dense velocity field. Their approach takes place in the 2D image plane and assumes that smoke density does not change much over time. Accompanying the *Appearance Transfer* method mentioned above [ODAO15], the authors propose to use 2D OF to estimate the volumetric motion. Yet, the resulting 3D velocity field is just a rough estimate and not divergence-free, which is a crucial constraint for flow motions in the graphics setting. Gregson et al. [GITH14] reconstruct fluid motion based on divergence-free OF exploiting the fact that the brightness constancy assumption and continuity equation determining fluid flows are equivalent as observed by Liu and Shen [LS08]. The authors make use of ADMM to simultaneously solve for the continuity equation and the divergence-free constraint. They, furthermore, rely on accurate and full volumetric density fields as input, which is often difficult or impossible to access in the visual effects setting. Following the recent success of deep learning approaches, Fischer et al. [DFI*15] learn OF with Flownet, a CNN-based neural network.

Opposed to that, we do not have access to accurate volumetric density fields as input but estimate 3D fluid motion based on 2D input images, which is a heavily under-constrained motion estimation problem. Therefore, we constrain a forward fluid simulation to meet the 2D input videos, couple the density and velocity estimation tightly, and assume zero divergence like Gregson et al. in [GITH14].

2.2.3. Scientific Data Sets

A reference and benchmark data set enables researchers to conduct more objective comparisons and evaluation of methods. One example for data sets with great impact is the ImageNet data set [DDS*09]. It is used in thousands of studies of image classification and contains over one million images with labels and bounding box information for image classification tasks. More date sets in the computer vision area are the CIFAR data set for simplified classification tasks [KNH14], a data set of video collections for action recognition [AEHKL*16], and databases with 3D scanned data for geometry reconstruction [KPZK17, CDF*18]. Fluid simulation for engineering applications can make use of the Johns Hopkins Turbulence Database, which contains eight different simulations of turbulent flows [LPW*08]. However, there exists only one data set for each of the eight setups and the focus lies on non-visual flows, i.e., flows without observable quantities. Other CFD and fluid flow databases are the THT Lab [MK90], KTH Flow [SO10], and FDY DNS [AHOGG14] databases similarly providing single simulated data sets with varying resolutions. Therefore, those databases are not directly applicable for graphics, vision, or deep learning use cases.

Our data set *ScalarFlow* contains many varied data sets of scalar transport processes, i.e., buoyant plumes of hot smoke. We evaluate simulation methods for single-phase flows concerning their ability of reproducing real-world fluid phenomena. In particular, we make use of perceptual evaluations, such as for evaluating rendering algorithms [CCL02], tone mapping [MAF*09], or animations of human characters [HRZ*13]. Most recently, the visual realism of different methods for liquid simulation was evaluated [UHT17]. While providing benchmark cases for fluid simulation methods is a very promising direction, our data set can also be beneficial for the construction of reduced models, e.g., for machine learning applications.

Following the outcome of their perceptual user study on realism of liquid simulations, Um et al. [UHT18] create detailed splashes for liquid simulations with neural networks. Deep learning has been applied to several topics in fluid simulation, e.g., driving particle-based simulations [LJS*15], replacing the pressure solver [TSSP17], augmenting simulated data with learned descriptors [CT17], generating real-time interactions with liquid effects [PBT19], learning controllers for rigid body interactions [MTP*18], super-resolution with strong temporal coherence through adversarial training [XFCT18], learning reduced parameter sets of fluid simulations [KCAT*19], or predicting pressure field changes in liquid simulations [WBT19].

2.3. Fluid Guiding

One limitation of purely physics-based fluid simulation is the lack of artistic control, e.g., simply changing a simulation's resolution alters its behavior significantly. Therefore, many research works target guiding fluid simulations towards a desired state. Pan et al. [PHT*13] use smoke shape keyframes for intuitive control over smoke simulations. Synthesizing turbulent structures decoupled from the large-scale motion facilitates the fine-tuning of details, such as in [BHN07, KTJG08, PTSG09]. Yet, the newly created features do not integrate tightly with the base flow, such that motion is sometimes perceived as unnatural. So-called up-res methods were proposed by [YCZ11, HMK11, HK13, RLL*13, NB11] where low-resolution simulations are edited and guided high-resolution simulations provide fine-scale details by using Lagrangian coherent structures, by sparse sampling of the target flow, by flow modulation using the Hilbert-Huang transform, or by guide shapes for free-surfaces flows. We also integrate an up-res method into our guiding scheme since the separation of low- and high-resolution motion can be quite useful for artists. Our technique allows for arbitrary flow fields as input producing realistic and tightly coupled details.

Fluid guiding through adjoint methods, e.g., [GP00, MTPS04, PM17] for engineering and graphics, tend to be expensive and require the differentiation of the entire fluid solver. Simpler techniques are [SY05b, FL04, SY05a] where they use control forces or velocities for liquids and smoke. However, Thuerey et al. [TKPR06] notice that artificial viscosity is introduced as well. They propose a scale-dependent force control based on automatically generated particles only targeting the coarse-scale velocity components. Gregson et al. [GITH14] present preliminary guiding results using the fast Fourier transform for filtering low-frequencies, which renders the technique impractical for more complex boundary conditions or spatially varying guiding. Most similar to our approach are the works by Nielsen et al. [NCZ*09, NC10]. They propose a multi-scale formulation decoupling large-scale motion and fine-scale detail control with a monolithic system requiring a specialized solver in order to reduce run time. Our technique nicely separates the complex problem of guiding into simpler sub-problems, which are easier to solve.

2.4. Boundary Conditions

As stated in [Bri15], the correct handling of BCs in fluid simulations is a quite difficult task. For example, Forster and Metaxas [FM96] voxelize solid obstacles onto the grid. Nevertheless, geometries not aligning with the grid produce stair-step artifacts, which do not converge to zero when increasing the simulation's resolution. The coupling of deformable objects and incompressible or compressible fluids is targeted with mass-spring systems in [GHD03] and the ghost fluid method in [NGF02]. Takahashi et al. [TFK*03] animate multi-phase flows with non-aligned solids by setting the velocity inside a cell that contains fluid and solid to the velocity of the solid. Rasmussen et al. [REN*04] develop special boundary conditions for the natural interaction of particle level set (PLS) simulations with immersing rigid bodies. The coupling of fluids and rigid bodies is improved by Carlson et al. [CMT04] using distributed Lagrange multipliers for two-way coupling and treating rigid bodies as a so-called rigid fluid. For particle-based fluid simulations, Schechter and Bridson [SB12] introduce ghost particles in surrounding air and solids to reduce particle clumping and thus diffusion. Becker et al. [BTT09] present one- and two-way coupling with rigid bodies based on a predictor-corrector scheme for both velocity and position allowing for different slip conditions while enforcing non-penetration.

Foster and Fedkiw [FF01] introduce the slip BCs where fluid motion tangential to the solid is allowed. They, for the first time, noticed unnatural fluid behavior of liquid sticking to solid obstacles, e.g., domain boundaries. To allow fluid to separate smoothly from solids, the normal velocity component at fluid-solid faces should be allowed to be greater or equal to zero, if the solid is non-moving. Batty et al. [BBB07] propose a second order variational formulation of the pressure equations as kinetic energy minimization problem with the following complementarity condition

$$0 \leq p \perp (\mathbf{u}_{fluid} - \mathbf{u}_{solid}) \cdot \hat{\mathbf{n}} \geq 0, \tag{2.1}$$

where p is the pressure, $\mathbf{u}_{fluid}$ and $\mathbf{u}_{solid}$ are the fluid's and solid's velocities, respectively, and $\hat{\mathbf{n}}$ is the normal of the fluid-solid face. In order to solve their LCP, they apply an expensive QP solver. The complementarity condition states that if the pressure is positive, the fluid is at rest and no suction exists towards the solid wall. A zero pressure, however, allows for positive normal velocities leading to separation of contact. Chentanez et al. [CMF12] incorporate the complementarity condition into their multigrid solver reducing run time drastically. Erleben [Erl13] provides a thorough survey for solving LCPs. More recently, Andersen et al. [ANE17] use a simple implementation of a CG solver to solve the linear complementarity problem. However, their solver is still quite expensive compared to using a traditional CG solver. [NGL10, NGCL09, GB13] solve for unilateral incompressibility to enable separation effects by taking QP approaches, which again require complex solvers. For instance, Gerszewski and Bargteil [GB13] enforce non-negative pressures in cells near liquid surfaces and allow positive divergence in fluid cells to create large-scale splashes.

In contrast, our goal is to adapt the popular CG pressure solver minimally to enable fluid to detach naturally and efficiently from obstacles or solid walls. We present a modular optimization framework allowing for the separate handling of BCs and incompressibility. Similar to our approach, Henderson [Hen12] separates the handling of BCs from the divergence-free projection step but do not handle separating BCs.

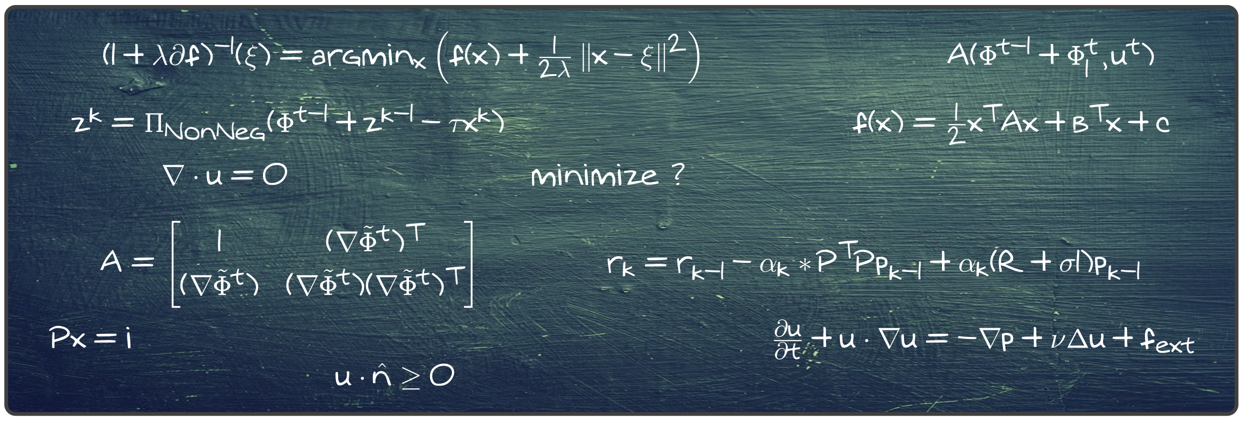
2.5. Convex Optimization

Convex optimization is a well-established and powerful technique, e.g., in computer vision. Considering the areas of statistics and machine learning, Boyd et al. [BPC*11] portray ADMM and review different related methods for convex optimization. Gregson et al. [GITH14] used ADMM as efficient splitting algorithm to make OF divergence-free. They noted that the proximal operator that ensures incompressible fluid velocity is equivalent to the pressure projection commonly solved in fluid simulation in the graphics context. With ADMM, Narain et al. [NOB16] combine a fast and robust non-linear elasticity model with hard constraints for animating deformable objects, while Pan and Manocha [PM17] present a space-time control for smoke animations. Xiong et al. [XIAP*17], on the other hand, applied ADMM to solve the minimization problems of their method *Rainbow PIV*.

The *fast first-order Primal-Dual method* was initially and then in more depth presented by [PCBC09] and [CP11]. PD is a preconditioned version of ADMM with convergence accelerated by improved update directions. In contrast to our results, O'Connor et al. [OV14] found that ADMM outperforms PD for few iteration counts. However, they used a restricted version of PD not exploiting the full potential of the optimal update directions. PD has recently become more popular and was used for projective dynamics simulations [NOB16], correcting deficiencies of simple lenses [HRH*13], for choosing optimal image priors and optimization algorithms for image processing tasks, such as deconvolution, denoising, or inpainting [HDN*16], and for reconstructing rapidly deforming objects [ZIT*19].

Another, specialized method for convex optimization is IOP, which requires all operators to be orthogonal projections. Molemaker et al. [MCPN08] use it to ensure both zero divergence as well as solid-wall BCs by applying both proximal operators iteratively and sequentially . However, they did not account for free-surface BCs as Dirichlet BCs do not translate to linear constraints on the fluid's velocity field. They noted that IOP exhibits slow convergence if the combined matrix of incompressibility and BCs constraints has eigenvalues close to one, which happens when the individual sub-spaces are almost parallel. The position-based fluids (PBF) method by Macklin and Müller [MM13] also makes use of iterative application of projections. As we found PD to outperform both ADMM and IOP for regular iteration counts for both fluid guiding and solid-wall BCs, we focus on using PD in our reconstruction methods.

3 Fundamental Methods


$(I+\lambda\partial f)^{-1}(\xi) = \text{argmin}_x \left(f(x) + \frac{1}{2\lambda}\|x-\xi\|^2\right)$
$A(\Phi^{t-1}+\Phi_1^t, u^t)$
$z^k = \Pi_{NonNeg}(\Phi^{t-1} + z^{k-1} - \tau x^k)$
$f(x) = \frac{1}{2}x^T A x + B^T x + c$
$\nabla \cdot u = 0$
minimize ?
$A = \begin{bmatrix} I & (\nabla\tilde{\Phi}^t)^T \\ (\nabla\tilde{\Phi}^t) & (\nabla\tilde{\Phi}^t)(\nabla\tilde{\Phi}^t)^T \end{bmatrix}$
$r_k = r_{k-1} - \alpha_k * P^T P p_{k-1} + \alpha_k(R + \sigma I)p_{k-1}$
$Px = i$
$\frac{\partial u}{\partial t} + u \cdot \nabla u = -\nabla p + \nu \Delta u + f_{ext}$
$u \cdot \hat{n} \geq 0$

In this chapter, we outline the fundamental theory of forward fluid simulation for visual effects, cover the theory behind PD and related schemes, explain OF, and outline the basics of CT.

3.1. Fluid Simulation

Fluids are substances that continually deform under applied shear stress, such as liquids and gases. Liquids have a defined interface where they usually meet another fluid like air. When simulating smoke plumes, we are truly simulating air with temperature differences and marker particles. For example, cigarette smoke is rising, as the blaze heats up the surrounding air. Hot air is less dense than cold air, which makes it lighter and therefore rise up due to buoyancy. As it happens, smoke particles are dissolved in the hot air making the hot air's motion visible.

There are two commonly established viewpoints when it comes to simulating fluids numerically: the Eulerian and the Lagrangian viewpoint. The Eulerian approach discretizes the simulation domain with a typically uniform grid and observes how different quantities, e.g., smoke density, temperature, and velocity, change over time at these discretization points. The Lagrangian viewpoint assumes that a multitude of particles represent the whole fluid volume and carries the quantities, which do not change over time. The most popular simulation methods with a purely Lagrangian viewpoint are Smoothed Particle Hydrodynamics (SPH) methods. In this thesis, we focus on Eulerian simulations, which are commonly used to simulate single-phase fluids as smoke. Bridson gives a thorough overview over popular fluid simulation methods for animation in [Bri15].

A fluid's motion is governed by the non-linear partial differential equations of second order: the famous NSE. For graphics applications, the goal is to generate visually plausible fluid animations implying that a human observer cannot identify the fluid's motion as physically incorrect. In engineering or physics applications, emphasis is placed on high physical accuracy. While fluids are compressible, compressibility usually takes place at microscopic levels, which is beyond the usual human perception, and is therefore insignificant for large-scale simulations of smoke or liquids. A fluid is incompressible if its velocity is divergence-free and therefore mass-preserving, meaning that there is no sink or source. The incompressible NSE are a widely established physical model for fluid simulation and are written as

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}_{\text{ext}} \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{3.1}$$

Incompressible Navier-Stokes equations.

where $\mathbf{u} = (u, v, w)$ is velocity, t is time, ∇ is the nabla operator, ρ is the physical density, p is pressure, ν is kinematic viscosity, and $\mathbf{f}_{\text{ext}}$ denotes external body forces. Note that the physical density is not the same as the fluid's density, such as smoke density. Smoke density is a quantity being transported throughout the fluid and can be rather interpreted as a marker of hot air. With the term *density*, we refer to smoke density and not to the physical density of the fluid if not stated otherwise.

The first equation is called momentum equation and is Newton's second law $ma = F$ [New99] in disguise. The left side is the material derivative $\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}$ describing the rate of change of some physical quantity, here velocity $\mathbf{u}$, in both space and time. Setting the material derivative to zero results in our advection equation displayed in Equation (3.2), meaning that the quantity moving around is not changing from the Lagrangian viewpoint. The right side of the momentum equation consists of inter-

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = 0 \tag{3.2}$$

Advection or transport equation.

nal forces, i.e., pressure and viscosity, and external forces, such as gravity. Those forces specify how the fluid accelerates. For example, high-pressure regions push towards low-pressure regions, which is expressed by the negative gradient of the pressure. For viscous fluids, particles prefer to move at the same speed as their neighbors, which is described by minimizing the differences of velocity between neighbors through the Laplacian operator $\Delta = \nabla^2$. The second equation in Equation (3.1) is the incompressibility condition where $\nabla \cdot \mathbf{u}$ denotes the divergence of the velocity field, which must be zero for incompressible fluids.

Viscosity is a measure of how much a fluid resists deformation, e.g., large-scale water has low viscosity and easily forms turbulent structures. Small-scale liquids or honey feature high viscosity where the latter is very difficult to stir. However, numerical simulation methods typically introduce an error term that can physically be interpreted as viscosity. Therefore, viscosity is often excessively present in the simulation resulting in not solving for the viscosity term explicitly. Omitting the viscosity term reduces the incompressible NSE to the incompressible Euler equations:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\frac{1}{\rho} \nabla p + \mathbf{f}_{\text{ext}} \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned} \tag{3.3}$$

Incompressible Euler equations.

Typical fluid simulators solve the NSE in Equation (3.1) for each time step t by splitting them into the components *advection*, *adding forces* (internal and external ones), and *pressure projection*. Each component is solved separately, one after the other. Such an approach is called operator splitting. First, an intermediate velocity field is computed through advection and adding forces. Then, the velocity field is made divergence-free and BCs are accounted for by the pressure projection. Note that quantities, such as smoke densities, should only ever be advected with an incompressible velocity field, i.e., after the pressure projection. Otherwise, mass-conservation is not guaranteed. An overview over typical steps in a forward fluid simulation is given in Figure 3.1, where Φ denotes the fluid density. With a given initial velocity field $\mathbf{u}^0$ at $t = 0$, velocity is advected forward, diffused, extended with external forces, and corrected with the pressure projection Π_{DIV}, which projects the velocity field onto the space of divergence-free velocity fields C_{DIV}. Considering smoke simulations, smoke density Φ is moved along with the velocity where, optionally, an inflow source is added first. We explain each of the simulator components and details about the discretization approach in the following.

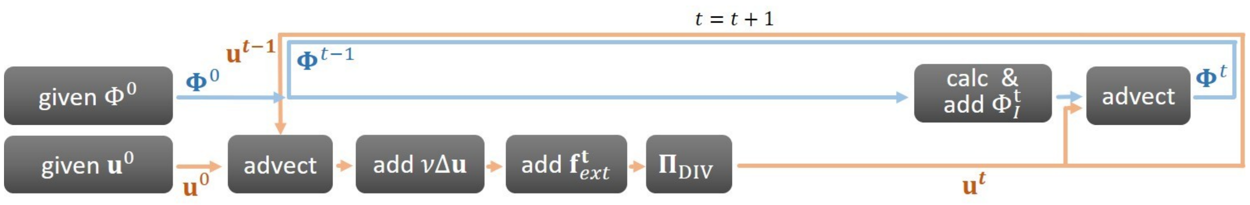


Figure 3.1.: Steps of a typical forward fluid simulator calculating velocity $\mathbf{u}^t$ and density Φ^t. The steps for velocity are denoted with orange arrows while blue stands for density. For both quantities, we start with their initial conditions $\mathbf{u}^0$ and Φ^0. For each time step t, velocity is advected forward, viscosity and external forces are added, and the velocity is projected to be divergence-free by the pressure solver. Density is evolved by determining boundary conditions through adding an inflow source and advecting it forward with the newly computed velocity field. Hence, the main steps are initial and boundary conditions, advection, viscosity, external forces, and pressure projection.

Advection The advection, convection, or transport equation in Equation (3.2) could be solved with forward Euler, which is, however, unstable in this case. Therefore, semi-Lagrangian advection [Sta99] is typically used, which is unconditionally stable and allows for large time steps. Though it is performed on an Eulerian grid, a Lagrangian viewpoint is taken, which leads to the name *semi-Lagrangian*. The new value of a centrally stored quantity q at location x_a is updated by first accessing the velocity $\mathbf{u}$ at x_a. Then, this velocity is traced backwards to a grid point x_b. The value of quantity q is interpolated at x_b and stored as new value for quantity q at x_a. A schematic overview over the back tracing and – here bilinear – interpolation is shown in Figure 3.2.

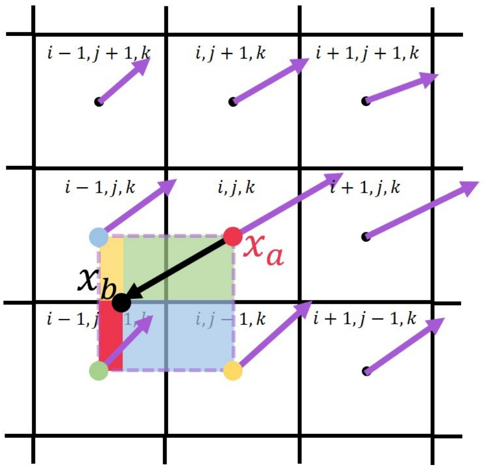


Figure 3.2.: Semi-Lagrangian advection scheme by tracing a position x_a backwards to position x_b, here denoted with a black arrow. The new value of a centrally stored quantity q at x_a is found by interpolating the surrounding data points of q at position x_b. The corresponding bilinear weights for each of the four neighbors are visualized with colored rectangles.

Note that tracing the velocity backwards is a linearization of the advection equation, as velocity is assumed to be piecewise constant. The semi-Lagrangian advection scheme is unconditionally stable, as the solution is bounded by existing values, e.g., through linear interpolation. However, it is only first-order accurate in time and if the time step size is increased, errors will increase. As interpolation operations take place in each advection step, sharp features are smoothed out and blurred, which is also known as dissipation. This kind of numerical dissipation is a viscosity-like term and referred to as *numerical viscosity*. Dissipation will still occur if higher-order spatial interpolation is applied and is accumulated over time steps. When using very small time steps, the sum of interpolation errors will increase. Therefore, it is important to choose an appropriate time step size where discretization and interpolation errors balance out.

Often, other quantities are advected within the velocity field. For example, when considering smoke simulations, smoke density Φ needs to be moved along the velocity field. In order to do so, the following advection equation must be solved: $\frac{\partial \Phi}{\partial t} + \mathbf{u} \cdot \nabla \Phi = 0$. Instead of solving the self-advection equation from Equation (3.2), we advect density and as such, use both temporal and spatial derivation of the density instead of the velocity.

Forces External body forces are simply added by updating the velocity with a forward Euler step. In order to incorporate viscosity in special cases where the fluid is too turbulent otherwise, the diffusion equation is typically solved with a regular CG solver:

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \Delta \mathbf{u}, \tag{3.4}$$

Diffusion equation for viscosity.

where the Laplacian operator Δ is the divergence of the gradient of the velocity, i.e., $\Delta = \nabla \cdot \nabla = \nabla^2$. It measures how far a quantity differs from the average around it. The diffusion equation is more difficult to solve at the surface where not every neighbor is available and for variable viscosity, which we will not discuss here.

Pressure Projection We interpret the pressure as the quantity that makes the velocity field divergence-free and ensures that both solid-wall and free-surface BCs are met. For example, a fluid must not flow into or out of a solid obstacle. The intermediate velocity $\mathbf{u}^*$, which is the resulting velocity field after advection and adding external and viscous forces, is corrected accordingly. In order to do so, the pressure gradient is subtracted from each velocity component. For example, the x-component of the velocity is corrected as following:

$$u_{i,j,k} = u^*_{i,j,k} - \frac{1}{\rho}(p_{i+1,j,k} - p_{i,j,k}), \tag{3.5}$$

Velocity correction through the pressure gradient, assuming a staggered MAC grid as in Figure 3.3. This choice of discretization is explained later in the paragraph *Discretization*.

assuming the grid spacing Δx to be equal to 1, see the paragraph about discretization below. In order to find the pressure values for which both incompressibility and BCs constraints hold, the following Poisson equation is solved:

$$\Delta p = -\frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}. \tag{3.6}$$

Poisson equation for solving the pressure.

This Poisson equation transforms into a system of linear equations $Ap = b$ with the negative divergence of the velocity being the right-hand-side b and the Laplacian as the system's matrix A.

A must be adapted such that both solid-wall and free-surface BCs hold. The free-surface BCs are a Dirichlet BC where the value of the solution is specified. The pressure in surrounding air cells is commonly set to be equal to zero: $p_{\text{air}} = 0$. The reason is that air is 700 times lighter than water or a comparable fluid and, therefore, has no significant impact on that fluid. Air is treated as region of constant atmospheric pressure, which can be set to zero, because only differences in pressure values matter.

In order to prevent fluid from flowing into or out of solid walls, the velocity relative to the solid at fluid-solid interfaces is set to zero. Either the whole velocity or only the normal component of the velocity is set to zero: $\hat{\mathbf{n}} \cdot (\mathbf{u} - \mathbf{u}_{\text{solid}}) = 0$. No-slip boundary conditions are typically used for viscid fluids and constrain all components of the velocity to be zero, which results in fluid not being able to move once meeting a wall. The no-stick or slip BCs only constrain the normal component of the velocity, hence allow tangential movement. The pressure solver does not change these velocities and hence, the pressure gradient that corrects the velocity must be zero: $\nabla p = 0$. Specifying the value of a derivative of the solution is a Neumann BC. Both BCs transform into linear constraints on the pressure and can easily be included in the system's matrix A.

A is a large, sparse, and symmetric coefficient matrix with one row per cell, which is mostly zero except for the possibly seven non-zero Laplacian entries (in 3D). As A is symmetric, only half of the coefficients need to be stored. After setting up A and b, the system is conventionally solved with a preconditioned CG solver, which is robust, easy to implement, and able to handle complex domain shapes. The CG solver is an iterative method that does not require an explicit representation of the system's matrix and is guaranteed to converge. Each iteration consists of only basic computations. However, the run time of a CG solver scales poorly with grid resolution and therefore features high computational costs for higher grid resolutions.

After solving for pressure values, the intermediate velocity field is corrected by the negative pressure gradient, see Equation (3.5). This pressure correction is also called pressure projection, because the velocity field is projected onto the space of divergence-free velocity fields C_{DIV}. Projecting it again leaves the velocity field unchanged. The divergence-free components of the flow are left untouched by the pressure solver due to the orthogonality of the curl-free and divergence-free components. The pressure projection is hence an orthogonal projection [Cho68, PB*14] only removing divergent parts of the velocity and changing the velocity as few as necessary.

Discretization Discretizing continuous partial differential equations for solving them numerically influences numerical stability substantially. Time and space need to be discretized. Time is simply discretized with forward Euler: $t_n = n\Delta t$for $n = 0, 1, ..., N$ and $\frac{\partial \mathbf{u}}{\partial t} = \frac{\mathbf{u}^{t_{n+1}} - \mathbf{u}^{t_n}}{\Delta t}$.

As we take an Eulerian approach, we discretize our spatial simulation domain into a uniform Cartesian grid resulting in a finite number of discrete points where each cell comes with indices i, j, k. In order to approximate partial derivatives discretely, we use finite differences between our discrete grid points. There are three commonly used differences: forward, backward, and central differences. Central differences are the most accurate ones, because their truncation error is on the order of $O((\Delta x)^2)$ instead of $O(\Delta x)$, where Δx denotes the (uniform) grid spacing and $\Delta x \to 0$. For example, finite difference approximations for the first-order derivative of $\mathbf{u}$ in the spatial $x-$direction is given by the following equations (considering a 2D domain):

$$\left(\frac{\partial \mathbf{u}}{\partial x}\right)_{i,j} = \begin{cases} \frac{\mathbf{u}_{i+1,j}-\mathbf{u}_{i,j}}{\Delta x} + O(\Delta x), & \text{forward difference} \\ \frac{\mathbf{u}_{i,j}-\mathbf{u}_{i-1,j}}{\Delta x} + O(\Delta x), & \text{backward difference} \\ \frac{\mathbf{u}_{i+1,j}-\mathbf{u}_{i-1,j}}{2\Delta x} + O((\Delta x)^2), & \text{central difference.} \end{cases} \tag{3.7}$$

Forward, backward, and central differences for the derivative of **u** in the spatial x—direction.

The basis for finite differences is the Taylor expansion:

$$f(x+\Delta x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x)}{k!}(\Delta x)^k. \tag{3.8}$$

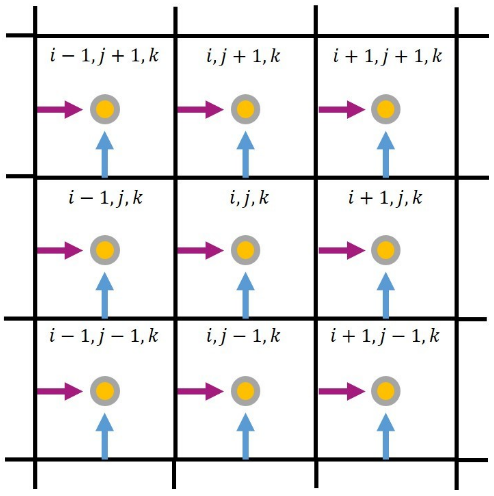
Taylor expansion of an infinitely differentiable function f around point x.

Applied to central differences for the derivative of **u** in the x—direction results in:

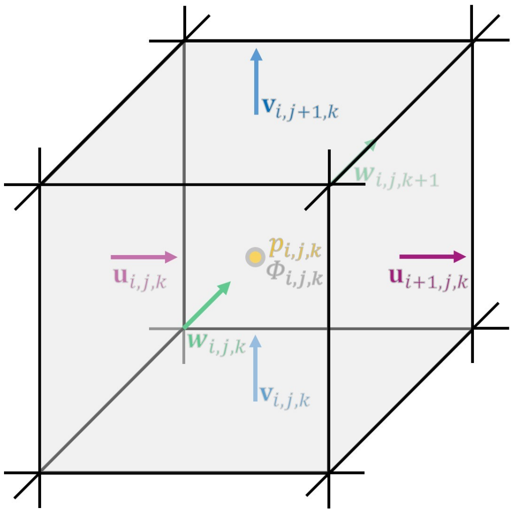
$$\begin{aligned} \mathbf{u}_{i+1,j} &= \mathbf{u}_{i,j} + \Delta x \frac{\partial \mathbf{u}_{i,j}}{\partial x} + \frac{(\Delta x)^2}{2}\frac{\partial^2 \mathbf{u}_{i,j}}{\partial x^2} + O((\Delta x)^3) \\ \mathbf{u}_{i-1,j} &= \mathbf{u}_{i,j} - \Delta x \frac{\partial \mathbf{u}_{i,j}}{\partial x} + \frac{(\Delta x)^2}{2}\frac{\partial^2 \mathbf{u}_{i,j}}{\partial x^2} + O((\Delta x)^3) \\ \implies \quad \mathbf{u}_{i+1,j} - \mathbf{u}_{i-1,j} &= 2h\frac{\partial \mathbf{u}_{i,j}}{\partial x} + O((\Delta x)^3) \\ \implies \quad \frac{\partial \mathbf{u}_{i,j}}{\partial x} &= \frac{\mathbf{u}_{i+1,j} - \mathbf{u}_{i-1,j}}{2h} + O((\Delta x)^2). \end{aligned} \tag{3.9}$$

Finite differences converge to the true value of $\frac{\partial \mathbf{u}}{\partial x}$ for $\Delta x \to 0$.

However, as central differences do not take the current value $\mathbf{u}_{i,j}$ into account, common central differences have a non-trivial null-space. The set of functions where the central differences evaluate to zero includes not only constant but also oscillating functions. This is especially unfavorable in the pressure projection, where the pressure gradient is used to correct the velocity components, and where the divergence of the velocity must be calculated for the center of each cell. Pressure solvers with regular, co-located discretization of velocity and pressure tend to produce checkerboard patterns. Furthermore, highly divergent velocity fields could appear to have zero divergence and are hence not corrected by the pressure solver. Such high-frequency oscillations could persist during simulation or even grow unstably. Therefore, Harlow and Welch [HW65] developed the MAC grid, as this staggered discretization enables the use of central differences without the non-trivial null-space issue. Velocity components and pressure values are stored at different locations in the grid, namely in the cell center and the center of cell faces as shown in Figure 3.3. Hence, the pressure gradient and the divergence of the velocity can be approximated by central differences with trivial null-spaces, i.e., only constant functions yield zero derivatives. The staggered grid approach is also useful for viscosity calculations.



(a) Nine interior cells of a 2D MAC grid with scalar and vector-valued components.



(b) One cell of a 3D MAC grid with its central scalar quantities and its velocity components at adjacent faces.

Figure 3.3.: 2D and 3D MAC grid with cell indices i, j, k, velocity components $\mathbf{u} = (u, v, w)$ in purple, blue, and green, pressure p in yellow, and density Φ in gray. To increase the accuracy of finite difference approximations of velocity, density, or pressure derivatives, the quantities are stored at different locations in the grid. Scalar quantities are stored in the cell center while vector-valued quantities are positioned at the centers of cell faces.

Fluid Simulator We use the open-source framework *mantaflow* [TP19] in order to simulate smoke or liquid. *mantaflow* contains Eulerian simulations with MAC grids, a preconditioned CG pressure solver, MacCormack advection, FLIP simulations for liquids, and many more fluid simulation techniques. It is written in C++ and comes with the possibility of python scripting to facilitate fast prototyping.

3.2. Convex Optimization

The advantage of convex over non-convex problems is that, generally, a global optimum can be computed within reasonable time, good precision, and independent of initialization. Typically, convex optimization problems take the following form:

$$\underset{\mathbf{x}}{\text{minimize}} \quad h(\mathbf{x}), \tag{3.10}$$

where h is a convex function, called objective function. The objective function might be non-smooth or even discontinuous. If h is a complex function, it is often easier to solve it through an iterative divide-and-conquer approach. Let us say h is the sum of two simpler, more manageable, closed, proper, and convex functions f and g: $h = f + g$. Then, optimization can be performed separately for f and g, which is simpler than optimizing h directly. Depending on the form of f and g, mathematical structures can be exploited, which leads to very efficient solvers. The split optimization problem takes the form

$$\begin{aligned} \underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{x} = \mathbf{z}, \end{aligned} \tag{3.11}$$

where the variable $\mathbf{x}$ is split into two parts: $\mathbf{x}$ and $\mathbf{z}$. Such divide-and-conquer approaches as we here refer to are called *proximal methods*. In general, the class of optimization algorithms called *proximal methods* [PB*14] splits difficult multi-objective optimization schemes into simpler sub-problems by making use of *proximal operators*, which are application specific solvers targeting smaller convex optimization sub-problems. Here, we have two proximal operators, i.e., solvers, for each sub-problem f and g, which are defined as following:

$$\begin{aligned} \mathbf{prox}_{f,\sigma}(\xi) &:= \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) + \frac{\sigma}{2} \|\mathbf{x} - \xi\|^2 \right) \\ \mathbf{prox}_{g,\sigma}(\xi) &:= \underset{\mathbf{z}}{\operatorname{argmin}} \left(g(\mathbf{z}) + \frac{\sigma}{2} \|\mathbf{z} - \xi\|^2 \right), \end{aligned} \tag{3.12}$$

with σ denoting the penalty weight, ξ being a generic input variable, and $\|.\|$ being the l2-norm. The convex functions f and g can be either quantities to minimize or hard constraints, which would translate to indicator functions. Both proximal operators are applied iteratively until convergence is reached and the solution in $\mathbf{z}$ minimizes h. Depending on the specific instance of algorithm for solving Equation (3.11), the stopping criterion is either related to the amount of change in the $\mathbf{z}$ or to the error in the first function f. The iterations are stopped when either error measurement or both fall below a threshold ϵ.

It would also be possible to solve a convex quadratic function h restricted to an affine set as Karush-Kuhn-Tucker (KKT) system [BPC*11]. However, the composed KKT matrix can usually not be solved by simply applying a preconditioned CG solver, as it is most likely not positive-definite and poorly conditioned. While there exist several extensions for simplifying the solving of such a system and other solving strategies, such as the Schur Complement Reduction, we focus here on iterative divide-and-conquer approaches splitting complex problems into simpler sub-problems.

In order to solve the optimization problem in Equation (3.11), we discuss three proximal methods: PD, ADMM, and IOP. Although we will mostly use PD in this thesis, we start by introducing ADMM, which is a simpler version of PD. Before that, we quickly review common forms of proximal operators: quadratic and indicator functions. An example for an indicator function is the pressure projection from fluid simulators.

Quadratic Functions Often, the convex functions to minimize are quadratic functions. Here is an exemplary definition for f:

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T A\mathbf{x} + b\mathbf{x} + r, \tag{3.13}$$

where A is symmetric, positive semi-definite, and square, b is the right-hand side, and r is a constant term with respect to $\mathbf{x}$. If $A + \sigma I$ is invertible, the corresponding proximal operator is given by

$$\mathbf{prox}_{f,\sigma}(\xi) = (A + \sigma I)^{-1}(\sigma\xi - b), \tag{3.14}$$

where σ is the penalty weight keeping the solution of $\mathbf{prox}_{f,\sigma}$ close to the input variable ξ.

Optimization Problems in Fluid Simulation As we target optimization problems for fluid simulation, we are often dealing with constraints on the velocity field. The velocity field must always be divergence-free in our application scenarios. Therefore, our optimization problems take on the following form:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in C_{\text{DIV}}, \end{aligned} \tag{3.15}$$

where $\mathbf{x}$ is the velocity field to be optimized and C_{DIV} denotes the space of divergence-free velocity fields. Here, g is an indicator function of the closed non-empty convex set C_{DIV} and an orthogonal projection, as shown in Section 3.1. As explained in [GKHH12], the proximal operator for making a velocity field divergence-free is equal to the pressure projection, which projects the velocity field onto the space of divergence-free velocity fields. Therefore, the proximal operator for g is the pressure projection itself:

$$\mathbf{prox}_{g,\sigma}(\xi) = \Pi_{\text{DIV}}(\xi), \tag{3.16}$$

where Π_{DIV} stands for a projection onto C_{DIV} with a commonly applied Poisson pressure solver.

Such optimization schemes can easily be incorporated into fluid simulators by replacing the call to the pressure projection with a call to the algorithm solving the optimization problem in Equation (3.15). However, as a pressure solver based on the CG scheme is typically quite expensive and proximal operators are applied iteratively, it is beneficial to employ an adaptive accuracy scheme. Here, the accuracy of the CG solver ϵ_{CG} is set to a high threshold, e.g., $\epsilon_{\text{CG}} = 10^{-2}$, for the first iterations and is adaptively decreased until the desired final accuracy is reached, e.g., $\epsilon_{\text{CG}} = 10^{-5}$. With this adaptive CG scheme, the number of required CG iterations is decreased drastically.

3.2.1. Alternating Direction Methods of Multipliers

ADMM, the Alternating Direction Method of Multipliers [BPC*11, GOSB14], was developed in the 1970s. It profits from strong convergence properties of the method of multipliers and from decomposability properties of the dual ascent method, and is a special case of the Douglas–Rachford splitting. ADMM solves the optimization problem Equation (3.11) with the following iterative variable updates:

$$\begin{aligned} \mathbf{x}^{k+1} &:= \mathbf{prox}_{f,\rho}(\mathbf{z}^k - \mathbf{y}^k) \\ \mathbf{z}^{k+1} &:= \mathbf{prox}_{g,\rho}(\mathbf{x}^{k+1} + \mathbf{y}^k) \\ \mathbf{y}^{k+1} &:= \mathbf{y}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1}, \end{aligned} \tag{3.17}$$

where ρ is the parameter for controlling ADMM's convergence, $\mathbf{y}$ is a dual variable, $\mathbf{x}$ and $\mathbf{z}$ are slack variables, and k is the iteration counter. The algorithm alternates between updating $\mathbf{x}$ and $\mathbf{z}$, which accounts for the alternating direction in the name ADMM. For convergence, the dual variable $\mathbf{y}$ approaches an optimal value and $\mathbf{z}$ holds the solution to Equation (3.11).

Convergence is reached if both primal and dual residual r and s lie below a threshold:

$$\left\|r^{k+1}\right\|_2 = \left\|\rho(\mathbf{z}^k - \mathbf{z}^{k+1})\right\|_2 \leq \epsilon^{\text{dual}}, \qquad \epsilon^{\text{dual}} = \sqrt{n_{\text{dim}}}\,\epsilon^{\text{abs}} + \epsilon^{\text{rel}}\max\left(\left\|\mathbf{y}^{k+1}\right\|\right) \tag{3.18}$$

$$\left\|s^{k+1}\right\|_2 = \left\|\mathbf{x}^{k+1} - \mathbf{z}^{k+1}\right\|_2 \leq \epsilon^{\text{pri}}, \qquad \epsilon^{\text{pri}} = \sqrt{n_{\text{dim}}}\,\epsilon^{\text{abs}} + \epsilon^{\text{rel}}\max\left(\left\|\mathbf{x}^{k+1}\right\|, \left\|\mathbf{z}^{k+1}\right\|\right), \tag{3.19}$$

with the primal and dual tolerances ϵ^{pri} and ϵ^{dual}, n_{dim} as the dimension of the residuals, l2-norm $||.||$, and the absolute and relative stopping criteria ϵ^{abs} and ϵ^{rel}. The relative criterion is typically set to $\epsilon^{\text{rel}} = 10^{-3}$ while ϵ^{abs} depends on the scale of the typical variable values, also usually set to $\epsilon^{\text{abs}} = 10^{-3}$. ADMM may require a high number of iterations for high accuracy. For modest accuracy, it is supposed to converge within a few tens of iterations [WBAW12].

3.2.2. Fast Primal-Dual Method

With PD, we are able to solve a slightly more general problem [PCBC09]:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \quad & f(K\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{x} = \mathbf{z}, \end{aligned} \tag{3.20}$$

for some linear operator K. In this thesis, K is simply the identity: $K = I$. The variable updates for PD are given by

$$\begin{aligned} \mathbf{x}^{k+1} &:= \mathbf{prox}_{f*,1/\sigma}(\mathbf{x}^k + \sigma\mathbf{y}^k) \\ \mathbf{z}^{k+1} &:= \mathbf{prox}_{g,1/\tau}(\mathbf{z}^k - \tau\mathbf{x}^{k+1}) \\ \mathbf{y}^{k+1} &:= \mathbf{z}^{k+1} + \theta(\mathbf{z}^{k+1} - \mathbf{z}^k), \end{aligned} \tag{3.21}$$

where the parameters $\{\sigma, \tau, \theta\}$ affect convergence, k is the iteration counter, and f^* is the convex conjugate of f. While $\mathbf{z}$ converges to the solution of Equation (3.11), $\mathbf{x}$ and $\mathbf{y}$ are helper variables. Through Moreau's identity, the $\mathbf{x}$-update can be transformed such that is not necessary to compute f^* directly:

$$\mathbf{prox}_{f^*,1/\sigma}(\xi) = \xi - \sigma\,\mathbf{prox}_{f,\sigma}(\xi/\sigma). \tag{3.22}$$

Thus, the final variable updates are reduced to

$$\begin{aligned} \mathbf{x}^{k+1} &:= \mathbf{x}^k + \sigma\mathbf{y}^k - \sigma\,\mathbf{prox}_{f,\sigma}(\tfrac{1}{\sigma}\mathbf{x}^k + \mathbf{y}^k) \\ \mathbf{z}^{k+1} &:= \mathbf{prox}_{g,1/\tau}(\mathbf{z}^k - \tau\mathbf{x}^{k+1}) \\ \mathbf{y}^{k+1} &:= \mathbf{z}^{k+1} + \theta(\mathbf{z}^{k+1} - \mathbf{z}^k). \end{aligned} \tag{3.23}$$

PD is said to have converged if the change in the solution $\left\|\mathbf{z}^{k+1}-\mathbf{z}^{k}\right\|$ gets lower than a specified threshold ϵ. As for ADMM, we make use of an absolute and relative criterion ϵ^{abs} and ϵ^{rel}. Both are typically set to $\epsilon^{\text{rel}} = \epsilon^{\text{abs}} = 10^{-3}$, where the absolute threshold depends on the scale of the solution values. In Algorithm 1, we summarize a pseudocode implementation for applying PD for a fluid velocity optimization problem with an application-dependent f and divergence-free constraint g.

Algorithm 1 Our PD-based method for fluid simulation

```
 1: procedure PD($\tau$, $\sigma$, $\theta$)
 2:     while $k < maxIters$ do
 3:         $\mathbf{x}^{k+1} = \mathbf{x}^k + \sigma\mathbf{y}^k - \sigma\text{proxF}(\sigma, \frac{1}{\sigma}\mathbf{x}^k + \mathbf{y}^k)$ // x-update
 4:         $\mathbf{z}^{k+1} = \Pi_{\text{DIV}}(\mathbf{z}^k - \tau\mathbf{x}^{k+1})$ // z-update (using adaptive CG accuracy)
 5:         $\mathbf{y}^{k+1} = \mathbf{z}^{k+1} + \theta(\mathbf{z}^{k+1} - \mathbf{z}^k)$ // y-update
 6:         // check stopping criterion
 7:         $r^{k+1} = \mathbf{z}^{k+1} - \mathbf{z}^k$
 8:         $\epsilon = \sqrt{n_{\text{dim}}}\epsilon^{\text{abs}} + \epsilon^{\text{rel}}\left\|\mathbf{z}^{k+1}\right\|$
 9:         if $\left(\left\|r^{k+1}\right\| \leq \epsilon\right)$ then break
10:     end while
11:     return z
12: end procedure
```

An in-depth discussion of PD can be found in [CP11]. The authors state that PD features optimal convergence properties for the class of non-smooth convex problems. Therefore, with a minimal increase of computational costs, i.e., more vector addition and multiplication, which is often negligible compared to the cost of proximal operators, we may achieve faster convergence compared to ADMM. We demonstrate that PD outperforms ADMM for both fluid guiding and solid-wall BCs in Chapters 7 and 8. However, for a special setting of parameters, PD reduces to ADMM as shown in the following.

ADMM to PD ADMM has only one parameter that controls convergence, i.e., ρ. If K is set to the identity and the convergence parameters of PD and ADMM are set to 1 ($\sigma = \tau = \theta = \rho = 1$), PD reduces to ADMM. The variable updates for ADMM are given by

$$\textbf{I.}_{\text{ADMM}} \qquad \mathbf{x}^{k+1}_{\text{ADMM}} = \mathbf{prox}_f(\mathbf{z}^k - \mathbf{y}^k_{\text{ADMM}}) \tag{3.24}$$

$$\textbf{II.}_{\text{ADMM}} \qquad \mathbf{z}^{k+1} = \mathbf{prox}_g(\mathbf{x}^{k+1}_{\text{ADMM}} + \mathbf{y}^k_{\text{ADMM}}) \tag{3.25}$$

$$\textbf{III.}_{\text{ADMM}} \qquad \mathbf{y}^{k+1}_{\text{ADMM}} = \mathbf{y}^k_{\text{ADMM}} + \mathbf{x}^{k+1}_{\text{ADMM}} - \mathbf{z}^{k+1}, \tag{3.26}$$

while the updates for PD are given by

$$\textbf{I.}_{\text{PD}} \qquad \mathbf{x}^{k+1}_{\text{PD}} = \mathbf{x}^k_{\text{PD}} + \mathbf{y}^k_{\text{PD}} - \mathbf{prox}_f(\mathbf{x}^k_{\text{PD}} + \mathbf{y}^k_{\text{PD}}) \tag{3.27}$$

$$\textbf{II.}_{\text{PD}} \qquad \mathbf{z}^{k+1} = \mathbf{prox}_g(\mathbf{z}^k - \mathbf{x}^{k+1}_{\text{PD}}) \tag{3.28}$$

$$\textbf{III.}_{\text{PD}} \qquad \mathbf{y}^{k+1}_{\text{PD}} = 2\mathbf{z}^{k+1} - \mathbf{z}^k. \tag{3.29}$$

Now, we reduce PD to ADMM, i.e., transform the PD updates in Equations (3.27), (3.28), (3.29) to the ADMM updates in Equations (3.24), (3.25), (3.26). As $\mathbf{z}^{k+1}$ converges to the solution of Equation (3.11) for both update schemes, $\mathbf{z}^{k+1}$ must be equivalent for both schemes and therefore the following must hold:

$$\begin{aligned}
&\textbf{II.}_{\text{PD}} && \mathbf{z}^{k+1} = \mathbf{prox}_g(\mathbf{z}^k - \mathbf{x}_{\text{PD}}^{k+1}) \\
&\textbf{II.}_{\text{ADMM}} && \phantom{\mathbf{z}^{k+1}} = \mathbf{prox}_g(\mathbf{x}_{\text{ADMM}}^{k+1} + \mathbf{y}_{\text{ADMM}}^{k})
\end{aligned} \tag{3.30}$$

$$\Rightarrow \quad \mathbf{x}_{\text{ADMM}}^{k+1} + \mathbf{y}_{\text{ADMM}}^{k} = \mathbf{z}^k - \mathbf{x}_{\text{PD}}^{k+1}$$

$$\Rightarrow \quad \mathbf{x}_{\text{PD}}^{k+1} = -\mathbf{x}_{\text{ADMM}}^{k+1} - \mathbf{y}_{\text{ADMM}}^{k} + \mathbf{z}^k \tag{3.31}$$

$$\Rightarrow \quad \mathbf{x}_{\text{PD}}^{k} = -\mathbf{x}_{\text{ADMM}}^{k} - \mathbf{y}_{\text{ADMM}}^{k-1} + \mathbf{z}^{k-1} \tag{3.32}$$

However, the two helper variables $\mathbf{x}$ and $\mathbf{y}$ take different roles for PD and ADMM. In order to connect $\mathbf{y}_{\text{ADMM}}^{k+1}$ with $\mathbf{y}_{\text{ADMM}}^{k}, \mathbf{x}_{\text{ADMM}}^{k+1}$, and $\mathbf{z}^{k+1}$, we add Equation (3.26) to the variable updates of PD:

$$\textbf{III.}_{\text{ADMM}} \quad \mathbf{y}_{\text{ADMM}}^{k+1} = \mathbf{y}_{\text{ADMM}}^{k} + \mathbf{x}_{\text{ADMM}}^{k+1} - \mathbf{z}^{k+1}. \tag{3.33}$$

Then, we use both Equation (3.27) and Equation (3.29) to arrive at Equation (3.24). The update for $\mathbf{x}$ reduces as following:

$$\begin{aligned}
\textbf{I.}_{\text{PD}} \quad \mathbf{x}_{\text{PD}}^{k+1} &= \mathbf{x}_{\text{PD}}^{k} + \mathbf{y}_{\text{PD}}^{k} - \mathbf{prox}_f(\mathbf{x}_{\text{PD}}^{k} + \mathbf{y}_{\text{PD}}^{k}) \quad | \triangleright \text{ use Eq. (3.31), (3.32), (3.29), simplify} \\
-\mathbf{x}_{\text{ADMM}}^{k+1} - \mathbf{y}_{\text{ADMM}}^{k} + \mathbf{z}^k &= -\mathbf{x}_{\text{ADMM}}^{k} - \mathbf{y}_{\text{ADMM}}^{k-1} + 2\mathbf{z}^k - \mathbf{prox}_f(-\mathbf{x}_{\text{ADMM}}^{k} - \mathbf{y}_{\text{ADMM}}^{k-1} + 2\mathbf{z}^k) \quad | \triangleright \text{ use Eq. (3.26)} \\
-\mathbf{x}_{\text{ADMM}}^{k+1} - \mathbf{y}_{\text{ADMM}}^{k} + \mathbf{z}^k &= -\mathbf{y}_{\text{ADMM}}^{k} + \mathbf{z}^k - \mathbf{prox}_f(-\mathbf{y}_{\text{ADMM}}^{k} + \mathbf{z}^k) \quad | \triangleright \text{ simplify} \\
-\mathbf{x}_{\text{ADMM}}^{k+1} &= -\mathbf{prox}_f(-\mathbf{y}_{\text{ADMM}}^{k} + \mathbf{z}^k) \quad | \triangleright \text{ invert, rearrange} \\
\textbf{I.}_{\text{ADMM}} \quad \mathbf{x}_{\text{ADMM}}^{k+1} &= \mathbf{prox}_f(\mathbf{z}^k - \mathbf{y}_{\text{ADMM}}^{k})
\end{aligned} \tag{3.34}$$

We showed that the PD variable updates $\textbf{I.}_{\text{PD}}, \textbf{II.}_{\text{PD}}, \textbf{III.}_{\text{PD}}$ reduce to the ADMM variable updates $\textbf{I.}_{\text{ADMM}}, \textbf{II.}_{\text{ADMM}}, \textbf{III.}_{\text{ADMM}}$ if $K = 1$ and $\sigma = \tau = \theta = \rho = 1$. We assumed the equivalence of both $\mathbf{z}$-variables and added Equation (3.26) to PD's system. The derived ADMM updates are given in Equations (3.34), (3.30), (3.33). For more appropriate choices of the convergence parameters, i.e., $\sigma, \tau, \theta \neq 1$, optimal control over convergence is obtained.

3.2.3. Iterated Orthogonal Projection

IOP, the Iterated Orthogonal Projection method [MCPN08], is similar to von Neumann's alternating projections [BPC*11] and is suitable for solving Equation (3.11) if both proximal operators are orthogonal projections. Both operators are applied iteratively until convergence is reached:

$$\mathbf{x}^{k+1} = \Pi_f(\mathbf{z}^k) \tag{3.35}$$

$$\mathbf{z}^{k+1} = \Pi_g(\mathbf{x}^{k+1}). \tag{3.36}$$

Fast convergence is not guaranteed, as this iterative scheme converges slowly if the individual subspaces are almost parallel. In practice, usually a limited number is sufficient for a satisfactory result. Iterations are stopped once the change in the solution or the error in the first proximal operator are small enough, e.g., $\left\|\mathbf{z}^{k+1} - \mathbf{z}^k\right\| \leq \epsilon$. In order to improve convergence, a Krylov method can be applied [MCPN08], as outlined in pseudocode in Algorithm 2. Algorithm 2 is performed after the two projections in Equations (3.35) and (3.36).

Algorithm 2 Krylov method speeding up convergence

```
1:  procedure KRYLOV($\mathbf{z}^k$, $\mathbf{z}^{k-1}$, $k$, $\epsilon^{k-1}$)
2:      $\epsilon^k = \text{error}(\mathbf{z}^k)$
3:      if $k > 1$ then
4:          $\mathbf{z}_{\epsilon_\Delta} = \mathbf{z}^k - \mathbf{z}^{k-1}$ // correction vector
5:          $\epsilon_{\text{ratio}} = \epsilon^k / \epsilon^{k-1}$
6:          $\mathbf{z}_{\epsilon_{\text{tmp}}} = \mathbf{z}^k - \epsilon_{\text{ratio}} \mathbf{z}_{\epsilon_\Delta}$
7:          $\epsilon_{\epsilon_{\text{tmp}}} = \text{error}(\mathbf{z}_{\epsilon_{\text{tmp}}})$
8:          if $\epsilon^{\epsilon_{\text{tmp}}} < \epsilon^k$ then $\mathbf{z}^k = \mathbf{z}_{\epsilon_{\text{tmp}}}$
9:      end if
10: end procedure
```

3.3. Optical Flow

In this thesis, we reconstruct velocity based on the global, divergence-free OF approach by Gregson et al. [GITH14]. Given two input density fields Φ^{t-1} and Φ^t, a velocity field $\mathbf{u}^t$ is calculated transporting Φ^{t-1} to Φ^t, also referred to as ensuring temporal coherence between density fields of different time steps. In order to estimate such a velocity field, the brightness constancy assumption is minimized, which is equal to the fluid transport or advection equation, see Equation (3.2). However, this equation does not fully determine one single 3D velocity field but is severely under-constrained such that many velocity fields are valid solutions. To alleviate under-determination, two regularizers are added: a smoothness term constraining each velocity vector not to differ from its neighbors drastically and a kinetic penalty. Typically, a Tikhonov regularizer is employed as kinetic penalty in order to favor minimally energetic flows, i.e., preferring small motion displacements over large displacements. As velocity fields are assumed incompressible for fluid animation purposes, the velocity field is additionally constrained to be divergence-free:

$$\begin{aligned}\underset{\mathbf{u}^t}{\text{minimize}} \quad & \left\| \frac{\Phi^t - \Phi^{t-1}}{\Delta t} + \nabla\Phi^t \cdot \mathbf{u}^t \right\|^2 + \alpha \left\| \nabla \mathbf{u}^t \right\|^2 + \beta \left\| \mathbf{u}^t \right\|^2 \\ \text{subject to} \quad & \nabla \cdot \mathbf{u}^t = 0,\end{aligned} \tag{3.37}$$

Divergence-free optical flow [GITH14].

where $\mathbf{u}^t$ and Φ^t are the velocity and density field at time t, Δt is the time step, $||.||$ is the l2-norm, and α and β are the weights determining the amount of smoothness and kinetic penalty, respectively. Standard OF implementations make use of a co-located discretization instead of a staggered MAC grid, meaning that all quantities are stored in the centers of the cells. However, the divergence-free constraint on the velocity is commonly enforced by a pressure solver, see Section 3.1, which performs best on a staggered discretization. To counteract checkerboard artifacts, Gregson et al. [GITH14] employed a pressure solver where pressure values are stored on cell vertices on a uniform rectilinear grid. Unfortunately, twenty-seven non-zero entries need to be stored instead of seven, increasing the memory footprint drastically.

PD Formulation of Incompressible Optical Flow The proximal operators for a PD implementation of divergence-free OF are inserted into the PD updates in Equation (3.23). Minimizing the sum of the norm of the brightness constancy assumption and the regularizers translates into a quadratic function like Equation (3.13) where the corresponding proximal operator is defined in Equation (3.14). The proximal operator for the indicator function making a velocity field divergence-free is shown in Equation (3.16). Therefore, the proximal operators for a divergence-free OF are:

$$\begin{aligned}\mathbf{prox}_{f,\sigma}(\xi) &= (A + \sigma I)^{-1}(\sigma\xi - b) \\ \mathbf{prox}_{g,1/\tau}(\xi) &= \Pi_{\mathrm{DIV}}(\xi),\end{aligned} \tag{3.38}$$

where f, A, and b are defined as follows:

$$\begin{aligned}f(\mathbf{u}^t) &= \tfrac{1}{2}{\mathbf{u}^t}^T A \mathbf{u}^t + b\mathbf{u}^t + r \\ A &= (\nabla\Phi^t)(\nabla\Phi^t)^T + \alpha\textstyle\sum_j \nabla_j^2 + \beta I \\ b &= \nabla\Phi^t \frac{\Phi^t - \Phi^{t-1}}{\Delta t}.\end{aligned} \tag{3.39}$$

When minimizing over the velocity $\mathbf{u}^t$, we omit terms that are constant with respect to $\mathbf{u}^t$, i.e., r. The dimensions of the matrix A are $n \times n$, where n is $3N$ and N is the total number of fluid grid cells. Each component j of one velocity vector covers one row. The first part of matrix A is the outer product of the density gradient vector $\nabla\Phi^t$. For each dimension, in our 3D case three dimensions, the density gradient is defined as $\nabla\Phi^t = (\frac{\partial\Phi^t}{\partial x_1}, \frac{\partial\Phi^t}{\partial x_2}, \frac{\partial\Phi^t}{\partial x_3})^T$ and therefore, the following holds:

$$\begin{aligned}(\nabla\Phi^t)(\nabla\Phi^t)^T &= \left(\frac{\partial\Phi^t}{\partial x_1}, \frac{\partial\Phi^t}{\partial x_2}, \frac{\partial\Phi^t}{\partial x_3}\right)^T \left(\frac{\partial\Phi^t}{\partial x_1}, \frac{\partial\Phi^t}{\partial x_2}, \frac{\partial\Phi^t}{\partial x_3}\right) \\ &= \begin{bmatrix} \left(\frac{\partial\Phi^t}{\partial x_1}\right)^2 & \frac{\partial\Phi^t}{\partial x_1}\frac{\partial\Phi^t}{\partial x_2} & \frac{\partial\Phi^t}{\partial x_1}\frac{\partial\Phi^t}{\partial x_3} \\ \frac{\partial\Phi^t}{\partial x_2}\frac{\partial\Phi^t}{\partial x_1} & \left(\frac{\partial\Phi^t}{\partial x_2}\right)^2 & \frac{\partial\Phi^t}{\partial x_2}\frac{\partial\Phi^t}{\partial x_3} \\ \frac{\partial\Phi^t}{\partial x_3}\frac{\partial\Phi^t}{\partial x_1} & \frac{\partial\Phi^t}{\partial x_3}\frac{\partial\Phi^t}{\partial x_2} & \left(\frac{\partial\Phi^t}{\partial x_3}\right)^2 \end{bmatrix}.\end{aligned} \tag{3.40}$$

The outer product of the density gradient is split onto three rows, one for each velocity component. The smoothness regularizer is applied for each velocity component j separately. Note that $\nabla_j^2 = \Delta$. The 2D version of the discrete Laplacian we employ here takes the form:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.41}$$

In 3D, the z-components are included as well and the coefficient of the central cell adds up to 6. For each velocity component j of a fluid cell, we add a Laplacian term to the corresponding row, which is weighted with α. We denote the Laplacian matrix with L_j. The Tikhonov regularizer is realized in a straightforward manner, as we just need to add β to the diagonal of A.

The right-hand side b is of dimension $3N \times 1$ and contains the product of the 3D density gradient and the scalar temporal density change $\frac{\Phi^t - \Phi^{t-1}}{\Delta t}$ for each fluid cell.

3.4. Computed Tomography

CT aims at reconstructing a function, e.g., a 3D volume, from a finite number of projections, e.g., 2D images. For example, multiple cameras are aligned in a semi-circle to record a fluid phenomenon taking place in their center, see Figure 3.4. Here, the 2D images are created with visible light.

The unknown volume is discretized into a finite number of uniform voxels stored in a vector v with n_v entries. The pixels of each image are combined into the vector p with n_p entries. For simplicity, we assume a linear image formation model, which we will specify in Section 3.4.1. Having a linear image formation model, we are able to express the relation between pixels and voxels as matrix-vector multiplication $Pv = p$, where P is the matrix projecting a volume to the images. With P, the dimensions of the 3D volume are reduced to 2D. For tomography problems, image values p are known while volume values v are unknown, hence, we are interested in solving for

$$v = P^{-1}p. \tag{3.42}$$

Linear system of equations for reconstructing voxel density values v based on pixel intensities p and a linear image formation model specified in matrix P.

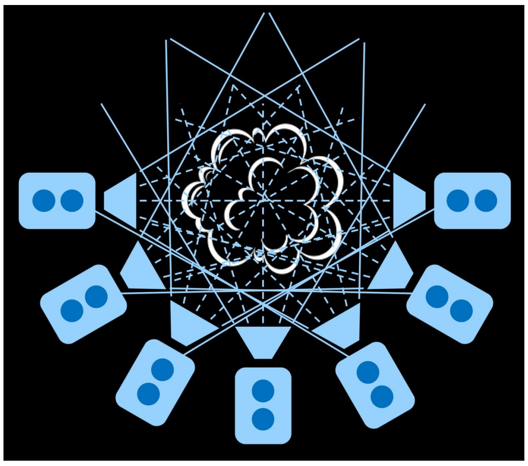


Figure 3.4.: Top view of seven perspective cameras recording a 3D fluid phenomenon.

However, as we have typically less pixels than voxels, i.e., $n_p << n_v$, our system of equations is severely under-constraint. Multiple different 3D volumes exist that project to the same 2D images, i.e., the system has more than one solution. Increasing the number of input images and therefore n_p reduces the degrees of freedom, but as it is very uncommon that the number of pixels matches the number of voxels, the matrix P is asymmetric and not invertible.

Furthermore, real-world recordings often feature noise and calibration errors in their pixel values. Thus, Equation (3.42) is not directly solved but with a least squares approach by multiplying both sides with the transpose of matrix P:

$$v = (P^T P)^{-1} P^T p. \tag{3.43}$$

The system's matrix $(P^T P)$ is symmetric and invertible but also larger, as its dimensions are $n_v \times n_v$ instead of $n_p \times n_v$ as before. The more input views are added, the denser the system's matrix gets. Even when storing only non-zero entries of the system's matrix, the matrix quickly becomes a bottleneck, as it requires huge amounts of memory for storage. Ihrke et al. [IM04] propose using a variant of the CG solver, called Conjugate Gradient Least Squares (CGLS), which approximates $P^T P$ during computation without explicitly storing the matrix product to avoid excessive memory and run time consumption.

After solving the equation system, there is no guarantee about the values in v being non-negative. In reality, voxel density values are never negative, i.e., $v_i \geq 0$ must hold for $i = 0, 1, ..., n_v - 1$. Ihrke et al. [IM04] avoid negative values by setting them to zero in each iteration of their CGLS solver.

Visual Hull Some voxels of the volume are known to be zero beforehand, i.e., the voxels that lie in an area that is seen by at least one black pixel. Removing these voxels from the system of equations not only drastically reduces the number of unknowns to solve for but also avoids spurious non-zero entries in voxels that should be exactly zero. Hence, reconstruction accuracy is increased. The so-called visual hull encloses all voxels that are possibly able to contain fluid density. The word hull is misleading,

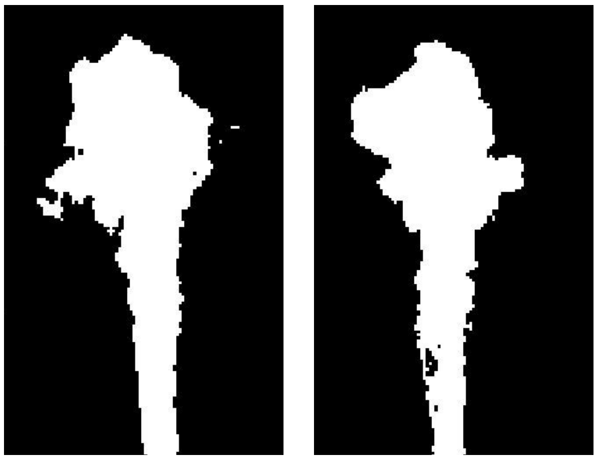


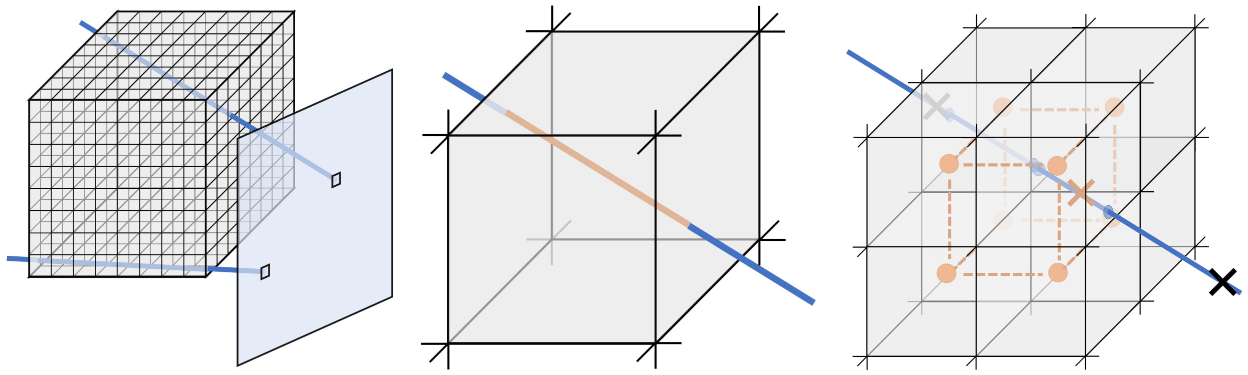
Figure 3.5.: Visual hull of a smoke plume seen from the front and side views, where the colors white and black denote the inside and the outside of the visual hull, respectively.

as the visual hull here does not necessarily have a connected boundary. The visual hull can rather be interpreted as the extent or silhouette of the fluid, and an example for the visual hull of smoke is shown in Figure 3.5. For an infinite number of viewing angles, the visual hull is equal to the convex hull of an object [Lau94]. The visual hull is created by evaluating which voxels lie in sight of a black pixel and marking them as not being included in the equation system.

Physical Capturing When recording videos with real cameras, pinhole camera models are assumed. Such pinhole cameras are perspective cameras with non-parallel rays spawned from a common origin, see Figure 3.4. For each pixel, one ray is spawned, which is specified by the origin, i.e., the pixel position, and the ray's direction, see Figure 3.6a. The larger the aperture angle of a camera, the more do the directions of the rays diverge. For some virtual cameras and for real-world cameras that are far from the object of interest, parallel rays are assumed for simplicity. Such cameras are called orthographic cameras. Reconstructions with orthographic cameras can be easier, as the distribution of the rays is evenly spaced, especially concerning the coverage of objects in the front and in the back.

As fluids are highly dynamic phenomena, synchronization of recording sensors is extremely important. Synchronization means that each camera takes their pictures at the very same time, which is more difficult to achieve the more cameras are used. Even more crucial is camera calibration. Here, camera parameters are estimated in order to determine the position and rotation of a camera relative to the scene and to correct lens distortions. In order to estimate those camera parameters, multiple images of calibration patterns are taken from different viewing angles. Such calibration patterns can be checkerboards but also circles where size and distances of the features must be known.

Because camera lenses are not infinitely thin and not perfectly aligned with the image sensor plane, they introduce distortion to the recorded images leading to altering the usual dimensions of a picture. There are two types of distortion: radial and tangential distortion. Radial distortion bends straight lines into curves. This effect increases the further the pixel is positioned from the center of the image.



(a) Tracing rays starting from image pixels and going through the volume's voxels.

(b) A ray traveling through a voxel where the red color denotes the part of the ray being inside the current voxel. The length of the red ray determines the voxel's contribution to the pixel value.

(c) Marching along a ray in equidistant steps and trilinearly interpolating the voxel's contributing amount at every step.

Figure 3.6.: Rays start at 2D pixels and travel through the 3D volume. The close-ups illustrate different approaches of how to determine the amount by which a voxel's density influences the originating pixel's intensity.

Tangential distortion leads to areas in the image looking closer than they should and stems from misalignment of optical centers of various lens elements [BB01]. Distortion is different for every camera, as it depends on manufacturing. Once the distortion coefficients are calculated based on the images of the calibration patterns, recorded images can be undistorted via post-processing.

3.4.1. Image Formation Model

In order to establish a connection between the voxel's density and the pixel's intensity values, an image formation model must be defined. In reality, cameras take photographs of the smoke's distribution being influenced by external lighting, emission, scattering, and absorption effects. Modeling real-world image formation is highly complex and therefore, exact modeling of reality is impossible but good approximations of photorealistic images exist [Max95]. Pharr et al. [PJH16] describe how to simulate the effect of participating media, such as smoke and fog, in Chapter 11. However, as we record thin smoke plumes, we assume negligible absorption and scattering effects as well as proportional self-emission just like [IM04], which results in a simple, linear image formation model.

The function beneath the image formation model in [IM04] is described as following:

$$I_p = \int_c \Phi(s)ds, \tag{3.44}$$

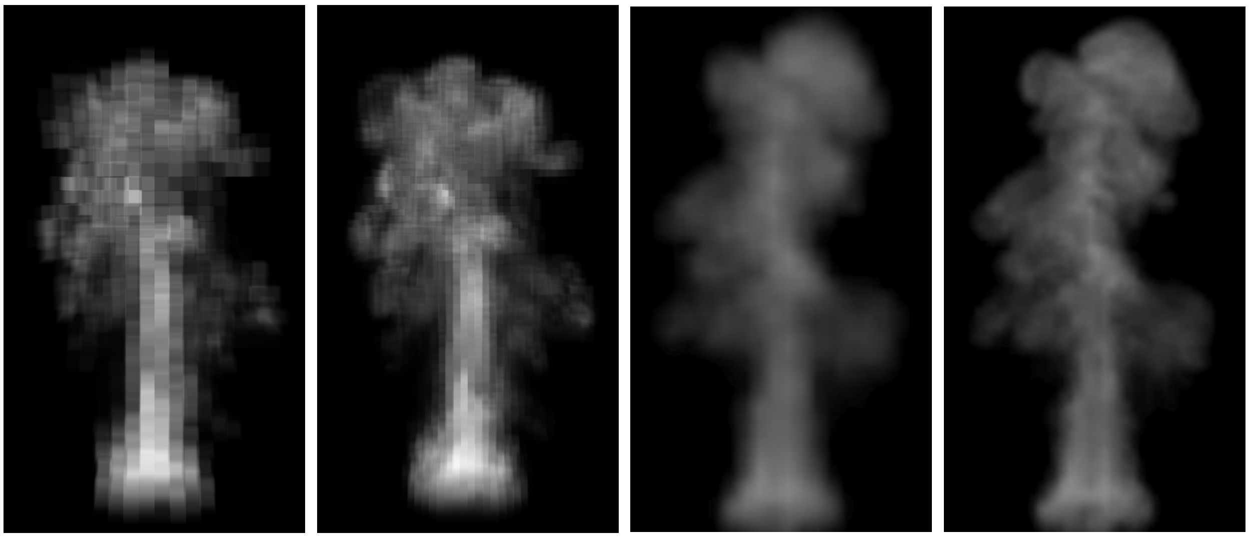
where I_p is the intensity value of pixel p, Φ is the density, s is the position, and c is a curve through space. The intensity function can be expressed as a linear combination of basis functions:

$$I_p = \int_c \sum_i (a_i \Phi_i)ds, \tag{3.45}$$

where Φ_i are the basis functions and a_i are the corresponding coefficients. The simplest basis function is the box function, which is one for a voxel with index i and zero for all others. The coefficients a_i are stored in the projection matrix P.

In order to obtain the pixel-voxel correspondences of an image formation of semi-transparent objects, we cast an invisible ray from each pixel passing through the 3D object, see Figure 3.6a. Ray tracing, ray casting, and ray marching are often interchangeably used for describing calculations of ray intersections. Ray tracing describes the technique of tracing a ray to its first intersection with an object, which leads to spawning new rays from the intersection point, which are traced as well. Advanced effects as reflections or refractions can be realized with ray tracing. Ray casting is an analytical approach of directly and very accurately calculating the intersection points of a ray with objects like voxels. Ihrke and Magnor [IM04] make use of that approach and store the distance that a ray traveled within a specific voxel in the projection matrix P, as visualized in Figure 3.6b. The distance a ray travels through a voxel hence determines how much a voxel's value influences the intensity of the corresponding pixel. Here, a piecewise constant image formation, i.e., box basis functions, is assumed.

Piecewise constant basis functions lead to block artifacts in the images, see Figure 3.7 on the left. Images rendering the same smoke plume voxelized onto two grids with different resolution will differ drastically due to the piecewise constant pixel-voxel-correspondences. Therefore, we propose to use ray marching for setting up the projection matrix P. Starting at the ray's origin, we march along the ray with a specified step size. At each step position on the ray, the trilinear interpolation weights of the surrounding eight voxels are stored as weights in P. An example process is shown in Figure 3.6c. Hence, we are using piecewise linear basis functions. Such ray marching leads to smoother images, which do not vary as much across different voxel resolutions as shown in Figure 3.7 on the right.



(a) Piecewise constant box basis functions. **(b)** Piecewise linear basis functions by trilinear interpolation.

Figure 3.7.: Visualization of block artifacts for piecewise constant basis functions in a) and smoother images from piecewise linear basis functions through ray marching in b). For each method, the a plume is rasterized onto two grids with base and double resolution on the left and right, respectively. Although block artifacts decrease for higher grid resolution when using piecewise constant basis functions, the images are very different. When using piecewise linear basis functions, the images for both resolutions are smoother and more similar, which is beneficial for tomographic reconstructions of continuous real-world phenomena and also for reconstructing synthetic plumes on multiple different scales.

4 Capturing Setup and Reconstruction Algorithms

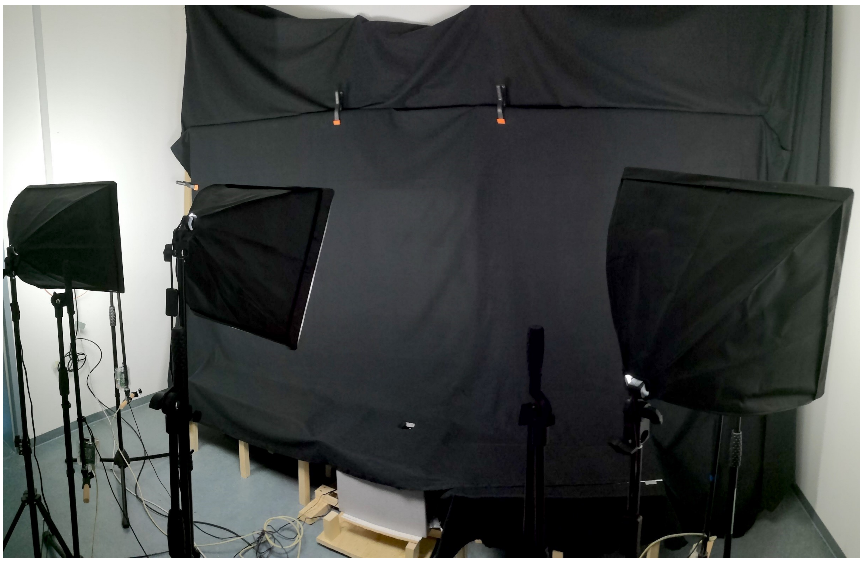
In this chapter, we present our simple, affordable, and reproducible hardware setup for capturing real fluid flows and our highly powerful reconstruction technique. We capture fog phenomena, which are representative for single-flow fluids, such as hot smoke plumes. Therefore, we continue to refer to our traceable markers as smoke although we are using a fog machine to create them. With our reconstruction technique, we are able to reconstruct greatly under-determined 3D smoke plumes recorded with an unusually simple hardware. Our reconstructions exhibit physically convincing behavior from novel views, since we couple the reconstruction of density and velocity tightly to counteract under-determination.

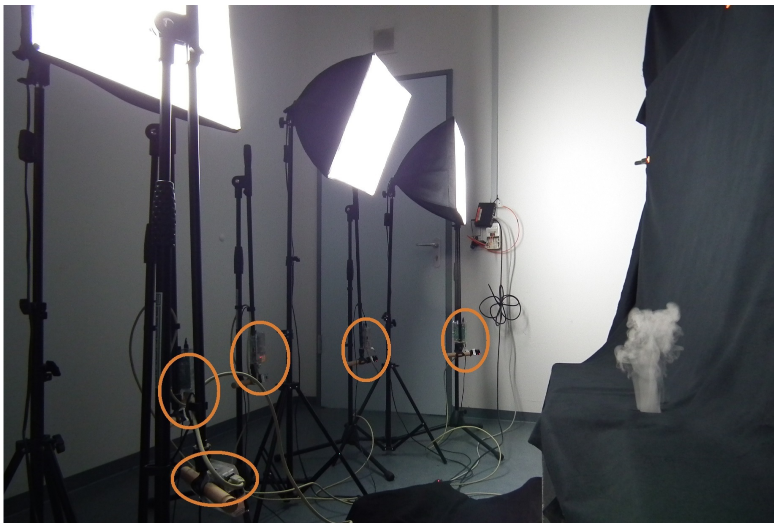
4.1. Hardware Setup

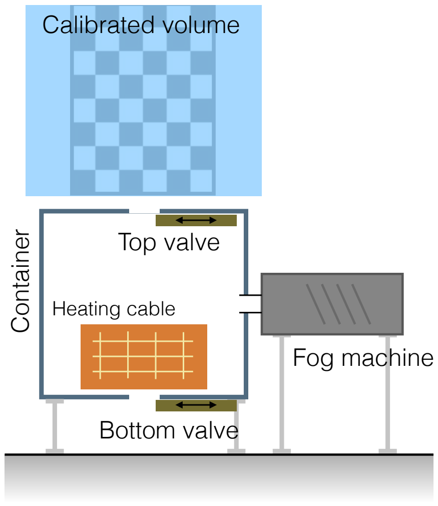
With our hardware setup, we create and record controllable, repeatable smoke plumes. Fog fluid for common fog machines is water-based while some fog machines diffuse water-free oils. Our visible tracers are droplets of distilled water with propylene glycol. Our setup consists of inexpensive components as listed in Appendix A, which are easily replaceable and adjustable to custom needs. In total, the whole setup consists of hardware that is available for less than 1100 \$. Therefore, in contrast to previous capturing setups in graphics [HED05, XIAP*17], our setup is economical and can more easily be recreated.

We make use of several Raspberry Pi computers for recording, for controlling the amount and speed of smoke release, and for positioning our calibration board. We first discuss challenges in Section 4.1.1, describe the smoke generation in Section 4.1.2, present background and lighting in Section 4.1.3, specify the details of our recording hard- and software in Section 4.1.4, and outline post-processing steps

(a) Rising smoke plume.


(b) Our setup with black background and three light sources.


(c) Our setup with five Raspberry Pis and mounted cameras, which are marked by orange circles, and rising smoke plume.



(d) Sketch of our smoke generation components.

Figure 4.1.: A rising smoke plume in a) and an overview of our simple fluid capturing setup with black molton as background, covered smoke box with opening on the top, Raspberry Pis including camera modules mounted on microphone stands, and three diffuser lights in b) and c). The smoke generation part consists of the alleviated smoke container with its top and bottom valves, an electric heating cable, and a fog machine as illustrated in d). The volume to reconstruct is placed above the top opening and is calibrated with a movable calibration board. These components are covered with black molton when recordings take place, as shown in the photographs in a), b), and c).

in Section 4.1.5. Our implicit and automatic camera calibration method is specified in Section 4.1.6.

An exemplary rising smoke plume produced with our capturing setup is displayed in Figure 4.1a. The capturing setup is shown in Figure 4.1b, where the black background cloth and the three lights for illumination are visible. A different angle of our setup is presented in Figure 4.1c, which shows five Raspberry Pis mounted on microphone stands recording a rising smoke plume. A sketch of the components necessary for smoke generation is illustrated in Figure 4.1d. While recording, those components are covered with black cloth. The dimensions of our reconstruction domain, which is highlighted in Figure 4.1d in blue, are 50 cm × 90 cm × 50 cm, where we refer to the height with $L = 0.9$ m. The kinematic viscosity of air at a temperature of 20° C is $\nu = 1.516 \cdot 10^{-5}\ \frac{\mathrm{m}^2}{\mathrm{s}}$, see [Enga]. Average smoke plumes rise with the upward speeds of approximately $u = 0.27\ \frac{\mathrm{m}}{\mathrm{s}}$ to $u = 0.4\ \frac{\mathrm{m}}{\mathrm{s}}$, leading to a Reynolds number Re=$\frac{uL}{\nu} = 0.27 \cdot 0.9 \cdot 1.516 \cdot 10^5 \approx 3.7 \cdot 10^4$ to Re=$5.4 \cdot 10^4$. These Reynolds numbers indicate that the generated flows contain structures transitioning from laminar to turbulent. We produce our captures with a temperature of 34° C, where interesting and swirly structures are formed in the smoke plumes while passing through our calibrated volume.

4.1.1. Challenges

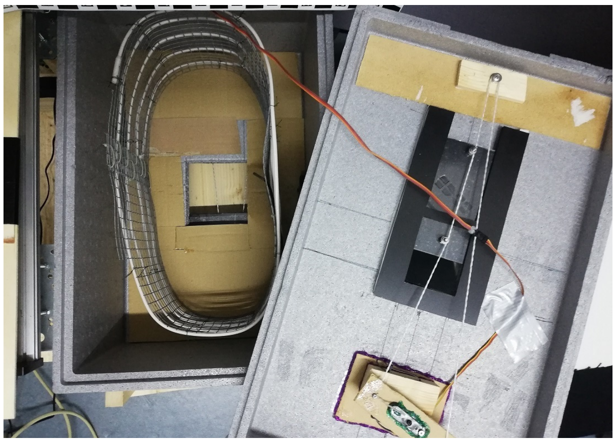
The first challenge we face when capturing fluid behavior from a common fog machine is the speed and the control of the exiting hot marker particles. Our machine produces a fast, hot stream of diffused water, which is not controllable as such. To slow down the stream, we fill the smoke into a container with an opening at the top. However, with an open outflow area, we are still not able to control the amount of smoke and the precise timing the smoke exits the container. Therefore, we need a valve for controlling the opening. As conservation of volume holds, we need a second opening accompanied with a valve to account for adequate air inflow. Otherwise, smoke is not able to leave the container. But, smoke cools down very fast, which reduces its buoyancy leading to smoke staying in the container even when opening both valves. Hence, we control the temperature of the smoke through heating the air in our container. With control over the smoke temperature as well as density and speed of outflowing smoke, we are able to generate controllable and varying smoke plumes on demand.

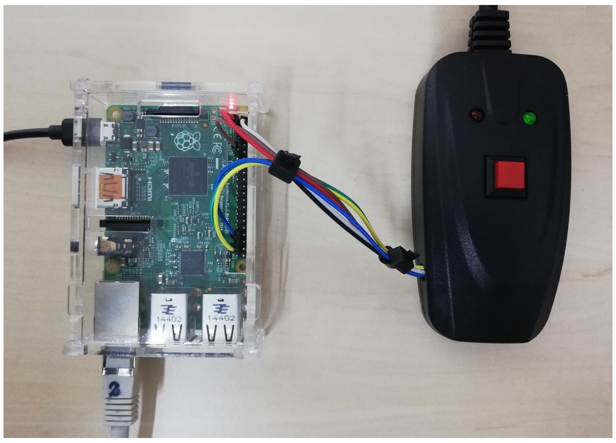
The second challenge is to create high contrast between the nonrelevant background and the foreground smoke plumes and to uniformly light the scene in order to simplify post-processing of the captured images. Cameras should see a bright and detailed smoke volume on dark background without obstructing obstacles or reflections. In order to fully make use of available information in an image, we avoid over- and underexposure of the smoke.

The third challenge involves synchronizing and storing highly resolved recordings. The cameras should record the dynamic smoke plumes at very similar times. Unfortunately, frames are dropped and lost if the amount of data to store exceeds the writing speed of the Raspberry Pi computers when writing onto their SD cards. It is challenging to find optimal temporal and spatial resolution as well as bit rate for high-quality recordings without dropping any frames.

The last challenge is to obtain information about how camera rays travel through the reconstruction volume in order to establish the relation of density in the volume and the pixel's intensity. We obtain the pixel-voxel correlations through accurate camera calibration, see Section 4.1.6.

4.1.2. Smoke Generation


(a) Heating cable and moving lids actuated by servo motors.



(b) Raspberry Pi driving remote control of fog machine.

Figure 4.2.: Components of our smoke generation setup: smoke box with heating cable, top and bottom openings with closing lids controlled by servo motors in a), and a Raspberry Pi driving the remote control of the fog machine in b).

A sketch and photographs of our smoke generation components are shown in Figure 4.1d and Figure 4.2. As smoke container, we use an inexpensive, insulated Styrofoam box. The dimensions of the box are 54.5 cm × 35 cm × 30 cm. It features a thermal conductivity of 0.03 $\frac{\text{W}}{\text{mK}}$ at a temperature of 25° C [Engb], and hence exhibits low dissipation of heat. We use an electric heating cable to heat the smoke to a target temperature. The maximum temperature is 60° C, which does not pose any safety risks for the box, as the melting point of polystyrene is 240° C. Furthermore, our heating cable is connected to a safety timer and is controlled by a thermostat. With the thermostat, we ensure that the desired temperature is reached, kept, and not exceeded. In order to avoid damaging the surrounding box and to uniformly distribute the warmth, our heating cable is wired around a metallic fence as visualized in Figure 4.2a.

As polystyrene is easily editable, we simply cut openings with a carpet cutter into our Styrofoam box. The top opening allows for smoke outflow while the bottom opening grants air to flow into the box. Both can be closed through a movable lid, called valve, as displayed on the right of Figure 4.2a considering the top valve. The valves are controlled via strings connected to servo motors driven by a Raspberry Pi computer and are closed when filling the box with smoke. For controlled release of smoke, the bottom and top valve are opened to a certain extent. If the bottom valve is closed, the smoke does not rise from the box due to volume conservation. The box is elevated from the ground by four pillars to allow for proper air flow below the box. The bottom opening is 12 cm × 12 cm big while the top opening is 7 cm × 7 cm large. The opening extent of both valves and the smoke temperature influence the amount and speed of rising smoke and hence, how much turbulent structures are created.

Besides the top and bottom openings, we cut a circular hole into the side of the box to permit smoke inflow. To guide the smoke from the fog machine into the side opening, we place a straight silicone hose from the machine's nozzle to the box. The silicone hose has a diameter of 25 mm, features low thermal conductivity, is resistant to humidity, and is physically stable. The smoke machine's model is an Eurolite N-10. To automate and accurately steer smoke captures, we control the smoke machine's remote control with a Raspberry Pi, which is shown in Figure 4.2b. When the green LED lights up, the machine is ready to release smoke and the Raspberry Pi eventually sends a signal to release smoke for a certain amount of time. The longer the signal is sent, the more smoke is pushed into the box.

With our box, adjustable smoke temperature, controlled smoke inflow, smoke outflow, and air inflow, we are able to create manipulable steady smoke flows forming interesting and turbulent structures. One example for such a plume is shown in Figure 4.1a.

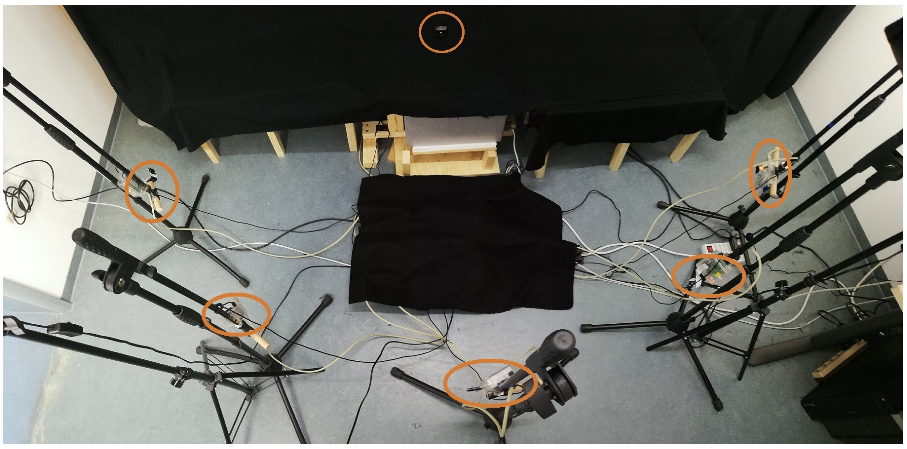
4.1.3. Background and Lighting

In order to decrease reflections in the background, we put black molton cloth behind our smoke plumes. We use two large movable walls extended to the height with long sticks to form a black background of approximately 2.8 m × 2.2 m, as shown in Figure 4.1b. To reduce the required size of the movable walls, we arrange both within a small angle to each other, imitating a curved background. Hence, the walls serve as background for a wider angle of cameras as they would without the imitated curve. Black molton cloth forms minimal wrinkles and absorbs light, which reduces reflections. Furthermore, we can bend it to fit the slight angle between our two movable walls. In contrast, black paper cannot be bent and produces glossy effects due to reflections, which influence the resulting pixel's intensities in the images tampering the reconstruction quality. The black background color is beneficial, as it increases the contrast between the background and the bright smoke plumes.

To further optimize contrast, illumination of the scene is important. Our goal is to illuminate the smoke plume uniformly from each angle, such that the perceived brightness is equal for each camera and smoke structures are clearly visible. We target fully exploiting the capacity of storing information in images with intensities from 0 to 255 where the background color should be black, i.e., 0, and the brightest parts of the smoke white, i.e., 255. Hence, we install three lights with diffuser softboxes in front of them, which produce a diffuse, soft, and nearly uniform illumination. We place the lights behind the cameras as shown in Figure 4.1b.

4.1.4. Recording

We record the smoke plumes with camera modules attached to Raspberry Pi computers. The computers are Raspberry Pi Models 1 or 2 while the camera are modules of version 1.


(a) Top-down view of microphone stands with cameras focusing on the smoke release, where both cameras and smoke release are marked by orange circles.

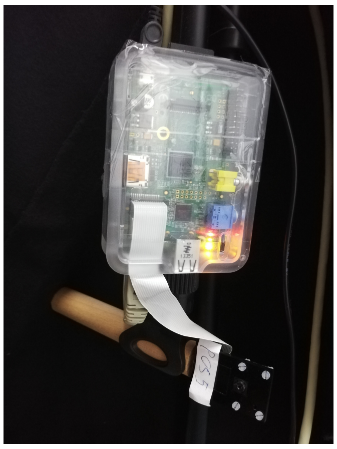

(b) Close-up of Raspberry Pi computer with mounted camera module.

Figure 4.3.: Arrangement and close-up of our Raspberry Pis with mounted camera modules.

Setup We mount both computers and cameras on microphone stands in order to flexibly adjust their position and orientation, as shown in Figure 4.1b and Figure 4.1c. We arrange the cameras in a partial circle of approximately 90° between the two outermost camera positions. The cameras are placed with equal distance of $\approx$ 120 cm to the smoke release, at a similar height as the smoke release, and facing slightly upwards. The positioning of the cameras is displayed in Figure 4.3a. A close-up of one Pi computer with a mounted camera module is visualized in Figure 4.3b. The Raspberry Pis are connected to a switch and a router via Ethernet, allowing them to communicate with a host computer. Each Raspberry Pi has a generic SD card or a MicroSD card with at least 10 $\frac{\text{MB}}{\text{s}}$ of writing speed. We ensure that cameras do not see each other in order to facilitate post-processing the images. This poses a limitation to the angle the cameras are able to cover and to the orientation of the outermost cameras.

Technical Details The cameras have three physical ports: a video, still, and preview port. We record the smoke plumes via the video port using an H264 encoder. Although the video data contains more noise compared to images taken via the still port, the video port is better suited for taking multiple images within short time intervals. The required frequency of taking images could not be realized by using the still port. We use camera video mode 1, where the spatial resolution is 1920 × 1080 pixels. We set the temporal resolution to 60 fps.

We empirically determined suitable exposure correction values for each camera to avoid over- and underexposure. The auto-white-balance is corrected by iteratively taking pictures of a white sheet of paper and adjusting the gain values until the color white is also perceived as white in each camera. White balancing is the process of eliminating chromatic discrepancies in images captured by digital cameras due to varying illumination conditions. The correction is performed by simple matrix multiplications.

Synchronization As we record dynamic smoke plumes, it is important that each camera captures the plume formation process at approximately the same time. We control the cameras by connecting all Raspberry Pi clients to a host computer, from which the whole capturing process is directed. The clients receive the command to start the capture at a certain time. Before starting the captures, the host sets exposure and white balance values for each camera.

Another important factor is the size of data to be stored and the writing speed of the Raspberry Pi computers. Video streams can only be fully stored if the combination of frame rate, resolution, and bit rate does not exceed the possible writing speed of the SD card. Otherwise, frames are dropped and lost, which leads to an inconsistent and varying frame rate for each camera. If the frame rate is set too low or the image resolution is too small, we lose information about the temporal or spatial behavior of the smoke plumes. The bit rate determines how many bits are used to encode a frame. It takes the motion rank into account, which specifies how much the motion varies across frames. If the bit rate is set too low, compressing artifacts occur, which average pixel values and blur interesting and fine-scale smoke details. Our final capture videos exhibit a bit rate of 19848 kbps, a spatial resolution of 1920×1080 pixels, and a temporal resolution of 60 fps.

4.1.5. Post-Processing

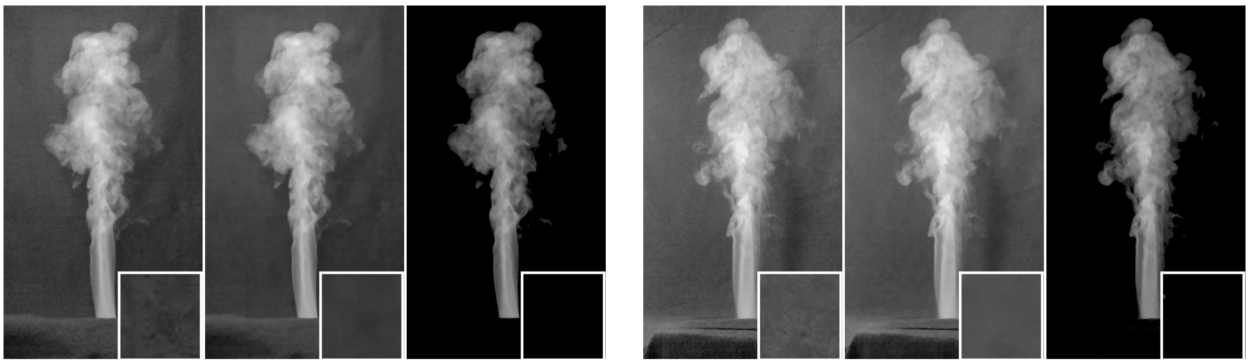
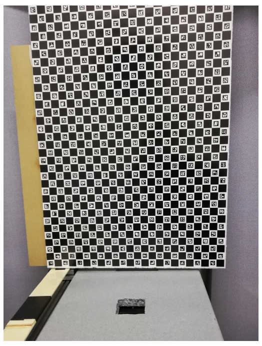


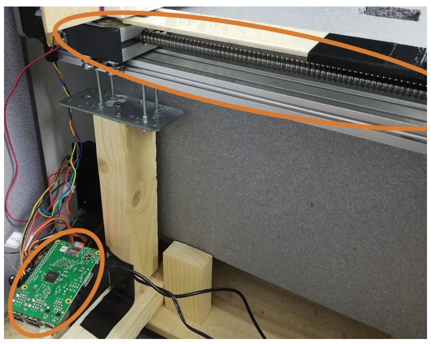
Figure 4.4.: Effect of our post-processing pipeline demonstrated on one smoke plume recorded from two distinct camera views. The first image is the captured image converted to gray scale, the second is after applying denoising, and the third shows the fully post-processed image, i.e., after subtracting the background and after thresholding. The denoising effect is not obvious to observe, but removes noise in the background as well as in the smoke, which leads to smoother temporal behavior. The small insets on the lower right show a close-up of the noisy background area left to the plume. In image sequences, this time-varying noise is perceived as temporal flickering.

We first convert the H264-encoded videos to .mp4 in order to facilitate visual inspection. Then, we extract single frames off the .mp4 videos and convert them to gray scale. In order to reduce noise and hence temporal flickering, we use a Non-Local Means Denoising procedure [BCM11], called `fastNlMeansDenoising` in OpenCV, to denoise our sequences. The mean of the spatial and temporal neighbors is used to adjust the value of each pixel. As such, pixel windows of different sizes are used. This denoising procedure is computationally expensive but provides excellent denoised image sequences. We use a filter strength of 3, a temporal window size of 7, and a spatial search window size of 21. In order to eliminate background unevenness like wrinkles or varying brightness, we subtract a frame of the early sequence from each frame. The frame of the early sequence does not show any smoke behavior yet, but we do not use the very first frame since recording parameters need to adjust such that artifacts become minimal. As final step, we apply zero thresholding where we set each pixel with a value below 8 to zero in order to avoid that remaining background noise is interpreted as moving smoke particles. If images were recorded with more complex lighting and diverse background, we could include advanced methods from the image processing area to smoothly extract the smoke plume. With our implicit camera calibration technique outlined in the following section, we do not need to undistort the captured images in our post-processing pipeline.

4.1.6. Calibration

We employ a simple, convenient, automatic, and accurate calibration method, which produces dense ray calibration data with optimized correspondence between rays and voxels. In Figure 4.5, we show our calibration board with affixed ChArUco markers, the rail on which the board moves, and a sketch of how reference points are found in space. The black molton cloth is lifted to uncover the marker. After calibration, the black cloth is put back to cover the box, the marker, and the fog machine.


(a) ChArUco marker on a flat polystyrene board above the smoke box.


(b) Rail (orange ellipse) on which the marker moves. The rail is actuated by a stepping motor driven by a Raspberry Pi (orange ellipse).

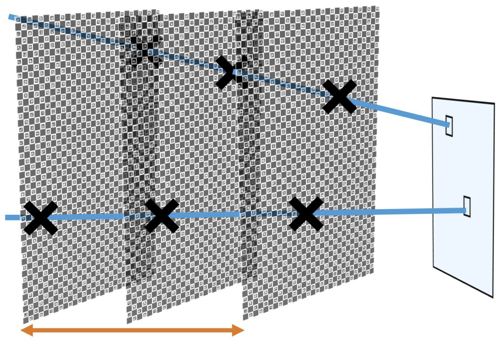

(c) Reference points on the marker board at three different positions in space observed from one camera. The orange arrow describes the linear marker movement.

Figure 4.5.: Implicit calibration with a movable ChArUco marker.

Implicit vs. Explicit Calibration In order to establish the connection between pixel intensities and voxel densities, traditional calibration approaches assume a specific camera model, determine its camera parameters, and undistort the captured images. Such approaches perform explicit calibration to define intrinsic (optical center, focal length, and skew coefficient) and extrinsic (location and orientation) camera parameters, as well as distortion coefficients. With these parameters, image distortions can be corrected and rays originating from each pixel and going through the volume can be calculated in world space. Such explicit methods are mechanically simple and flexible, as they only require static markers and a small amount of reference points. The disadvantage is that possibly simplified camera and distortion models must be assumed, which have a major impact on the accuracy of the calibration.

In regard to our capturing setup, we use an implicit method for camera calibration. Without undistorting images or determining position and orientation of the cameras, the corresponding rays are calculated directly by interpolating between reference points in space. The calculation of intrinsic or extrinsic camera parameters is dispensable. However, in order to calculate the line of sight for each pixel, at least two reference points with different depth levels are required, for which dynamic markers are needed. Moving markers are used where their exact positions in space are known. There exist many variations of such implicit camera calibrations [MBK81, WM93]. Implicit calibration methods are not limited to a camera or a distortion model, but are able to handle any types of cameras and even refracting materials between camera and target without losing accuracy. A high calibration accuracy can be achieved where outliers do not affect the calibration accuracy as much. With our implicit camera calibration, we obtain a starting position and a direction for each ray in world space. We step along the rays to determine which voxels influence the given pixel and to which degree. The weights specifying the degree of influence of each voxel to each pixel are stored in our image formation matrix P, see Section 3.4.1.

Implementation For implicit calibration, it is inherently important to have access to the precise location of the marker. We mount a marker board on a ball screw rail controlled and driven by a stepper motor and a Raspberry Pi. Through rotating the motor by a given angle, the spindle turns a small amount, which in turn moves the rail forward linearly. The board is only moving into one direction, namely straight along the rail. In every position, each camera takes an image after waiting for the wobbling of the marker to stop, which typically occurs after moving the marker to a new position. The marker is stopped at twenty-one equidistant positions. The home position is accurately defined, as the rail has an end switch, which is pressed when the marker is returned completely. We use a ChArUco marker of size 50 cm × 100 cm glued to a flat polystyrene board, which is mounted onto the rail. A ChArUco marker combines the benefits of chessboard and ArUco markers. The corners of two squares in a chessboard are detected very accurately. ArUco markers have a unique ID and are hence identified even when the marker is partially invisible. Placing ArUco markers in the white squares of a chessboard as shown in Figure 4.5a ensures that the patterns of the ChArUco marker are detected with a high accuracy even with partial occlusion.

For each pixel, an average line going through all reference points is calculated by interpolating between marker points and calculating image-to-world transformation matrices. The resulting ray directions for each of the five cameras, from left to right, are encoded in color as shown in Figure 4.6. Pixels without an assigned ray are black. The borders of the markers are usually not well detected. Rays for such pixels and also for pixels, which are geometrically not able to see the marker, are extrapolated from the neighboring pixels' rays. For good calibration accuracy, it is necessary that each camera is able to see the marker board in each of its positions. Steeper viewing angles, such as for the outermost cameras, lead to less accurate calibration data. The volume right above the box, which is covered by the marker, is calibrated.

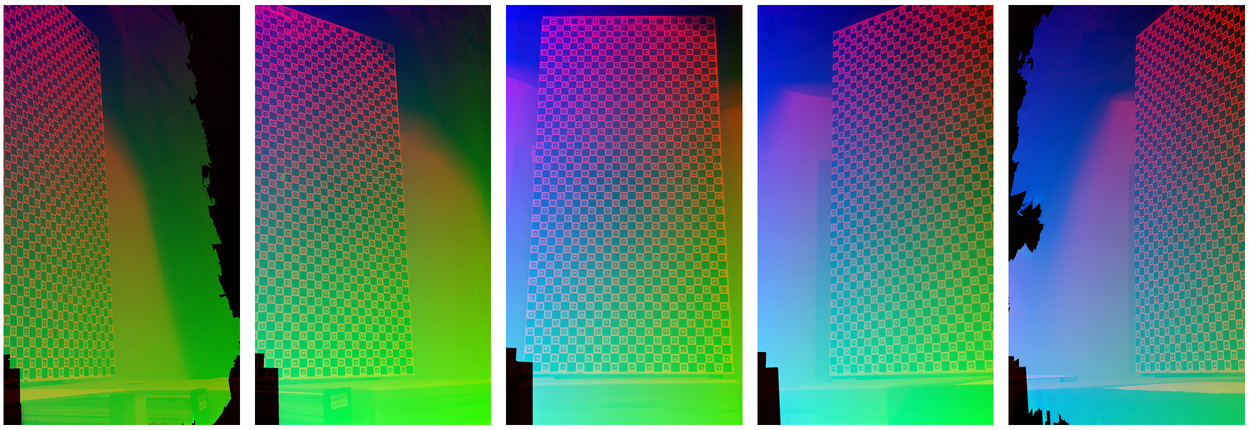


Figure 4.6.: Calibrated ray directions in world space encoded in RGB values for our five cameras, from left to right.

Accuracy The accuracy of such implicit calibration methods is high if a large number of reference points is given as in our case. The rays are directly calculated from the marker images without any assumptions about intrinsic or extrinsic camera parameters. However, estimated rays covering the margins of the marker or no marker at all possess a low fidelity. Calibration errors still occur due to inaccuracies of corner detection, due to sub-pixel lens distortion, or due to inaccurate marker positions. As the marker exhibits high contrasting black and white areas, images of the marker tend to be over-exposed, which leads to bright areas bleeding into darker areas. While chessboard detection is still accurate with overexposure artifacts, the identification of ArUco markers is more difficult. Such undetected markers are discarded, which results in incomplete calibration data. However, we found our calibration method to work well, as our reconstruction technique is robust to small calibration errors.

4.2. Reconstruction Technique

In Chapter 3, we outlined fundamental methods of fluid simulation, OF, and CT. We make use of these techniques to reconstruct volumetric density and velocity based on few input images, which poses an under-determined, sparse reconstruction problem. Typical fluid simulators solve the NSE via operator splitting, where advection, pressure projection, and the incorporation of internal and external forces are central components, see Section 3.1 and Figure 3.1. However, we cannot rely on a pure forward simulation to reproduce the behavior given in the real-world input images as initial density distribution Φ^0, density inflow Φ^t_I, initial velocity $\mathbf{u}^0$, velocity inflow $\mathbf{u}^t|_I$, external forces $\mathbf{f}_\mathrm{ext}$ like buoyancy, and ambient air motions are unknown. Furthermore, forward simulators introduce artificial viscosity hindering the creation of turbulent fluid structures, as explained in Section 3.1.

Hence, in order to recreate the turbulent motion from the captured real-world smoke plumes, we guide a forward fluid simulation with the input images to reconstruct real-world density and velocity. Finding an instance of a numeric simulation that matches the given set of observations, is the inverse problem to forward fluid simulation. To find such an instance, we need to account for all unknown and missing velocity parts, i.e., forces, which we call residual velocity $\Delta\mathbf{u}^t$. Furthermore, we must estimate the unseen density inflow Φ_I^t to create the required amount of smoke density determined by the input images. A schematic overview of our reconstruction technique based on a forward simulation is shown in Figure 4.8. The components advection, pressure projection, and viscosity are building blocks from the forward simulator in Figure 3.1. The blue and orange parts are the main parts of our reconstruction technique, which are the calculation of the residual velocity $\Delta\mathbf{u}^t$, the density inflow estimation ($\tilde{\Phi}_\mathrm{I}^t$ and Φ_I^t), and the initial density field Φ^0. The procedure will be explained in more detail in Section 4.2.2. A sketch of our reconstruction domain is shown in Figure 4.7. The reconstruction domain is denoted with Γ and the inflow region with Γ_I. As the reconstruction domain is finite in contrast to the real world, we apply outflow BCs at the top and side domain boundaries. Outflowing densities are deleted and velocities are moving freely. The upper region of the domain is more difficult to reconstruct due to the inconsistency of infinite world and finite domain, but we found our algorithm to handle it well in practice as demonstrated in Chapters 5 and 6.

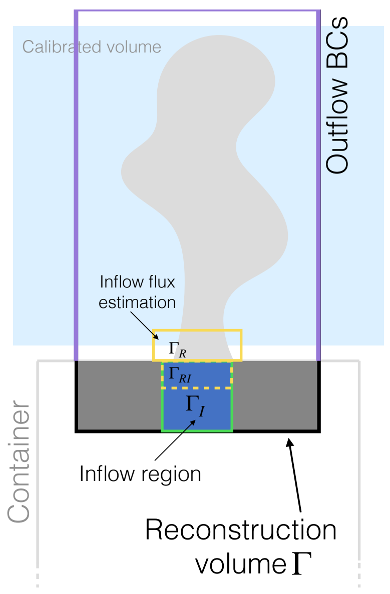


Figure 4.7.: Numerical simulation and reconstruction domain Γ, which contains outflow BCs at the top and sides. The invisible inflow region is denoted with Γ_I. The part of the visible domain that is *reached* when advecting the inflow area Γ_I with a given velocity is denoted with Γ_R. The subarea of Γ_I that reaches the visible domain after advection is called Γ_{RI}.

We make use of our implicit camera calibration from Section 4.1.6 to obtain a ray direction for each pixel, as visualized in Figure 4.6. In contrast to Ihrke and Magnor [IM04], we march along each ray in equidistant steps and store the trilinearly interpolated influence of each voxel in our image formation matrix P producing smooth and realistic images, see Figure 3.7.

Furthermore, our main contributions are:

- joint and tightly coupled reconstruction of both density and velocity through a strong optimization formulation making use of physics-based and geometric constraints as well as an efficient primal-dual optimization implementation,
- an advanced method for estimating unseen density inflow and a prescribed inflow velocity estimated from the speed of the rising plumes in the input images,
- a staggered discretization of OF allowing to use a MAC grid in the pressure projection,

- a progressive tomography scheme incorporating smoothness and kinetic regularizers into an efficient CGLS solver, enforcing non-negative density values through an additional primal-dual optimization, using implicit camera calibration without assumptions about camera or distortion models, and using an adapted visual hull technique, which considers the number of seeing cameras as well as pixel windows with improved thresholding for both pixel values and voxel weights.

We use this reconstruction technique for both multi- and single-view reconstructions in Chapters 5 and 6, where the single-view technique is slightly adapted as explained in Section 6.1.

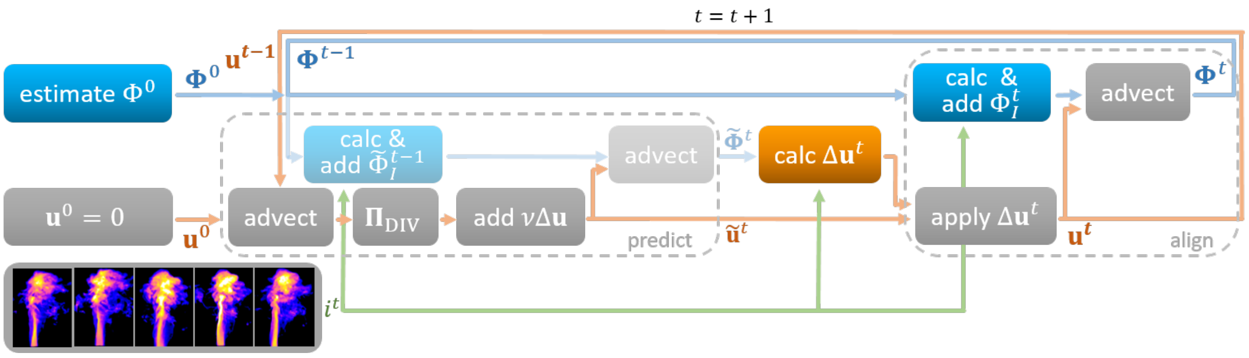
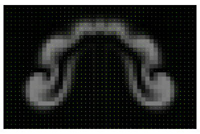


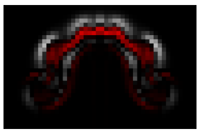
Figure 4.8.: Steps of our reconstruction technique calculating velocity $\mathbf{u}^t$ and density Φ^t. As in Figure 3.1, the velocity's solving path is denoted with orange arrows, while the density is solved following the blue lines. As initial conditions, we assume the velocity to be at rest, i.e., $\mathbf{u}^t = 0$, and estimate the density Φ^0 with a tomography solve prohibiting negative densities. For each time step t, we reuse the advection, pressure projection, and diffusion components of the forward simulator to predict a velocity $\tilde{\mathbf{u}}^t$. The density is predicted by calculating and adding a density inflow $\tilde{\Phi}_{\mathrm{I}}^t$, which is estimated considering the predicted velocity $\tilde{\mathbf{u}}^t$ and input images i^t. As the predicted density $\tilde{\Phi}_{\mathrm{I}}^t$ is only an intermediate helper variable, these steps are grayed out. Based on the predicted density $\tilde{\Phi}^t$ and the input images i^t, a residual velocity $\Delta\mathbf{u}^t$ is calculated, which is added to the predicted velocity $\tilde{\mathbf{u}}^t$. Considering the input images i^t and the final velocity $\mathbf{u}^t$, a final density inflow Φ_{I}^t is estimated. After advecting the sum of density inflow Φ_{I}^t and previous density Φ^{t-1}, the final density Φ^t is obtained.

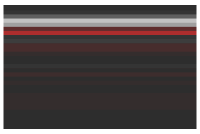
4.2.1. Problems of Decoupled Reconstruction Methods

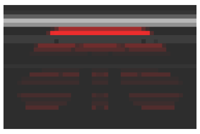
Traditional methods directly compute the density field Φ^t, which works well for accurate tomographic reconstructions using a large number of input images. As the number of projections decreases, the tomography problem becomes more ill-posed, such that reconstruction accuracy decreases drastically when using existing methods. In order to demonstrate the demand for more sophisticated reconstruction techniques when using sparse views, we present preliminary results of early stages of our reconstruction technique in Figure 4.9. Here, we focus on density reconstruction, as density implicitly leads to insights about motion as well. For simplicity, we first use an orthographic camera and a 2D smoke plume, which is projected onto a one-dimensional (1D) projection. We use only one single input view to emphasize the reconstruction difficulties and compute a density change to an existing plume, such as in Figure 4.9a. Here, our density change is required to match the target change in 1D space, which is the projected ground truth density change between two time steps, see in Figure 4.9b. We visualize the


(a) Original smoke plume.

(b) Change in 1D projection.


(c) Original density change.


(d) Uniform expansion.


(e) Uniform expansion, non-negativity.

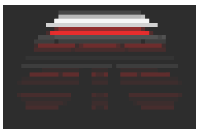

(f) Uniform expansion, non-negativity, and mask.

Figure 4.9.: Based on an original 2D smoke plume in a) and a 1D projection b) of the ground truth density change c), which is required to meet the next frame, we estimate the density change in 2D with different methods d), e), and f). First, we uniformly distribute the values of the 1D projection along each line of sight as shown in d). Then, we add the constraints of non-negativity in e) and additionally, a local mask in f).

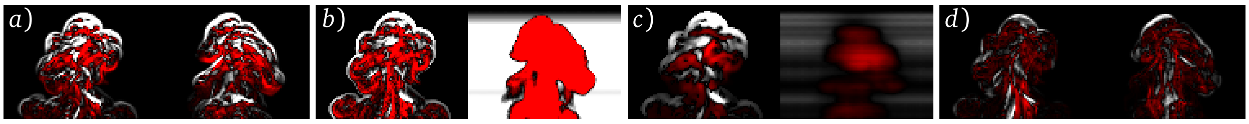


Figure 4.10.: Front and side views of 3D density changes. The ground truth density change leading from one time step to another is shown in a). Estimating the density of the next time via traditional tomography results in a density field uniformly distributed along each line of sight. By building the difference to the current ground truth density field, we obtain the density change shown in b). First calculating the difference in 2D image space and then projecting it into 3D space plus using a smoothness prior, leads to the density change in c). Our calculated density change is shown in d). While the front views match the ground truth density change well, our method overall obtains the most plausible and natural density changes as observed in the side views. Our density values are slightly too small but match the ground truth density change well.

density change where positive density changes are indicated with white and negative density changes with red.

The most naïve density estimation expands the values of the 1D projection uniformly into depth, i.e., all cells in one line of sight contain the same value, as shown in Figure 4.9d. Such a density change does not represent natural fluid motions and is hence not suitable for estimating fluid motion. There is no divergence-free velocity field that is able to infer such density change by advecting the original smoke plume in Figure 4.9a forward. Adding a non-negativity constraint leads to a slightly more plausible density shape, as displayed in Figure 4.9e, where total density values below zero are prohibited. To favor local changes, a mask constraint is added, which concentrates density change in the surrounding cells of the original smoke plume as shown in Figure 4.9f. However, such a density change still does not represent the behavior of fluids like the original density change does in Figure 4.9c.

Furthermore, we investigate the failure of traditional methods for single-view 3D reconstructions in Figure 4.10 with slightly different test cases. Here, we show front (left) and side views (right) of the 3D density change. The ground truth density change in Figure 4.10a) exhibits natural distributions of negative and positive density changes representing the fluid's transporting motion. Most positive values are found at the top, since the smoke plume is rising upwards. In Figure 4.10b), we reconstruct the consecutive 3D density volume, subtract the current ground truth density plume, and visualize the resulting density change. This contrasts the previous 2D to 1D examples, where simply a projected density change is mapped into 2D space. When reconstructing a target 2D image naïvely, the image intensities are distributed uniformly into 3D space and as such, the front view looks very accurate while the side view reveals an unrealistic distribution, compare Figure 4.10b). Subtracting the original plume from the newly estimated uniform density field results in a large negative change inside the plume and a strong positive change in the surrounding.

When mapping the projected density change into 3D comparable to the 1D to 2D tests, the density change is slightly more realistic, see Figure 4.10c). Here, we use a regularized tomography including smoothness and kinetic energy penalties. However, both density changes b) and c) do not represent natural fluid motions and no divergence-free velocity field exists, that could induce or match these observations. Hence, we need to develop a novel basis for reconstructing density changes for highly under-determined reconstruction problems.

We propose a tight coupling of residual density, i.e., density change, and residual velocity, such that the residual density is induced by advection with a divergence-free velocity, i.e., the residual velocity. The result of such a joint density estimation is visualized in Figure 4.10d), which exhibits natural structures. Our method not only improves temporal coherence but also eliminates undesired stripe artifacts and uniform depth expansion as typically present in tomography approaches, as also observed previously in Figure 1.1. Our joint reconstruction is indispensable for inherently under-determined single-view reconstructions but also beneficial for accurate multi-view reconstructions.

4.2.2. Estimation Procedure

Although OF is not perfectly suited for dynamic and non-smooth fluid motions, it works well when applied to ground truth density volumes. When using full density volumes, the flow is corrected at each time step to match the ground truth, such that errors do not accumulate. In contrast to other works, we do not assume that accurate volumetric densities are available, but instead infer the densities together with the velocity by tightly coupling both quantities, such that we obtain robust reconstructions even for sparse input views. We estimate both quantities for a single time step t based on a given state for time $t-1$. Our full reconstruction procedure is outlined in Algorithm 3. This pseudocode illustrates the schematic overview of Figure 4.8, where advection, pressure projection, and diffusion are re-used from a common fluid simulator. First, both velocity and density are initialized (lines 2-3). The fluid is assumed to be at rest and hence, the velocity is simply set to zero. The density is estimated with a regular tomography solve, as outlined later in Algorithm 6 and Section 4.2.3. The remainder of our reconstruction method is described in the following paragraphs.

Prediction To ensure temporal coherence and reduce under-determination, we make use of a physics-based prediction scheme, similar to Gregson et al. [GITH14]. The prediction is illustrated in Figure 4.8 as the left gray box labeled with *predict*. The predicted velocity $\tilde{\mathbf{u}}^t$ is created by projecting the first advected previous velocity $\mathbf{u}^{t-1}$ to the divergence-free space and adding viscosity (line 5). The advection is denoted with $\mathcal{A}$ and is realized with MacCormack advection. The viscosity step is described in Section 3.1. The predicted density $\tilde{\Phi}^t$ is created by advecting it forward with the predicted velocity $\tilde{\mathbf{u}}^t$,

Algorithm 3 Full Reconstruction Procedure

```
1: function reconVelDen($i^t$)                                  ▷ reconstruct velocity and density
2:     $\mathbf{u}^0 = 0$;                                       ▷ init velocity
3:     $\Phi^0 =$ reconDen($i^0, 0$);                             ▷ init density, use Equation (4.5)
4:     for all t do
5:         $\tilde{\mathbf{u}}^t = \Pi_{\text{DIV}}\left(\mathcal{A}(\mathbf{u}^{t-1}, \mathbf{u}^{t-1})\right) +$ addViscosity($\mathbf{u}^{t-1}$);   ▷ predict velocity
6:         $\tilde{\Phi}_{\text{I}}^t =$ estimInfl($i^t, \Phi^{t-1}, \tilde{\mathbf{u}}^t$);   ▷ estimate density inflow
7:         $\tilde{\Phi}^t = \mathcal{A}(\Phi^{t-1} + \tilde{\Phi}_{\text{I}}^t, \tilde{\mathbf{u}}^t)$;   ▷ predict density
8:         $\Delta\mathbf{u}^t =$ reconVelMS($\tilde{\mathbf{u}}^t, \tilde{\Phi}^t, \Phi^{t-1}, i^t$);   ▷ solve for residual velocity Equation (4.2)
9:         $\mathbf{u}^t = \tilde{\mathbf{u}}^t + \Delta\mathbf{u}^t$;   ▷ finalize velocity
10:        $\Phi_{\text{I}}^t =$ estimInfl($i^t, \Phi^{t-1}, \mathbf{u}^t$);   ▷ estimate density inflow
11:        $\Phi^t = \mathcal{A}(\Phi^{t-1} + \Phi_{\text{I}}^t, \mathbf{u}^t)$;   ▷ finalize density
12:    end for
13: end function
```

where we first add an estimated density inflow $\tilde{\Phi}_{\text{I}}^t$ (lines 6-7). The inflow estimation is explained in Section 4.2.4 and outlined in pseudocode in Algorithm 8.

Residual Velocity and Density As mentioned above and as highlighted in orange and blue in Figure 4.8, the central pieces of our reconstruction procedure are the estimation of the residual velocity $\Delta\mathbf{u}^t$ and the estimation of the unseen density inflow Φ_{I}^t. Both steps ensure that the motion, density distribution, and density amount of the reconstruction match the given real-world input images i^t, which are depicted in the lower left of Figure 4.8. The residual velocity estimation is used in Algorithm 3 (line 8), is elaborated in detail in Section 4.2.3, and outlined in pseudocode in Algorithm 4.

The density is never modified directly except for the invisible inflow area. All changes on the density volume are induced by advecting it with the velocity field (line 11). Hence, the velocity must be calculated such that the advected density complies to all constraints. The physical constraints on the density have to transfer onto the residual velocity $\Delta\mathbf{u}^t$, such that the velocity will induce those constraints on the density field through advection. Therefore, the residual density $\Delta\Phi^t$ is only an intermediate variable required to enforce constraints on the density, which in turn constrain the residual velocity through our combined optimization scheme. After calculating the residual velocity, the residual density is discarded. The densities' interim character is illustrated by graying the corresponding steps in Figure 4.8 out. The residual density, however, is inherently important and without its iterative refinement, the reconstructed density and velocity are not as realistic as with our coupled method, which is demonstrated in Figure 5.6b in Chapter 5. The detailed estimation of the residual density is described in Section 4.2.3.

Final Quantities To create the final velocity $\mathbf{u}^t$, the predicted and the residual velocity $\Delta\mathbf{u}^t$ and $\tilde{\mathbf{u}}^t$ are accumulated (line 9). The final density Φ^t is generated by first adding a density inflow Φ_{I}^t to the previous density Φ^{t-1} and then advecting it with the final velocity $\mathbf{u}^t$ (lines 10-11).

Regularization Within our reconstruction procedure, we make use of smoothness and a kinetic energy (Tikhonov) regularizers for both density and velocity as is common practice in other works [WC11] to counteract the ill-posedness of an under-determined problem. As such, we make our problem convex [PB*14]. The smoothness and kinetic energy regularizers for both density and velocity are given by

$$\begin{aligned} E_{\text{smooth}}(\Delta\mathbf{u}^t) &= \tfrac{1}{2}\left\|\nabla(\Delta\mathbf{u}^t)\right\|^2, & E_{\text{smooth}}(\Delta\Phi^t) &= \tfrac{1}{2}\left\|\nabla(\Delta\Phi^t)\right\|^2, \\ E_{\text{kin}}(\Delta\mathbf{u}^t) &= \tfrac{1}{2}\left\|\Delta\mathbf{u}^t\right\|^2, & E_{\text{kin}}(\Delta\Phi^t) &= \tfrac{1}{2}\left\|\Delta\Phi^t\right\|^2, \end{aligned} \tag{4.1}$$

where the discrete forms are explained in Section 3.3. We denote the Laplacian matrix for scalar quantities with L and for vector quantities with L_j, where j indicates the j-th vector component.

Those regularizing energies are weighted with scalar parameters α and β for smoothness and kinetic energy, respectively, for velocity and density: $(\alpha_{\mathbf{u}}, \beta_{\mathbf{u}}, \alpha_{\Phi}, \beta_{\Phi}) = (1.1, 0.11, 0.55, 0.0011)$. It is important to choose moderate weights, as too large weights lead to overly smooth and too small residual densities and velocities due to outweighing the objective physical constraints. The choice of weights strongly depends on the quantities' values, which in turn, e.g., depend on the image brightness and volume resolution for density and velocity, respectively. We determined our weights empirically.

4.2.3. Velocity Estimation

We couple the reconstruction of density Φ^t and velocity $\mathbf{u}^t$ through the transport equation. In our reconstruction method, we estimate a residual velocity $\Delta\mathbf{u}^t$ in combination with a residual density $\Delta\Phi^t$, accounting for the required change to meet the target input images i^t based on predicted velocity $\tilde{\mathbf{u}}^t$ and density $\tilde{\Phi}^t$. The transport equation hence states that the unknown residual density $\Delta\Phi^t$ must be induced by advecting the current density $\tilde{\Phi}^t$ with the unknown residual velocity $\Delta\mathbf{u}^t$. The transport equation is equivalent to the brightness constancy assumption, see Section 3.3. Additional physical constraints include that the velocity must be divergence-free and that the sum of the predicted and residual densities, $\tilde{\Phi}^t$ and $\Delta\Phi^t$, must be non-negative as well as its projection must match the input images i^t. We set the velocity inflow to a constant c, which is approximating the average speed of the rising plume observed in the input images.

The corresponding optimization problem is given by

$$\begin{aligned} \underset{\Delta\Phi^t, \Delta\mathbf{u}^t}{\text{minimize}} \quad & f(\Delta\mathbf{u}^t, \Delta\Phi^t) = \left\|\Delta\Phi^t + \nabla\tilde{\Phi}^t \cdot \Delta\mathbf{u}^t\right\|^2 \\ \text{subject to} \quad & P\,\Delta\Phi^t = i^t - \tilde{i^t}, \quad \Delta\Phi^t + \tilde{\Phi}^t \geq 0, \\ & \Delta\mathbf{u}^t|_I = c - \tilde{\mathbf{u}}^t|_I, \quad \nabla\cdot\Delta\mathbf{u}^t = 0, \end{aligned} \tag{4.2}$$

where $f(\Delta\Phi^t, \Delta\mathbf{u}^t)$ is our convex objective function with both $\Delta\Phi^t$ and $\Delta\mathbf{u}^t$ unknown, $\Delta\Phi^t$ is the residual density, and as such, the finite difference for approximating $\frac{\partial\tilde{\Phi}^t}{\partial t}$, P is the matrix projecting the 3D volume to each 2D image, and $\tilde{i^t}$ is the projected predicted density $\tilde{\Phi}^t$. The dimensions of P are $N_p \times N_v$ where N_p and N_v denote the total number of pixels and voxels, respectively. The derivation for Equation (4.2) is described in Appendix B.

To enforce both image formation and non-negativity constraints on the residual density $\Delta\Phi^t$, we compute the smallest density correction $\Delta\Phi^t_{cor}$ that, when added to $\Delta\Phi^t$, ensures that $\Delta\Phi^t$ complies to both remaining constraints. Hence, the residual density $\Delta\Phi^t$ is projected onto the space of density fields that match the image formation and result in a non-negative total density. Our second minimization problem concerning the density correction $\Delta\Phi^t_{cor}$ is given by

$$\begin{aligned} \underset{\Delta\Phi^t_{cor}}{\text{minimize}} \quad & h(\Delta\Phi^t_{cor}) = \left\| P\Delta\Phi^t_{cor} - (i^t - \tilde{i}^t - P\Delta\Phi^t) \right\|^2 \\ \text{subject to} \quad & \Delta\Phi^t_{cor} + \Delta\Phi^t + \tilde{\Phi}^t \geq 0. \end{aligned} \tag{4.3}$$

This can equally be formulated for the residual density

$$\begin{aligned} \underset{\Delta\Phi^t}{\text{minimize}} \quad & h(\Delta\Phi^t) = \left\| P\Delta\Phi^t - (i^t - \tilde{i}^t) \right\|^2 \\ \text{subject to} \quad & \Delta\Phi^t + \tilde{\Phi}^t \geq 0 \end{aligned} \tag{4.4}$$

and for the density itself

$$\begin{aligned} \underset{\Phi^t}{\text{minimize}} \quad & h(\Phi^t) = \left\| P\Phi^t - i^t \right\|^2 \\ \text{subject to} \quad & \Phi^t \geq 0. \end{aligned} \tag{4.5}$$

We find the optimal $\Delta\Phi^t_{cor}$ by making use of Algorithm 7 for the tomography problem in h and simultaneously enforcing the non-negativity constraint as a projection by employing a second PD loop. Ihrke and Magnor [IM04] achieve non-negative density values by simply setting negative entries of their CGLS solver in each iteration to zero. We make use of PD to simultaneously ensure image formation and non-negativity constraints. Note that the residual density and residual density correction are allowed to be negative while the total density, i.e., the sum of predicted, residual, and correcting density, must be in the non-negative orthant $\mathbb{R}^N_{\geq 0}$, where N is the number of cells $N = n_x * n_y * n_z$.

To account for large motion displacements, we employ a multi-scale scheme typically used for OF, see [MLPK13]. We employ a hierarchical approach solving from coarse to fine spatial scales, as outlined

Algorithm 4 MS Velocity Update, solve Equation (4.2)

```
1:  function RECONVELMS($\tilde{\mathbf{u}}^t, \tilde{\Phi}^t, \Phi^{t-1}, i^t$)    ▷ calculate residual velocity on multiple scales
2:      if grid size large enough then
3:          $\Delta\mathbf{u}^t_C = \Pi_{DIV}(\text{UP}(\text{RECONVELMS}(\text{DOWN}(\tilde{\mathbf{u}}^t), \text{DOWN}(\tilde{\Phi}^t), i^t)))$;
4:          $\tilde{\mathbf{u}}^t_{new} = \tilde{\mathbf{u}}^t + \Delta\mathbf{u}^t_C$;    ▷ add coarse residual to prediction
5:          $\tilde{\Phi}^t_{I,new} = \text{ESTIMINFL}(i^t, \Phi^{t-1}, \tilde{\mathbf{u}}^t_{new})$;    ▷ density inflow
6:          $\tilde{\Phi}^t_{new} = \mathcal{A}(\Phi^{t-1} + \tilde{\Phi}^t_{I,new}, \tilde{\mathbf{u}}^t_{new})$;    ▷ adapt prediction
7:          $\Delta\mathbf{u}^t = \Delta\mathbf{u}^t_C + \text{RECONVEL}(\tilde{\mathbf{u}}^t_{new}, \tilde{\Phi}^t_{new}, i^t)$;    ▷ sum up velocity residuals on different scales
8:      else
9:          $\Delta\mathbf{u}^t = \text{RECONVEL}(\tilde{\mathbf{u}}^t, \tilde{\Phi}^t, i^t)$;
10:     end if
11:     return $\Delta\mathbf{u}^t$;
12: end function
```

Algorithm 5 Velocity Update, solve Equation (4.2)

```
1:  function RECONVEL($\tilde{\mathbf{u}}^t, \tilde{\Phi}^t, i^t$)    ▷ calculate residual velocity on single scale
2:      for $k \leftarrow 1, 10$ do    ▷ Prima-Dual iterations
3:          ▷ shared part: advection equation, $\mathbf{x}^k = \mathbf{x}^k_{\mathbf{u}} \cup \mathbf{x}^k_{\Phi}$
4:          $\mathbf{x}^k = \mathbf{x}^{k-1} + \sigma_f \mathbf{y}^{k-1} - \sigma_f$ SOLVELSE($A_f + \sigma_f I, \mathbf{x}^{k-1} + \sigma_f \mathbf{y}^{k-1} - b_f$);
5:          ▷ velocity part: inflow and divergence-free on $\mathbf{z}^k_{\mathbf{u}}$
6:          $\Delta\mathbf{u}^t_{\text{tmp}} = \mathbf{z}^{k-1}_{\mathbf{u}} - \tau_{\Delta\mathbf{u}} \mathbf{x}^k_{\mathbf{u}}$;    ▷ create temporary residual velocity
7:          $\Delta\mathbf{u}^t_{\text{tmp}}|_I = c - \tilde{\mathbf{u}}^t|_I$;    ▷ set velocity inflow
8:          $\mathbf{z}^k_{\mathbf{u}} = \Pi_{\text{DIV}}(\Delta\mathbf{u}^t_{\text{tmp}})$;    ▷ make div-free
9:          ▷ density part: tomo. and non-neg. (Equation (4.3)) on $\mathbf{z}^k_{\Phi}$
10:         $\Delta\Phi^t_{\text{tmp}} = \mathbf{z}^{k-1}_{\Phi} - \tau_{\Delta\Phi} \mathbf{x}^k_{\Phi}$;    ▷ create temporary residual density
11:         $\mathbf{z}^k_{\Phi} = \Delta\Phi^t_{\text{tmp}} +$ RECONDEN($i^t, \tilde{\Phi}^t + \Delta\Phi^t_{\text{tmp}}$);    ▷ add density correction
12:         ▷ shared variable update, $\mathbf{y}^k = \mathbf{y}^k_{\mathbf{u}} \cup \mathbf{y}^k_{\Phi}$
13:         $\mathbf{y}^k = \mathbf{z}^k + \theta_f(\mathbf{z}^k - \mathbf{z}^{k-1})$;
14:     end for
15:     return $\mathbf{z}^k_{\mathbf{u}}$;
16: end function
```

in pseudocode in Algorithm 4. Coarse quantities are upsampled and advected forward where fine-scale solves further refine the reconstructed residual velocity. The multi-scale scheme employs the single-scale residual velocity calculation, which is illustrated in Algorithm 5. In order to calculate the residual velocity accurately on each scale based on the predicted density $\tilde{\Phi}^t_{\text{new}}$ incorporating velocities from lower scales $\tilde{\mathbf{u}}^t_{\text{new}}$, we calculate and add an intermediate inflow density $\tilde{\Phi}^t_{\text{I,new}}$.

Proximal Operators We solve our joint optimization problem efficiently with PD, as outlined in Section 3.2.2. Our shared objective function f in Equation (4.2) is quadratic, hence we formulate the proximal operator as described in Section 3.2:

$$\mathbf{prox}_{f,\sigma_f}(\xi) = (A_f + \sigma_f I)^{-1}(\sigma_f \xi - b_f), \tag{4.6}$$

where

$$A_f = \begin{bmatrix} (\nabla\tilde{\Phi}^t)(\nabla\tilde{\Phi}^t)^T & (\nabla\tilde{\Phi}^t)^T \\ (\nabla\tilde{\Phi}^t) & I \end{bmatrix} + \begin{bmatrix} \alpha_{\mathbf{u}} \sum_j L_j & 0 \\ 0 & \alpha_{\Phi} L \end{bmatrix} + \begin{bmatrix} \beta_{\mathbf{u}} I & 0 \\ 0 & \beta_{\Phi} I \end{bmatrix} \quad \text{and} \quad b_f = 0.$$

L is the Laplacian matrix and ξ is the concatenation of residual velocity $\Delta\mathbf{u}^t$ and density $\Delta\Phi^t$, i.e., $\xi = (\xi_{\Delta\mathbf{u}}, \xi_{\Delta\Phi})$. If a variable acts only on either velocity or density, we indicate this by a corresponding subscript. The PD convergence parameters $(\sigma_f, \tau_f, \theta_f)$ differ for velocity and density. The joint convergence parameters are again the concatenation of both separate ones for residual velocity and density $(\sigma_f, \tau_f, \theta_f) = ((\sigma_{\Delta\mathbf{u}}, \sigma_{\Delta\Phi}), (\tau_{\Delta\mathbf{u}}, \tau_{\Delta\Phi}), (\theta_{\Delta\mathbf{u}}, \theta_{\Delta\Phi}))$. We employ a standard CG solver in the proximal operator in Equation (4.6). This proximal operator is used in Algorithm 5 (line 4).

The constraints that f must comply to, are all separate orthogonal projections for both unknowns

$\Delta\mathbf{u}^t$ and $\Delta\Phi^t$. Therefore, we split the second proximal operator $\mathbf{prox}_{g,1/\tau_g}$ into two distinct proximal operators for residual velocity $\mathbf{prox}_{g,1/\tau_{\Delta\mathbf{u}},\,\Delta\mathbf{u}}$ and density $\mathbf{prox}_{g,1/\tau_{\Delta\Phi},\,\Delta\Phi}$ indicated by subscripts. The proximal operator concerning velocity ensures the incompressibility of $\Delta\mathbf{u}$:

$$\mathbf{prox}_{g,1/\tau_{\Delta\mathbf{u}},\,\Delta\mathbf{u}}(\xi_{\Delta\mathbf{u}}) = \Pi_{\text{DIV}}(\xi_{\Delta\mathbf{u}}). \tag{4.7}$$

As explained in Section 3.2, the incompressibility constraint is an orthogonal projection onto the space of divergence-free velocities, is denoted with Π_{DIV}, implemented via a standard CG pressure solver, and employed in Algorithm 5 (line 8). Before enforcing divergence-free velocities, we set the inflowing residual velocity to the difference between the prescribed constant c and the predicted velocity $\tilde{\mathbf{u}}^t$ as described in Algorithm 5 (line 7).

The proximal operator $\mathbf{prox}_{g,1/\tau_{\Delta\Phi},\,\Delta\Phi}$ projects the residual density onto the space of densities that match the image formation model and lead to a non-negative total density. We simply add a density correction to the residual density to enforce both constraints:

$$\mathbf{prox}_{g,1/\tau_{\Delta\Phi},\,\Delta\Phi}(\xi_{\Delta\Phi}) = \xi_{\Delta\Phi} + \Delta\Phi^t_{cor}. \tag{4.8}$$

In order to obtain the density correction $\Delta\Phi^t_{cor}$, we solve Equation (4.3) with a separate PD loop. The first proximal operator solves the quadratic objective function h:

$$\mathbf{prox}_{h,\sigma_h,\Delta\Phi_{cor}}(\xi_{\Delta\Phi_{cor}}) = (A_h + \sigma_h I)^{-1}(\sigma_h \xi_{\Delta\Phi_{cor}} - b_h), \tag{4.9}$$

where

$$A_h = P^T P + \alpha_\Phi L + \beta_\Phi I \quad \text{and} \quad b_h = P^T \Delta i^t_{cor} = P^T(i^t - \tilde{i}^t - P\Delta\Phi^t).$$

This is exactly the regularized tomography.

Enforcing the non-negativity constraint is an orthogonal projection onto the space of non-negative N-dimensional real-valued vectors $\mathbb{R}^N_{\geq 0}$. This projection is efficiently realized by simply setting the values in $\Delta\Phi^t_{cor}$ to the minimal value such that $\Delta\Phi^t_{cor} + \Delta\Phi^t + \tilde{\Phi}^t \geq 0$. The corresponding proximal operator, here indicated with l, is given by:

$$\mathbf{prox}_{l,1/\tau_h,\,\Delta\Phi_{cor}}(\xi_{\Delta\Phi_{cor}}) = \Pi_{\text{NN}}(\xi_{\Delta\Phi_{cor}}). \tag{4.10}$$

A pseudocode description of density estimation is outlined in Algorithm 6, where the procedure is called RECONDEN$(i^t, \Phi^t_{\text{IN}})$. This tomography solve is used for initialization of the density field and as a component of the residual velocity estimation in Algorithm 5.

The convergence parameters of each PD loop, which we determined empirically, are set to the constant values
$(\sigma_{\Delta\mathbf{u}}, \tau_{\Delta\mathbf{u}}, \theta_{\Delta\mathbf{u}}, \sigma_{\Delta\Phi} = \sigma_h, \tau_{\Delta\Phi} = \tau_h, \theta_{\Delta\Phi} = \theta_h) = (0.01, 10, 1, 0.01, 100, 1)$. These parameters control the speed of convergence, are problem-specific, and are likewise applied to all our reconstructions.

Algorithm 6 Tomography, solve Equation (4.5), Equation (4.4), or (4.3)

```
1: function reconDen($i^t$, $\Phi^t_{\mathrm{IN}}$)    ▷ reconstr. density, $\Phi^t_{\mathrm{IN}}$ can be 0
2:     $\Delta i^t = i^t - P\Phi^t_{\mathrm{IN}}$;    ▷ calculate residual image
3:     for $k \leftarrow 1, 10$ do    ▷ Prima-Dual iterations solving Equation (4.4)
4:         ▷ solve least squares problem, e.g., with CGLS (Algorithm 7)
5:         $\mathbf{x}^k = \mathbf{x}^{k-1} + \sigma_h \mathbf{y}^{k-1} - \sigma_h$ solveLSE($A_h + \sigma_h I, \mathbf{x}^{k-1} + \sigma_h \mathbf{y}^{k-1} - b_h$);
6:         $\mathbf{z}^k = \Pi_{\mathrm{NonNeg}}(\Phi^t_{\mathrm{IN}} + \mathbf{z}^{k-1} - \tau_h \mathbf{x}^k)$;    ▷ cut off negative values
7:         $\mathbf{y}^k = \mathbf{z}^k + \theta_h(\mathbf{z}^k - \mathbf{z}^{k-1})$;
8:     end for
9:     return $\mathbf{z}^k$;
10: end function
```

Discretization The standard Horn-Schunck OF and previous OF approaches [GITH14] discretize the transport equation on a co-located grid. However, pressure solvers perform suboptimally on co-located grids and introduce checkerboard artifacts or do not remove all divergence for non-smooth velocity fields. As outlined in Section 3.1, the divergence of a velocity is best eliminated on a staggered MAC grid. Instead of using a co-located grid for the OF part and a staggered grid for the pressure solver, which introduces smoothing and interpolation errors, we employ a staggered grid for our complete reconstruction pipeline.

Visual Hull In practice, it is disadvantageous to apply a visual hull straightforward as described in Section 3.4, where voxels influencing a black pixel are excluded from the system of equations. Black pixels are pixels with intensities less or equal to a threshold, e.g., $\epsilon_{\mathrm{vh}} = 1 \times 10^{-9}$. On the one hand, a voxel's weight could be very small, and as such, the pixel would still be assumed to be black although the voxel contains a relevant amount of smoke density. On the other hand, if pixels are small compared to voxels, multiple pixels could be influenced by a similar range of voxels while exhibiting intensities varying from dark to bright, as our real world is continuous. Excluding voxels of either category from the system unnecessarily limits the solution space, which could result in wrongful over-determination of the system. First, we only consider pixels as truly black if their neighboring pixels inside a given window are black as well. For an image resolution of 600×1062 and a volume resolution of 100×177, we use a window size of 31, which we found to perform best. Moreover, a voxel is only ever marked as outside the system of equations if its influence on a black pixel is above 10 %. Lastly, we only include voxels in our system if at least two cameras see that voxel (in the case of multi-views). As such, we reduce unsupervised motions in the under-determined areas.

Regularized Tomography

When solving Equation (4.9) with a common CG solver, it is necessary to explicitly calculate the matrix product $P^T P$. However, $P^T P \in \mathbb{R}^{N_v \times N_v}$ is significantly larger than $P \in \mathbb{R}^{N_p \times N_v}$ due to the sparsity of the views $N_p \ll N_v$. Calculating and storing $P^T P$ is infeasible for larger resolutions. Therefore, we avoid computing $P^T P$ directly by employing a CGLS solver like [IM04], which allows for feasible run times.

Algorithm 7 Regularized CGLS

```
1: function solveCGLSReg($P, R, i^t, \xi_\Phi, \sigma_h$)
2:     $r_0 = \sigma_h \xi_\Phi - P^T i^t - P^T P x_0 - (R + \sigma_h I) x_0$;
3:     $p_0 = r_0$;
4:     while accuracy not reached do
5:         $\alpha_k = \|r_{k-1}\|^2 / (p_{k-1}^T P^T P p_{k-1} + p_{k-1}^T (R + \sigma_h I) p_{k-1})$;
6:         $x_k = x_{k-1} + \alpha_k p_{k-1}$;
7:         $r_k = r_{k-1} - \alpha_k * P^T P p_{k-1} + \alpha_k (R + \sigma_h I) p_{k-1}$;
8:         $\beta_k = \|r_k\|^2 / \|r_{k-1}\|^2$;
9:         $p_k = r_k + \beta_k * p_{k-1}$;
10:    end while
11: end function
```

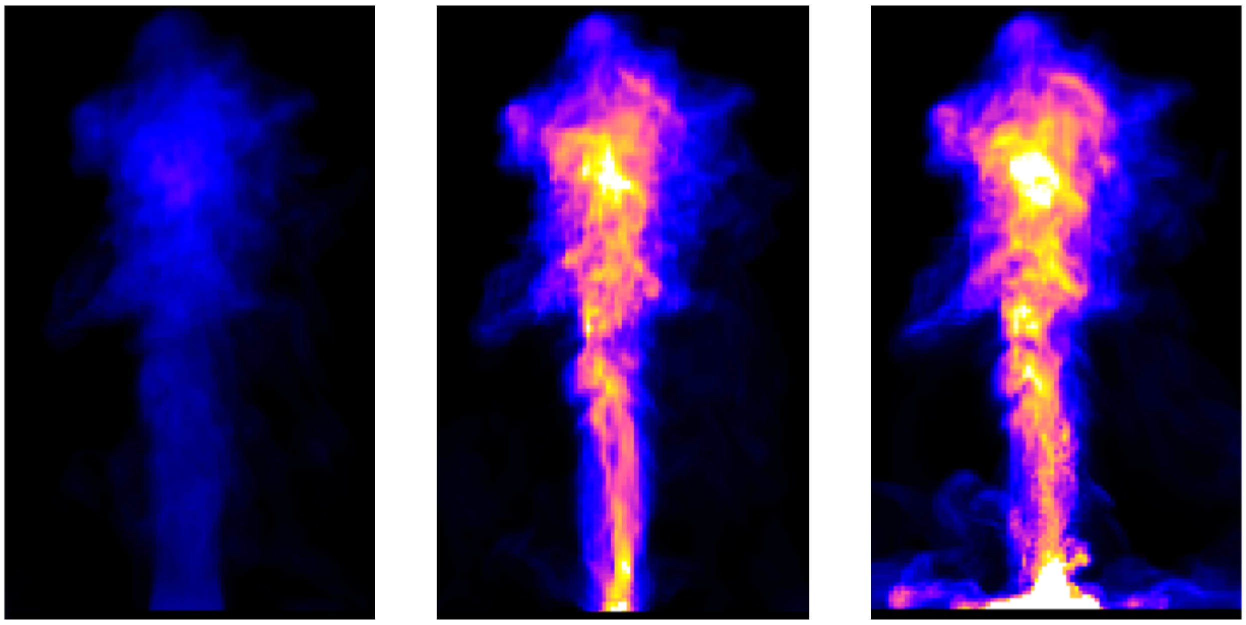
A performance comparison is shown in Section 5.1.3. We use our regularized CGLS solver to solve each tomography problem of our reconstruction pipeline.

For ill-posed inverse problems such as sparse tomography, regularizers are crucial to obtain a stable solution. For example, it is not guaranteed that every voxel in the grid is hit by a pixel's ray. Including a smoothness regularizer ensures even density distribution. However, it is not straightforward to incorporate square and symmetric regularizing matrices $R \in \mathbb{R}^{N_v \times N_v}$ in a CGLS solver as commonly done for regular CG solvers. In Algorithm 7, we show how to include a very sparse, regularizing matrix of smoothness and kinetic energy regularizers R in a CGLS solver. As R contains only seven entries per row in 3D (diagonal and neighboring cells for smoothness, see Equation (4.1)), it can be stored and multiplied efficiently. We apply R to the residual and step size calculations. Additionally, the weighted diagonal matrix $\sigma_h I$ and the PD variable updates $\sigma_h \xi_\Phi$, see Equation (4.9), are added to the system's matrix and to the right-hand side, respectively.

4.2.4. Density Inflow Estimation

Just as for common numerical forward simulations, suitable boundary conditions are crucial for our reconstruction technique. The invisible inflow region, i.e., the amount of density outflux of our smoke container, has a big influence on our overall reconstruction quality. Underestimating the amount of density inflow leads to a plume unable to fill the desired volume in later stages of the reconstruction, as demonstrated in Figure 4.11a. Overestimating, on the other hand, yields strong instabilities and evasive motions to the sides of the plume, see Figure 4.11c. Figure 4.11b demonstrates that our inflow estimation solver leads to stable plumes nicely filling the target smoke volume.

The information in our image data is limited as the density injection from one time step to another is not directly visible. In order to calculate the required amount of density, we project the current target images i^t into the volume to obtain the target density field $\Phi^t_{\text{tar}} = \text{reconDen}(i^t, 0)$. Our goal is to estimate a density inflow Φ^t_{I} that, when added to the previous density field Φ^{t-1}, leads to the right amount of density in Φ^t. Φ^t_{I} has non-zero entries only in the invisible inflow area Γ_I. The region that traces back into the inflow region Γ_I while considering $\mathbf{u}^t$ is denoted with Γ_R, where R stands for



(a) Too few density inflow. **(b)** Our density inflow estimation. **(c)** Too much density inflow.

Figure 4.11.: These examples highlight the importance of the density inflow for reconstructing smoke plumes. Depending on the amount of density inflow, the reconstructed plumes contains too less density a), a suitable amount of density b), or excessive density inflow resulting in unstable behavior and evasive motions c).

reachable. The part of the inflow area, which is reached through back tracing from the visible domain, is denoted with Γ_{RI}, i.e., the *reached inflow area*. These domain areas are visualized in Figure 4.7.

Advecting the combination of the previous density Φ^{t-1} and the density inflow Φ_{I}^t with the known velocity $\mathbf{u}^t$ should result in a density field where the accumulated visible density intensities match the accumulated visible density intensities of the target density Φ_{tar}^t. This constraint is derived and formulated as follows

$$\begin{aligned}
\sum_{e\in\Gamma\setminus\Gamma_I} \mathcal{A}(\Phi^{t-1}+\Phi_{\mathrm{I}}^t,\mathbf{u}^t) &\overset{!}{=} \sum_{e\in\Gamma\setminus\Gamma_I} \Phi_{\mathrm{tar}}^t && |\triangleright \text{ assume linear advection}\\
\sum_{e\in\Gamma\setminus\Gamma_I} \mathcal{A}(\Phi^{t-1},\mathbf{u}^t) + \sum_{e\in\Gamma\setminus\Gamma_I} \mathcal{A}(\Phi_{\mathrm{I}}^t,\mathbf{u}^t) &\overset{!}{=} \sum_{e\in\Gamma\setminus\Gamma_I} \Phi_{\mathrm{tar}}^t && |\triangleright \text{ move to other side}\\
\sum_{e\in\Gamma\setminus\Gamma_I} \mathcal{A}(\Phi_{\mathrm{I}}^t,\mathbf{u}^t) &\overset{!}{=} \sum_{e\in\Gamma\setminus\Gamma_I} \left(\Phi_{\mathrm{tar}}^t - \mathcal{A}(\Phi^{t-1},\mathbf{u}^t)\right) && |\triangleright \text{ define } \Delta\Phi_{\mathrm{tar}}^t\\
\sum_{e\in\Gamma_R} \mathcal{A}(\Phi_{\mathrm{I}}^t,\mathbf{u}^t) &\overset{!}{=} \sum_{e\in\Gamma\setminus\Gamma_I} \Delta\Phi_{\mathrm{tar}}^t, && (4.11)
\end{aligned}$$

where $\mathcal{A}(\cdot,\mathbf{u}^t)$ denotes the advection with the final velocity $\mathbf{u}^t$, the visible domain is $\Gamma\setminus\Gamma_I$, e is a voxel element in the volume, $\Delta\Phi_{\mathrm{tar}}^t = \Phi_{\mathrm{tar}}^t - \mathcal{A}(\Phi^{t-1},\mathbf{u}^t)$ is the residual target density field, and Γ_R is the part of the visible domain that is influenced by advecting the inflow area Γ_I with $\mathbf{u}^t$. When advecting Φ_{I}^t with $\mathbf{u}^t$, the amount of smoke densities in the visible area $\Gamma\setminus\Gamma_I$ should be equal to the amount of densities in the residual target density $\Delta\Phi_{\mathrm{tar}}^t$.

Algorithm 8 Estimate inflow, such that Equation (4.12) holds

```
1: function ESTIMINFL($i^t, \Phi^{t-1}, \mathbf{u}^t$)
2:     $\Phi^t_{\text{tar}} = $ RECONDEN($i^t, 0$);                              ▷ target density
3:     $\Delta\Phi^t_{\text{tar}} = \Phi^t_{\text{tar}} - \mathcal{A}(\Phi^{t-1}, \mathbf{u}^t)$;                  ▷ residual density
4:     $A_m = A_I^T A_I + \alpha_{\Phi_I} L + \beta_{\Phi_I} I$
5:     $b_m = \max\left(\Delta\Phi^t_{\text{tar}}|_{\Gamma_R} + d, -\mathcal{A}(\Phi^{t-1}, \mathbf{u}^t)|_{\Gamma_R}\right)$
6:     for $k \leftarrow 1, 10$ do                                       ▷ Prima-Dual iterations
7:         $\mathbf{x}^k = \mathbf{x}^{k-1} + \sigma_m \mathbf{y}^{k-1} - \sigma_m$ SOLVELSE($A_m + \sigma_m I, \mathbf{x}^{k-1} + \sigma_m \mathbf{y}^{k-1} - b_m$);
8:         $\mathbf{z}^k = \Pi_{\text{NN}}(\Phi^{t-1} + \mathbf{z}^{k-1} - \tau \mathbf{x}^k)$;
9:         $\mathbf{y}^k = \mathbf{z}^k + \theta(\mathbf{z}^k - \mathbf{z}^{k-1})$;
10:    end for
11:    EXTRAPOLATESRCVALUES($\mathbf{z}^k$);
12:    return $\mathbf{z}^k$;
13: end function
```

In order to solve for Φ^t_{I}, we set up a linear system of equations based on $\mathbf{u}^t$, i.e., we solve $A_I \Phi^t_{\text{I}} = b_I$ for Φ^t_{I}. We discretize the advection operator $\mathcal{A}$ via a semi-Lagrangian method to obtain A_I. It holds that $A_I \in \mathbb{R}^{|\Gamma_{RI}| \times |\Gamma_R|}$, since we look for density values in the reached inflow area Γ_{RI} based on density values in the reachable visible domain Γ_R. Hence, $A_I = \mathcal{A}(\Phi^t_{\text{I}}, \mathbf{u}^t)|_{\Gamma_{RI} \text{ to } \Gamma_R}$.

In total, the right-hand-side vector b_I must contain the missing density summed over each cell e in the domain $\Gamma \setminus \Gamma_I$, i.e., $\sum_{e \in \Gamma \setminus \Gamma_I} \Delta\Phi^t_{\text{tar}}$. However, as $b_I \in \mathbb{R}^{|\Gamma_R|}$, it covers only a subset of the domain, i.e., the reachable area Γ_R. Therefore, we set b_I to the missing density value in $\Delta\Phi^t_{\text{tar}}|_{\Gamma_R}$ plus a constant d, which accounts for further density discrepancies in the rest of the domain $\Delta\Phi^t_{\text{tar}}|_{\Gamma \setminus (\Gamma_R \cup \Gamma_I)}$. Possible choices for d are outlined below. Furthermore, if the total missing density is negative, we also need to ensure that b_I is never less than $-\mathcal{A}(\Phi^{t-1}, \mathbf{u}^t)$, which would lead to negative and hence unrealistic densities in the visible domain after advecting $\Phi^{t-1} + \Phi^t_{\text{I}}$.

Additionally, we ensure that the final density inflow $(\Phi^{t-1} + \Phi^t_{\text{I}})|_{\Gamma_I}$ is non-negative, which we found crucial for plausible and numerically stable reconstructions. Our optimization problem is given by

$$\begin{aligned} \underset{\Phi^t_{\text{I}}}{\text{minimize}} \quad & m(\Phi^t_{\text{I}}) = \left\| A_I \Phi^t_{\text{I}} - \max\left(\Delta\Phi^t_{\text{tar}}|_{\Gamma_R} + d, -\mathcal{A}(\Phi^{t-1}, \mathbf{u}^t)|_{\Gamma_R}\right) \right\|^2 \\ \text{subject to} \quad & \Phi^{t-1} + \Phi^t_{\text{I}} \geq 0. \end{aligned} \tag{4.12}$$

The whole inflow estimation is outlined in pseudocode in Algorithm 8. To ensure subsequent advection steps make use of smooth inflow densities, we slightly extrapolate the inflow values in Γ_{RI} into cells that were excluded from the solve, see line 11 of Algorithm 8. Here, we average the values of the surrounding 3×3 neighbors that were included in the inflow estimation.

Proximal Operators We formulate the first proximal operator for Equation (4.12) as following

$$\mathbf{prox}_{m,\sigma_m}(\xi) = (A_m + \sigma_m I)^{-1}(\sigma_m \xi - b_m), \tag{4.13}$$

where

$$A_m = A_I^T A_I + \alpha_{\Phi_\mathrm{I}} L + \beta_{\Phi_\mathrm{I}} I \quad \text{and} \quad b_m = \max\left(\Delta\Phi^t_{\mathrm{tar}}|_{\Gamma_R} + d,\ -\mathcal{A}(\Phi^{t-1}, \mathbf{u}^t)|_{\Gamma_R}\right).$$

We solve this system of equations with a CGLS solver, just as for the tomography. However, as this system's matrix is much smaller, i.e., $A_I \in \mathbb{R}^{|\Gamma_{RI}| \times |\Gamma_R|}$, its costs are negligible compared to the rest of the optimization procedure. The choice of regularizing weights is $(\alpha_{\Phi_\mathrm{I}}, \beta_{\Phi_\mathrm{I}}) = (0.2, 0.02)$ while the PD convergence parameters are $(\sigma_m = \sigma_{\Delta\Phi} = \sigma_h, \tau_m = \tau_{\Delta\Phi} = \tau_h, \theta_m = \theta_{\Delta\Phi} = \theta_h) = (0.01, 100, 1)$.

Inflow Visual Hull Similar to the visual hull for tomography problems, we remove inflow cells known to be zero from the system to prevent spurious density in cells that should be empty. Otherwise, halo-like densities possibly accumulate over time. We mark target density cells in Γ_R as outside the system if their density value in Φ^t_{tar} is below 0.01. Inflow cells, that, when advected, would end up in such cells, are also marked as outside the system of equations, but only if their contributing weight is above 1×10^{-5}. We determined these thresholds experimentally and use them for all our reconstructions.

Choice of d The most naïve distribution of the total missing density $\Delta\Phi^t_{\mathrm{tar}}|_{\Gamma \setminus (\Gamma_R \cup \Gamma_I)}$, which can be positive or negative, is to assign each target cell the same value. This leads to the following definition of d:

$$d = \frac{\sum\limits_{\Gamma \setminus (\Gamma_R \cup \Gamma_I)} \Delta\Phi^t_{\mathrm{tar}}}{|\Gamma_R|}. \tag{4.14}$$

However, we chose a more sophisticated approach by using varying values d_e for each cell $e \in \Gamma_R$, which we set in relation to the existing value in $\Delta\Phi^t_{\mathrm{tar}}|_e$. Hence, d is not uniform for all cells, but individual for each cell e. A larger value in $\Delta\Phi^t_{\mathrm{tar}}|_e$ leads to larger d_e where 'larger' means more positive, not larger in its absolute value. We introduce an offset o, which ensures that $\Delta\Phi^t_{\mathrm{tar}}|_e$ is greater or equal to zero for each cell, i.e., $\Delta\Phi^t_{\mathrm{tar}}|_e + o \geq 0$. In doing so, we can treat 'larger' in terms of higher absolute value and as such, achieve correct proportional scaling of positive and negative values:

$$d_e = \frac{\sum\limits_{\Gamma \setminus (\Gamma_R \cup \Gamma_I)} \Delta\Phi^t_{\mathrm{tar}}}{\sum\limits_{\Gamma_R} (\Delta\Phi^t_{\mathrm{tar}} + o)} (\Delta\Phi^t_{\mathrm{tar}}|_e + o), \tag{4.15}$$

where only truly greater values attract larger portions of the missing density. The offset o is zero or the absolute value of the largest negative entry in $\Delta\Phi^t_{\mathrm{tar}}|_{\Gamma_R}$, which means $o = |\min(0, \Delta\Phi^t_{\mathrm{tar}}|_{\Gamma_R})|$.

In this chapter, we described our hardware setup and our powerful reconstruction method with all its components. The results and accuracy of our multi- and single-view reconstructions obtained with this method will be evaluated in the following Chapters 5 and 6.

5 Accurate Multi-View Reconstruction

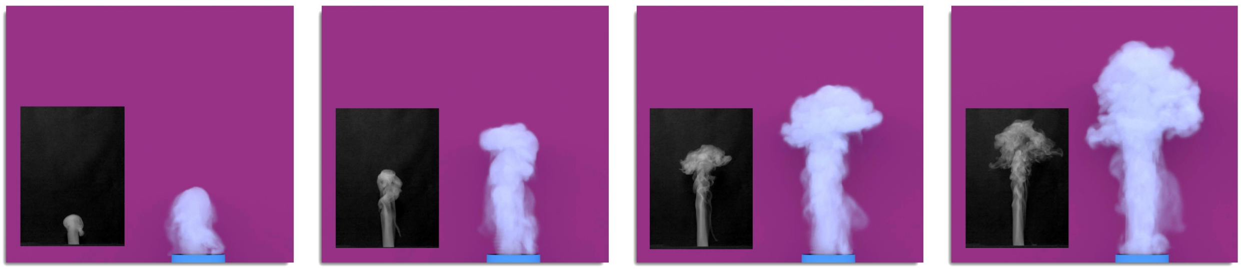


Figure 5.1.: A reconstructed density plume from our *ScalarFlow* data set. Four frames are re-rendered as thick smoke, which demonstrates the flexibility of a 3D data set. The corresponding input images, i.e., the real-world references, are shown as insets in the lower left corners.

We create our data set *ScalarFlow* by gathering various accurate multi-view reconstructions of real-world smoke plumes, which we record with five cameras as described in Section 4.1. Our reconstruction method is described in detail in Section 4.2. In order to verify our method's accuracy, we first reconstruct synthetically simulated smoke plumes, for which we have access to ground truth density and velocity. We furthermore compare our reconstructions to previous methods. Then, we illustrate samples from *ScalarFlow* and discuss a perceptual user study as demonstration of a first application. The results for accurate multi-view reconstructions are shown in our video[1] and on the *ScalarFlow* website [2].

Our data set contains 104 reconstructions with 150 frames of 3D density, 3D velocity, 2D rendered images, and 2D input images. The 3D quantities have a resolution of $100 \times 178 \times 100$, while the rendered images have a resolution of 600×1062 and the input images have a resolution of 1080×1920. All of our reconstructions use the same camera calibration, which is given as one text file per camera and contains the ray's starting points and directions for each pixel.

Each reconstruction is approximately 10 GB large and lives in its own folder, which contains thumbnail images and .mp4 videos summarizing the reconstructed quantities visually, a folder 'input' , and a folder 'reconstruction'. The 'input' folder encloses the original recorded videos and the extracted, unprocessed, and post-processed images as .jpg and .npz files. We store the 2D images as 3D arrays where the images are stacked along the z-axis. The 'reconstruction' folder contains .npz files for density,

[1]`https://youtu.be/J0y8zakWQ4Y`
[2]`https://ge.in.tum.de/publications/2019-scalarflow-eckert/`

Statistics of *ScalarFlow*					
	min	**max**	**mean**	**std. deviation**	**% of nz cells**
all					
density	0	74.488	0.136	1.140	8.39
velocity	0	9.007	0.078	0.195	91.26
images	0	5.773	0.216	0.599	34.10
000010					
density	0	50.68	0.158	1.202	9.66
velocity	0	7.544	0.080	0.211	91.26
images	0	4.608	0.253	0.653	38.23
000030					
density	0	60.350	0.134	1.099	8.51
velocity	0	7.018	0.080	0.200	91.26
images	0	3.958	0.212	0.581	34.09
000060					
density	0	55.067	0.122	1.050	7.55
velocity	0	5.067	0.0676	0.177	91.26
images	0	4.232	0.190	0.547	31.38
000090					
density	0	55.064	0.140	1.149	8.47
velocity	0	7.286	0.084	0.205	91.26
images	0	4.667	0.221	0.601	34.87

Table 5.1.: Statistics of ScalarFlow.

velocity, and rendered and input images in its original form and with cropped inflow section. For cropping the inflow, we delete the density and velocity in the lower 15 cells. The quantities with cropped inflow area are additionally visualized as .jpg. The visualization of each 104 reconstructed density seen from the front and side views are shown in Figure 5.2 and 5.3. In each 'reconstruction' folder, we additionally store a 'description.json' file where all relevant grid names, parameter values, and meta data are listed. Furthermore, the python script used for initiating the reconstruction is stored.

Table 5.1 shows the max, min, mean, and standard deviation values over all reconstructions and for some individual samples. We calculate the Euclidean norm of the velocity. Since many values are zero, we also indicate the average percentage of cells with non-zero values.

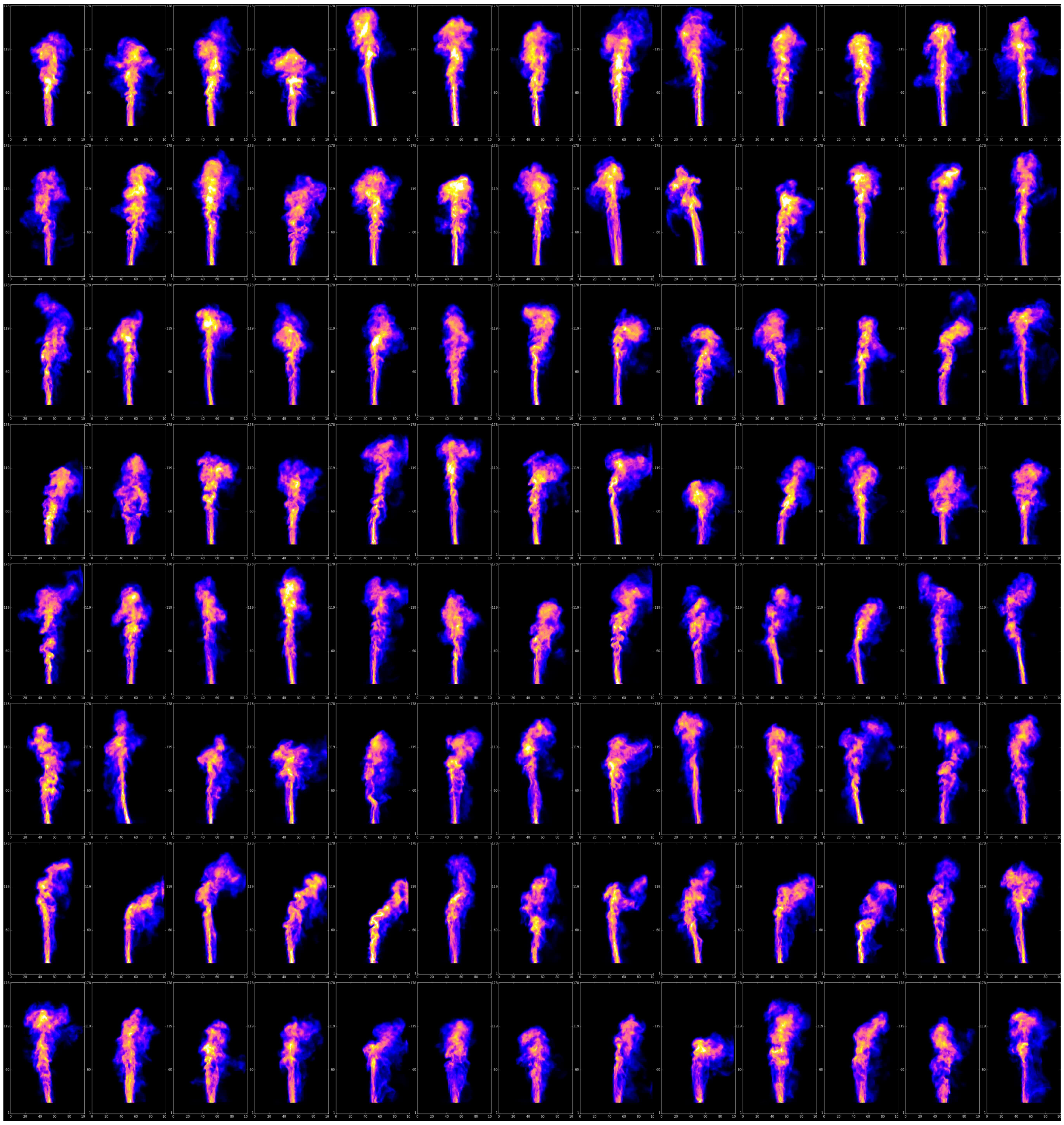


Figure 5.2.: Overview of the 104 reconstructed densities in ScalarFlow, front view.

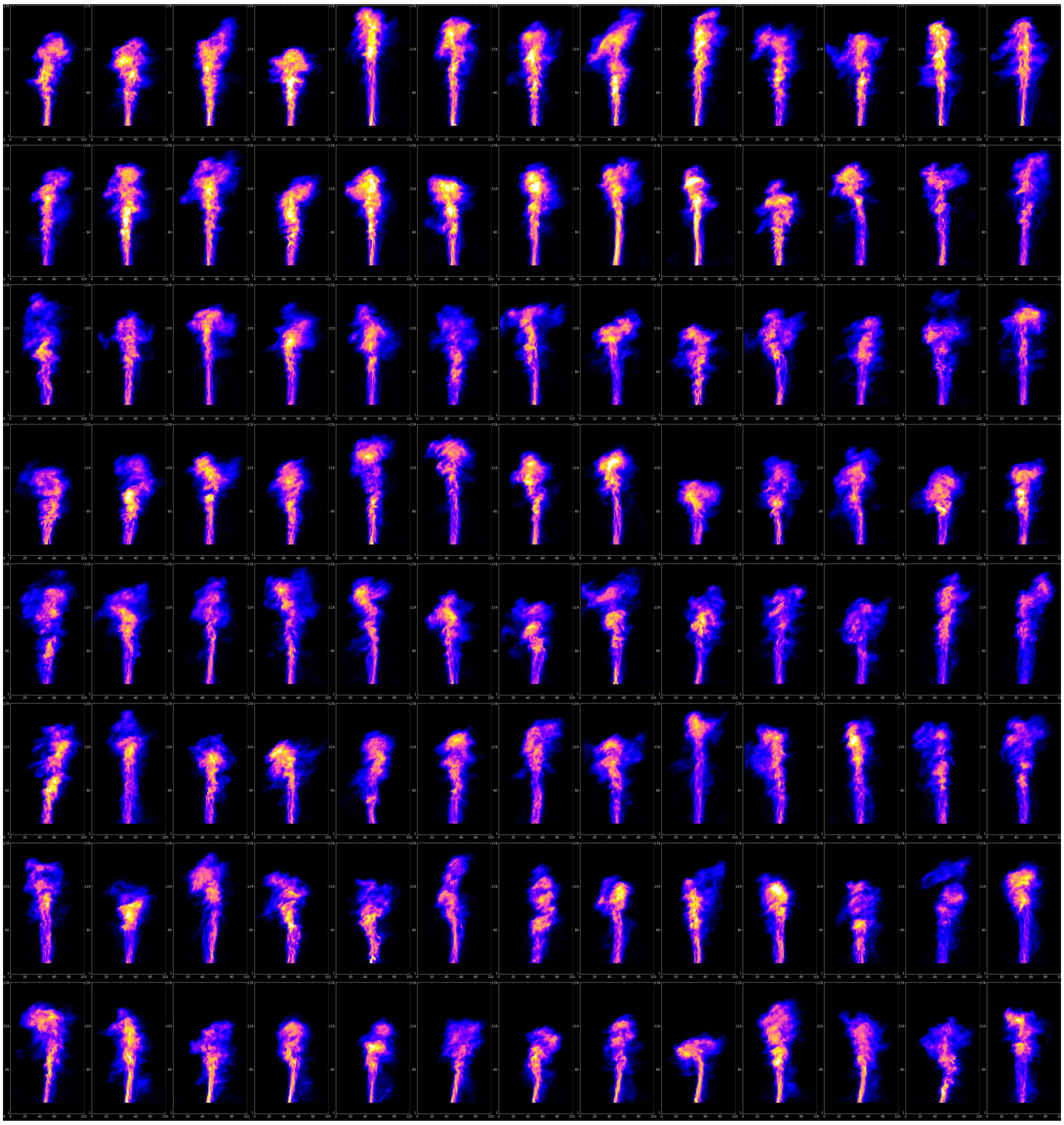
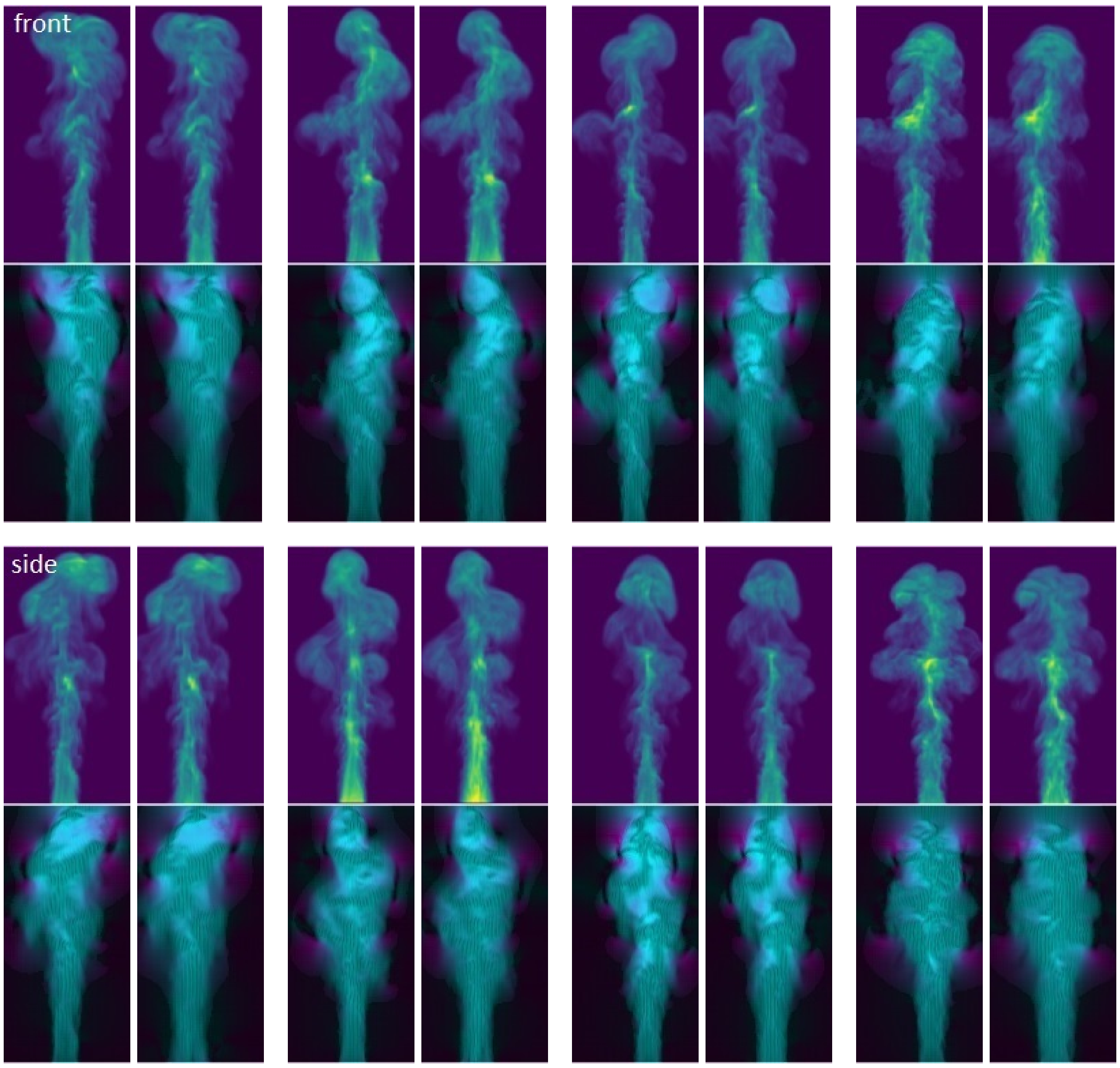


Figure 5.3.: Overview of the 104 reconstructed densities in ScalarFlow, side view.



(a) Base simulation, $t = 170$.

(b) Larger inflow area, $t = 145$.

(c) 50 % more buoyancy, $t = 122$.

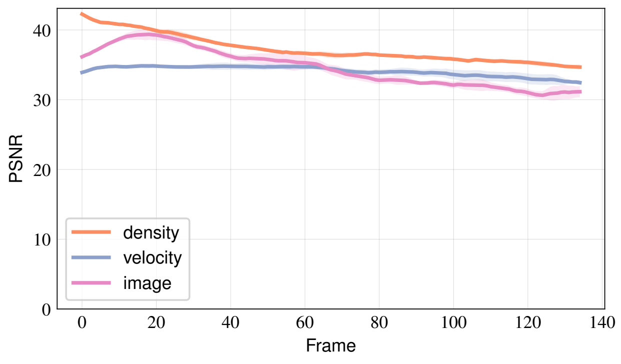
(d) High-resolution, $t = 155$.

Figure 5.4.: Four different synthetic smoke simulations with ground truth density (upper left) and reconstructed density (upper right), with ground truth velocity (lower left) and reconstructed velocity (lower right), front (top rows) and side views (bottom rows). The high-resolution simulation and reconstruction in d) are downsampled to facilitate visual comparison. We use our real-world calibration data to generate the synthetic input images and use our inflow estimation for reconstruction. Independent of varying simulation parameters as well as resolution, our reconstructions recover both density and motion accurately.

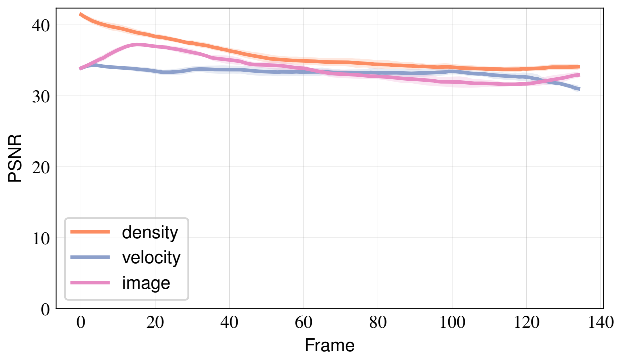
5.1. Evaluation

In order to validate the accuracy of our novel reconstruction technique, we simulate multiple and varying smoke plumes, for which we have access to the 3D density and velocity. We evaluate the reconstruction accuracy, compare our results to alternative methods, and elaborate computational resource consumption.

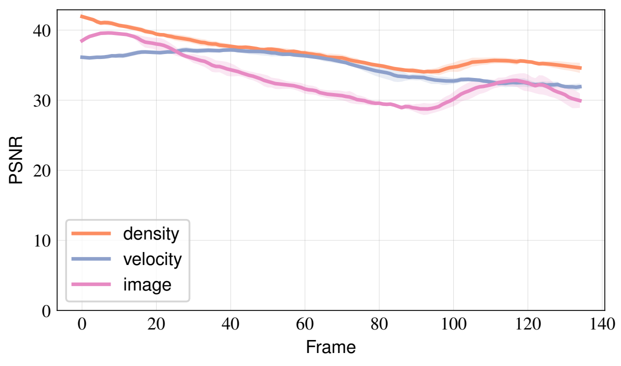
5.1.1. Reconstruction Accuracy



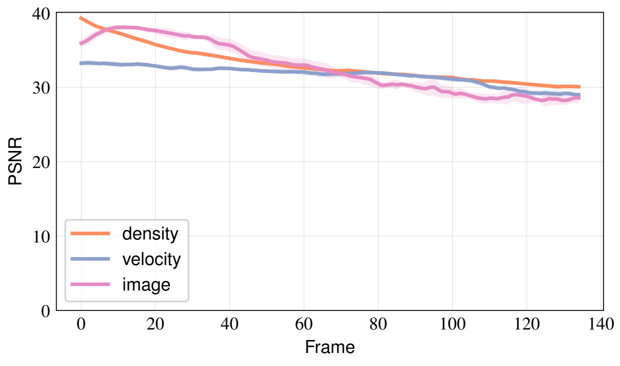
(a) Base Simulation.



(b) Larger inflow area.



(c) 50 % more buoyancy.



(d) High-resolution.

Figure 5.5.: Mean and standard deviation of PSNR values of density, velocity, and image differences for our four synthetic reconstructions in Figure 5.4 with five instances for each.

In order to examine the accuracy of our reconstruction method, we simulate different smoke plumes and reconstruct them with our full method outlined in Section 4.2 without exploiting any additional information from our synthetic setup, such as the ground truth inflow. With changing simulation parameters, grid resolution, and noise, we evaluate the robustness of our method to handle unknown and varying behavior as in real-world flows. By increasing the size of the smoke source, the amount of buoyancy, and simulation resolution, we generate different flow dynamics with a variety of fluid motion, as shown in Figure 5.4. Furthermore, we add different noise to the density sources and, as such, create five different instances of each simulation setup. To generate input images of the synthetic simulations, we render the synthetic density volumes by using our implicit camera calibration data from five real cameras as described in Section 4.1.6.

Our base simulation in Figure 5.4a is simulated with the Boussinesq approximation on a $100 \times 177 \times 100$ grid. Our reconstructions use the same grid resolution. We measure the accuracy of our reconstructed quantities velocity, density, and images by calculating the PSNR of the difference between ground truth and reconstructed quantity, since the PSNR is an established quality measurement of compressed images. The PSNR values are closely related to the standard l2-norm, but use a logarithmic scale and are normalized against the maximum value of the ground truth quantity. In Figure 5.5a, we visualize the mean PSNR values and its standard deviation (light halo around mean) over all five simulation instances of our baseline comparison. The average PSNR values over time are (37.3, 34.1, 34.6) for density, velocity, and images, respectively. Even the highly under-determined side view is reconstructed accurately, as illustrated in the bottom rows of Figure 5.4a.

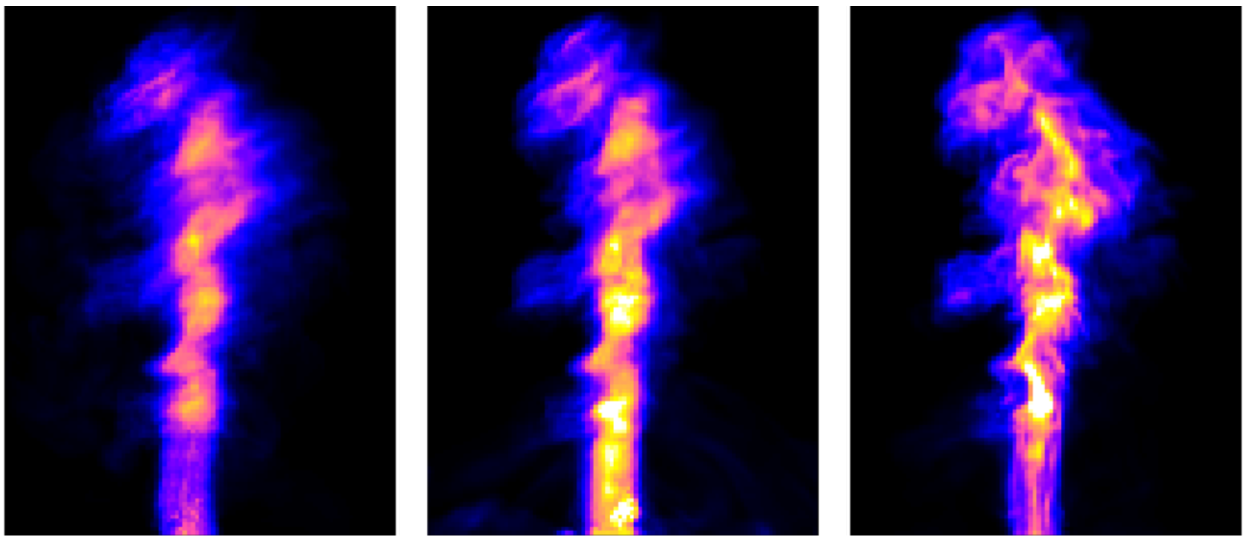
Enlarging inflow size and buoyancy amount leads to significantly different flow behaviors, as demonstrated in Figures 5.4b and 5.4c. However, our algorithm accurately reconstructs these smoke plumes as well, both in terms of density and velocity. Comparable PSNR values are obtained, as shown in Figures 5.5b and 5.5c. The corresponding PSNR values for density, velocity, and images are (35.7, 33.3, 33.8) and (36.8, 34.9, 33.1), respectively. Doubling the buoyancy leads to much faster rising plumes, which can be observed in Figure 5.4c where the plume reaches the top of the domain early at $t = 122$. Since smoke exiting the finite reconstruction domain poses a challenge to reconstruction, the PSNR values exhibit a slight valley as observed in Figure 5.5c. Our three tests a), b), and c) yield distinct data points in the space of buoyant smoke plumes. The comparable PSNR values for all three indicate that our algorithm is capable of robustly and accurately reconstructing a variety of different single-phase flows.

Increasing the simulation's resolution to $200 \times 354 \times 200$ while keeping the reconstruction's resolution the same, imitates the challenging real-world conditions, since space in our real world is continuous, i.e., infinitely high resolved. Furthermore, a standard forward fluid simulator would not be able to recreate the motion of high-resolved simulations on a lower grid. Our reconstruction method, however, captures both density and velocity accurately as visualized in Figure 5.4d. The PSNR values for density, velocity, and images are (33.0, 31.6, 32.6).

Our reconstructions contain slightly less overall density, but properly capture the intricate shapes of the complex reference flows, as demonstrated in Figure 5.4. All our tests robustly result in comparable PSNR values for density, velocity, and images over multiple time steps, as shown in Figure 5.5. The PSNR values slightly drop over time as the flow becomes more complex over time and by occupying a larger volume in the domain, but yield high averages overall. For example, the PSNR values for our baseline simulation at $t = 150$ still indicate accurate reconstructions with (34.7, 32.5, 31.1). These test results imply that our simple capturing setup with five camera views is sufficient for our strong physics-based optimization to capture realistic flow behavior matching the given input images.

5.1.2. Alternative Methods

In addition to synthetic evaluations, we compare our approach to previous work by Gregson et al. [GITH14], referred to as V_a and a simpler version of our method, called V_b. For V_a, we first reconstruct the volumetric densities via tomography, then apply a divergence-free OF velocity estimation. In



(a) V_a: divergence-free optical flow reconstruction based on tomographic densities from previous works.

(b) V_b: our method without iterating over coupled reconstruction of residual velocity and density.

(c) Our full reconstruction method.

Figure 5.6.: Visual comparison of three methods reconstructing volumetric density. Our tightly coupled reconstruction method in c) is required to obtain a realistic and detailed fluid motion without unnatural artifacts.

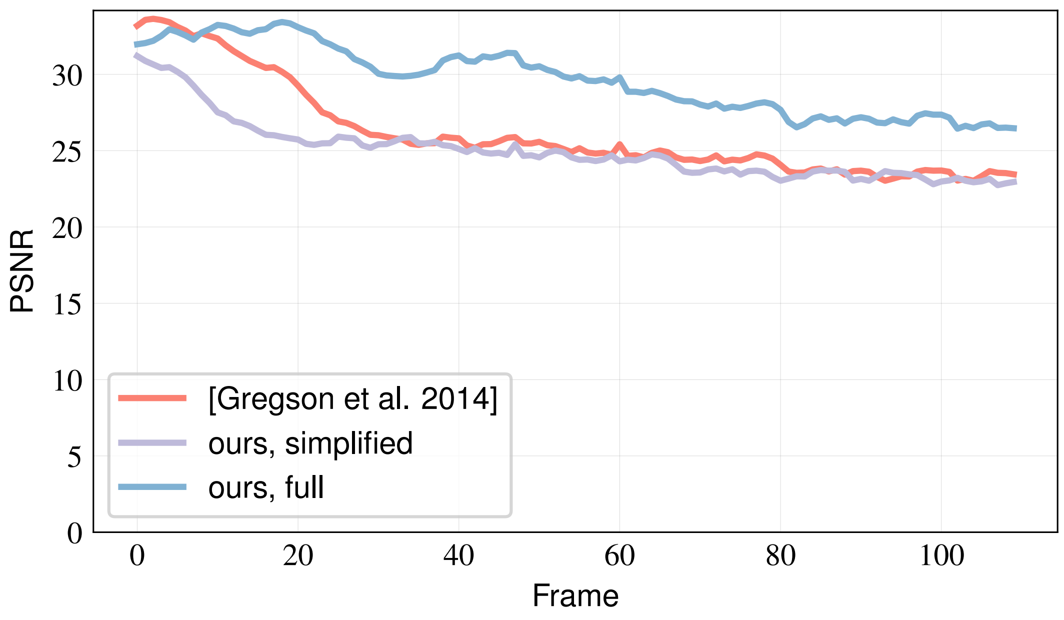


Figure 5.7.: PSNR values for the difference between input and re-rendered reconstructed density for V_a (tomography plus 3D optical flow), V_b (simplified version of our method), and our full approach.

our simplified version V_b, we do not iterate between residual velocity and density and hence, remove their coupling. This leads to an advection-unaware density update, which does not change per iteration of the residual velocity estimation, in contrast to Equation (4.2). Instead, the residual density is fixed for each time step and is retrieved from a single tomography step.

As demonstrated in Figure 5.6, our method is able to reconstruct natural, detailed, and artifact-free fluid-like motion, while V_a and V_b are less detailed and exhibit spurious smoke densities outside the plume. Especially V_a exhibits the typical stripe artifacts along the camera's line of sights. On the other hand, as the densities reconstructed with V_b deviate from the images over time, the solver is incapable of reconstructing more than two thirds of the overall sequence. These quality differences are also notable in the averaged PSNR values of 2D image differences, namely $25.8, 25.0$, and 29.2 for V_a, V_b, and our method, respectively. The PSNR values over time are illustrated in Figure 5.7. Hence, our method recreates the target images with the highest accuracy.

This evaluation indicates that our iteratively computed residual density lies within the image formation null-space, but matches fluid motions undoubtedly better than density fields generated with simpler or previous methods. The reason beneath are the improved physical constraints finding more realistic solutions from the said null-space. Hence, our method produces more natural motions, density distributions without the typical tomographic artifacts, and is capable of handling under-constrained problems.

5.1.3. Performance Evaluation of Regularized Conjugate Gradient Least Squares

As outlined in Section 4.2.3, we employ a CGLS solver to avoid computing and storing $P^T P$ explicitly. In our multi-view reconstruction setting, a regular CG solver would require on 535 s per tomography solve on average and 182 s for matrix construction. Our CGLS solver obtains the same residual accuracy by only requiring 69 s per solve and 13 s for setting up the matrix. Instead of being a bottleneck, our tomography solver now takes up a smaller part of the overall run time, which is 809 s and 350 s for CG and CGLS, respectively. Both solvers are equally regularized with the matrix R. Furthermore, the CG solver needs up to 15 GB of memory for storing $P^T P$, while CGLS requires only 2.5 GB. As both memory and computational resource consumption grows super-linearly when increasing the grid resolution, our CGLS is a crucial component for reasonable run times of higher resolution reconstructions.

5.2. Reconstructions in *ScalarFlow*

Our *ScalarFlow* data set contains 100 fluid flow reconstructions with a resolution of $100 \times 177 \times 100$ and with 150 to 200 time steps. As such, our data set contains approximately twenty-six billion data voxels in total. A selection of 20 reconstructions is shown in Figure 5.10. As shown, our data set contains similar smoke plumes with a variety of interesting and natural variations and as such, thoroughly sample a chosen space of physical behavior. For example, some captures exhibit secondary plumes forming at various stages and separating from the main plume.

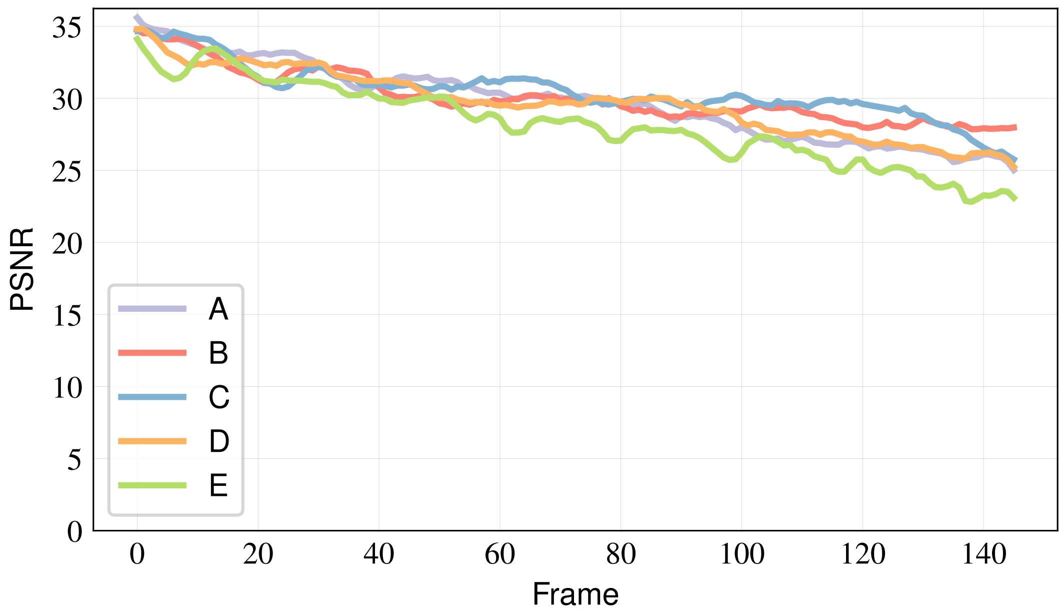


Figure 5.8.: Robust PSNR values of image differences for five different real-world reconstructions.

As we provide 3D data of both velocity and density, our collected flows can be used flexibly, e.g., for method evaluations, comparisons, re-simulations, data-driven applications, or visualizations. Four plume sequences of *ScalarFlow* are shown in Figure 5.9. With these renderings, we demonstrate an example of flexible visualization, namely rendering the smoke thickly on a colored background to highlight fine-scale details. The density visualization in Figure 5.10 is closer to the real-world recordings except for the bright colormap, which facilitates the observation of density details. We evaluate the accuracy of our real-world reconstructions by computing the PSNR values between input images and the rendered captured densities, just as for our comparison of alternative methods in Figure 5.7. Figure 5.8 shows that our reconstructions match the captured images well with average PSNR values between 27.4 and 29.7. All of our five exemplary reconstructions produce comparable error measurements.

5.3. Perceptual Evaluation

As a first application of our data set, we conduct a perceptual evaluation of established fluid simulation methods as well as varying resolutions via user studies where 189 non-experts from 48 countries participate. The participants are recruited via crowd-sourcing. Key requirements for robust and reliable evaluation results are the access to ground truth or reference data and the availability of a large number of different instances of the same phenomenon. To ensure objective answers, we design our studies simple and clear by showing two simulated smoke phenomena next to a reference video. The participants are asked to select the video, which looks closer to the reference. As such, we adopt the two-alternative forced choice (2AFC) design [Fec60] and calculate scores with the Bradley-Terry

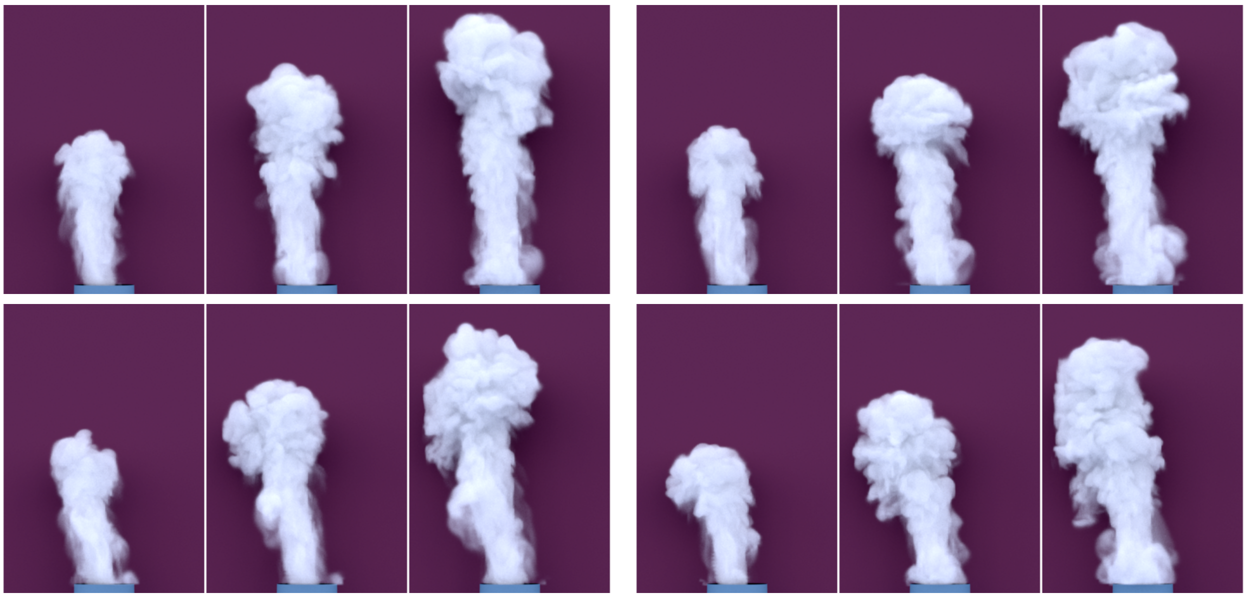


Figure 5.9.: Four reconstruction sequences rendered as thick smoke. To demonstrate the flexibility that accompanies 3D data sets, we visualize the captured plumes very differently to their real-world counterparts. This rendering style highlights the amount of small-scale details present in our data set.

model [Hun04, UHT17]. We randomize the questions, where each participants answers every question twice. In summary, we obtained 100 answers per question and hence, gather statistically relevant results.

First, we study well established fluid solvers regarding their capability of reproducing realistic flows. We evaluate semi-Lagrangian advection [Sta99], MacCormack advection [SFK*08], advection-reflection [ZNT18], and lastly, wavelet turbulence [KTJG08] as a representative for up-res methods. Although these methods have been analyzed visually in the corresponding publications, we are not aware of any evaluation of these methods against a real-world reference. To facilitate comparison, we implemented each simulation method in the same fluid solver framework [TP19] and employ the same initial conditions. Simulations are conducted on $100 \times 177 \times 100$ grids where the base resolution of wavelet turbulence version was halved first in order to enable the resulting synthetic turbulence to be added with a 2× upsampling. A visual overview over the four simulation methods and one specific instance of our data set are shown in Figure 5.11.

The result of our 2AFC user study of comparing simulation methods is summarized in Figure 5.12a. As shown, there is a 64 % chance that viewers prefer the advection-reflection method over MacCormack advection in terms of their similarity to our real-world reconstruction.

Due to the large amount of available reconstructions in *ScalarFlow*, we are able to compare such simulation methods to multiple real flows. We conduct four different studies A', 'B', 'C', and 'D' of the same methods but with a different reference data to evaluate the robustness of the users' preferences and to prevent outliers to impede insights. The joint preference is 'All'. Although the real smoke plumes differ visually, a range of similar salient flow features appear across all of them. We found

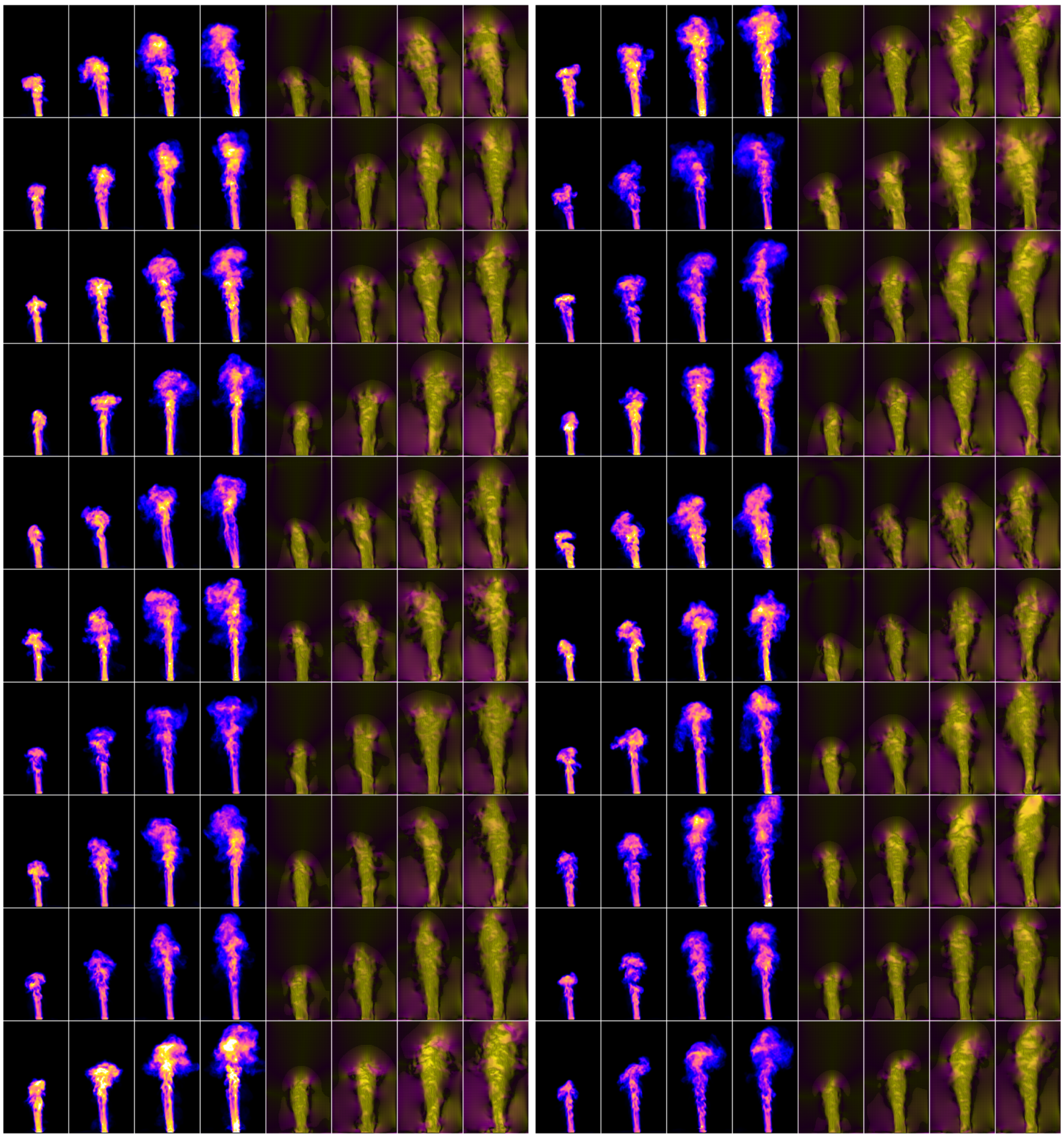


Figure 5.10.: A selection of 20 captured flows from *ScalarFlow* visualized in terms of density and velocity. Each row on both columns shows four frames of a single capture sequence.

our evaluation results to be stable and robust, as the preferences of all four studies in Figure 5.12a are highly correlated with $\rho \simeq 0.98$ ($P \simeq 0.02$) on average. In this context of user studies, we refer with ρ and P to the correlation coefficient and its p-value, respectively. In summary, our non-expert participants surprisingly prefer the relatively old and procedural wavelet turbulence method alike to the advection-reflection solver. MacCormack advection is slightly outperformed by both, while semi-Lagrangian advection is the least preferred simulation method, probably due to its high dissipation.

Furthermore, we investigate the influence of grid resolution on the realism of simulated flows. As explained in Section 3.1, fluid simulators in the graphics area typically do not explicitly add viscosity but rely on unknown amounts of numerical viscosity instead. With this study, we examine the amount of numerical viscosity present in forward simulations and aim at finding a suitable grid resolution, which produces the most similar fluid behavior compared to our reference data. In order to simulate five simulations on different resolutions, we use the MacCormack advection method as a *de facto* standard advection scheme. Our base resolution is $50 \times 88 \times 50$, which we increase by factors of $2\times$, $4\times$, $8\times$, and $12\times$, where the highest resolution is $600 \times 1062 \times 600$ as visualized in Figure 5.13. Note that the resolution of the y-direction is always $1.77\times$ x-resolution.

Our user study results in Figure 5.12b show that the $8\times$ and $12\times$ resolutions are considered closest to our real-world reference data. Interestingly, $8\times$ performs slightly better, which indicates that $8\times$ exhibits a similar amount of viscosity as in real-world smoke plumes of an approximate size of 0.9 m. This result vice versa indicates that our reconstructed flows contain visual detail and dynamics that are comparable to those of finely resolved simulations as the reconstructions use $64\times$ fewer voxels than the $8\times$ version. Hence, high-resolution simulations are required for numerical simulations to capture the behavior of real-world smoke plumes.

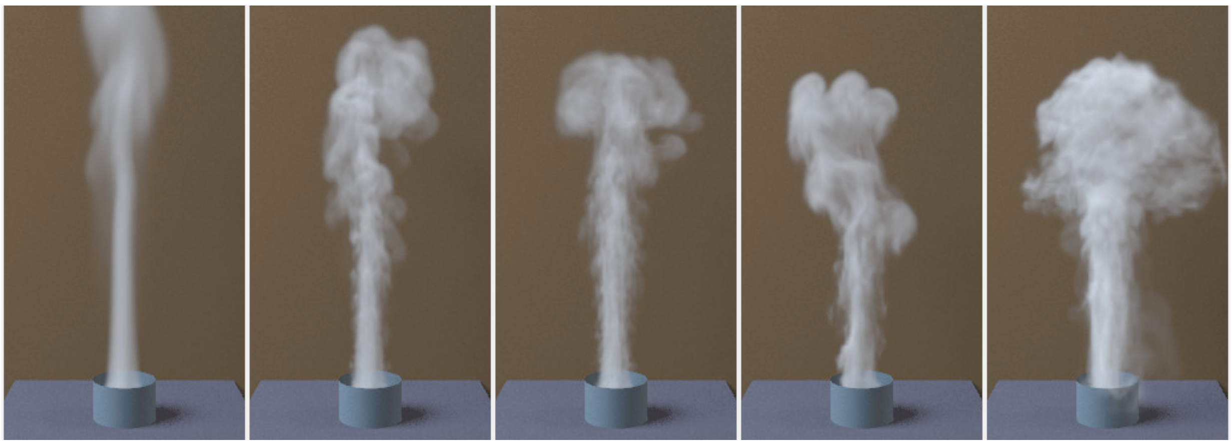


Figure 5.11.: Four different simulation methods and one *ScalarFlow* data set as used for the user studies. F.l.t.r.: Semi-Lagrangian, MacCormack, advection-reflection, wavelet turbulence, and reconstruction.

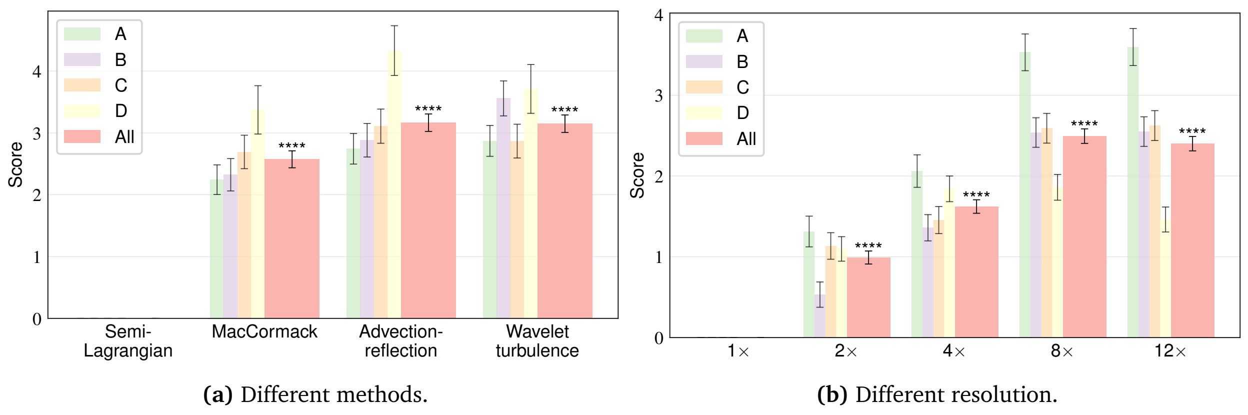


Figure 5.12.: Four different evaluations 'A', 'B', 'C', and 'D' of established simulation methods and varying simulation resolutions via user studies. 'All' denotes the combination of all evaluations. $^{****}P < 0.0001$.

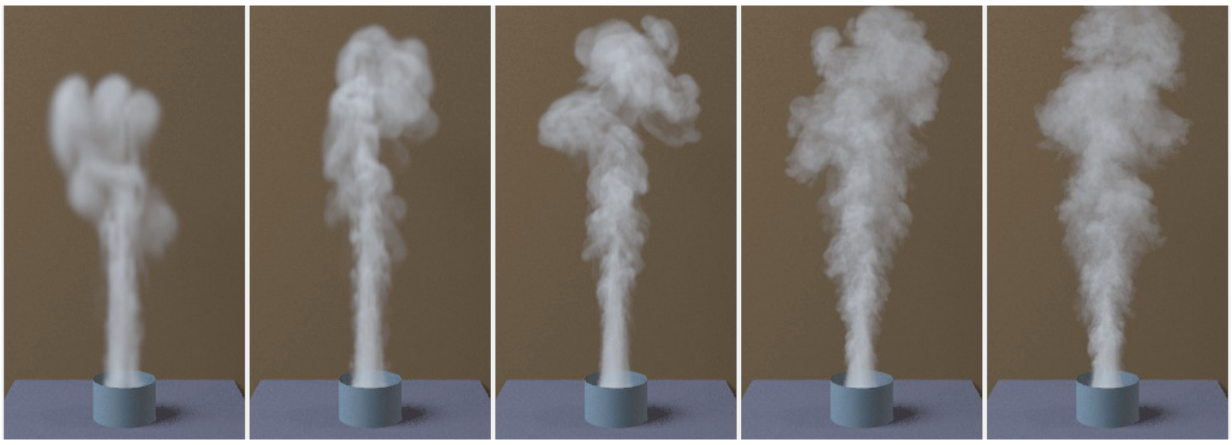


Figure 5.13.: Simulations on different grid resolutions: 1×, 2×, 4×, 8×, and 12×, where the base resolution is 50 × 88 × 50.

6 Single-View Reconstruction

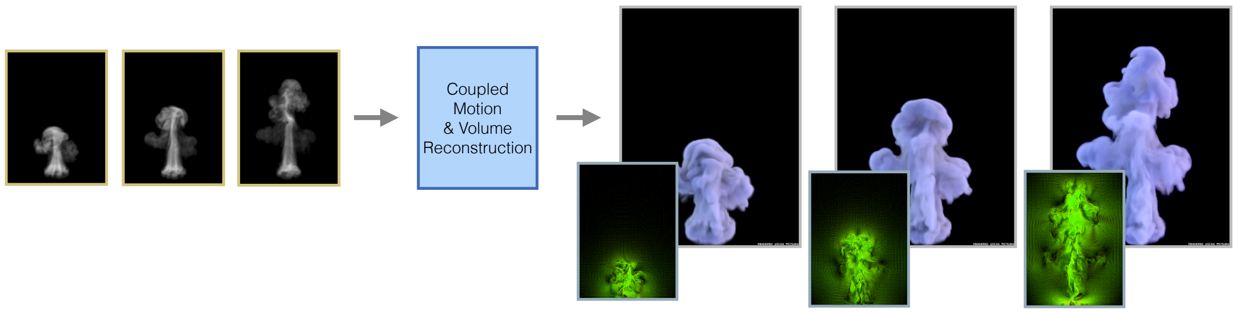


Figure 6.1.: Computing volumetric density and velocity from 2D images by employing an optimization formulation for the joint reconstruction of density and motion.

In this chapter, we reconstruct a fluid's density and motion based on just one single input image sequence of a real fluid phenomenon. Single-view reconstructions are especially under-constrained, particularly in the depth direction. We first explain details of the single-view reconstruction technique, which is slightly adapted compared to Section 4.2, then evaluate the results of synthetic reconstructions, and show some real-world single-view reconstructions. As in the previous chapter, we do not use any ground truth information available from synthetic input simulations when evaluating our method in order to make them representative of real-world scenarios. The results for single-view reconstructions can be found in our accompanying video[1].

6.1. Method Details

Our single-view results were obtained with a predecessor of the reconstruction method described in Section 4.2. In particular, we use a CG solver for the tomography part instead of a CGLS solver, use a simpler density inflow estimation, and do not add any viscosity term. We only prescribe inflow velocities for real-world reconstructions, but not for synthetic ones due to the simpler approximation of smoke inflow. Our single-view reconstruction procedure a version of Algorithm 3, where we simplify the aforementioned parts. Furthermore, we employ a different velocity alignment step instead of

[1]`https://youtu.be/J2wkPNBJLaI`

line 9. In [Thu17, PBT19], the authors explain that the additive combination of Eulerian flow fields additionally requires an alignment of the individual fields $\mathbf{u}^t = \Pi_{\text{DIV}}(\text{advect}(\tilde{\mathbf{u}}^t, \Delta\mathbf{u}^t)) + \Delta\mathbf{u}^t$.

Inflow Estimation Instead of using our advanced inflow estimation solver, we reconstruct one of the first capture time steps t_0, which shows a small part of the rising smoke, to obtain a smoke source Φ^0 in the visible area of our domain, i.e., in $\Gamma \setminus \Gamma_I$. In each time step, we set the source area to the said reconstructed smoke source for modeling a continuous smoke inflow. In order to receive a plausible smoke source, we limit the source in depth and in other spatial directions, if necessary. As such, we restrict the highly under-determined tomography solver to a desired target shape. For our synthetic reconstructions, we use a cylinder as target shape and assume the input plume to be initially at rest while we use a box for real-world reconstructions and assume the smoke enters the visible domain with an initial velocity, which we choose manually and keep constant for each time step.

Perspective versus Orthographic Cameras As outlined in Section 3.4, real-world cameras are perspective pinhole cameras. However, it can be useful for synthetic reconstructions to assume orthographic cameras for simplicity. For large distances between camera and objects to record, the assumption of having parallel rays is a good approximation. Since our cameras are relatively close to the smoke plumes, the rays we obtain from camera calibration are non-parallel, and hence, we assume perspective cameras throughout this thesis. Nevertheless, we compare reconstructions using perspective and orthographic cameras in this paragraph to illustrate difficulties of single-view reconstructions.

Yet, the two different camera models of perspective and orthographic cameras lead to very different reconstruction results when using only a single input view, which we demonstrate in Figure 6.2. The input to both reconstructions is a synthetic smoke plume as visualized in Figure 6.2a). A perspective camera is used to reconstruct Figure 6.2b), while we use an orthogonal camera for Figure 6.2c).

Both reconstructed densities expand into depth, which is visible in the side view, where orthogonal cameras lead to farther but symmetric expansion and perspective reconstructions evoke density expansions similar to the mirrored alphabetical letter 'C'. This curved motion is caused by residual velocities being reconstructed orthogonal to the viewing rays. While the residual density mostly acts along each line of sight, OF reconstructs velocities along density gradients, i.e., orthogonal to the viewing direction. Since the camera is placed at central height of the domain, the rays going through the top part of the domain are directed upwards, while the rays through the bottom domain are directed downwards in the perspective case. This causes the smoke plume to tilt towards or away from the viewing point in the top and bottom domain, respectively. This distortion is created due to the inherent under-determination of single-view reconstructions. One additional view prohibits such distortion successfully, as illustrated in Figure 6.5 and Figure 6.6. However, we develop a novel depth regularization term to improve single-view reconstructions by themselves.

Depth Regularization To increase control over depth motion, we additionally constrain velocities in the depth as their effect is not directly visible in the 2D input images as is the case for other velocity

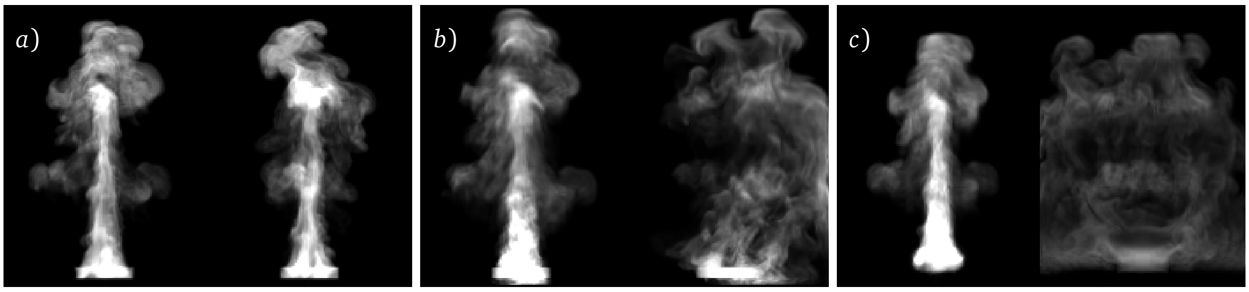


Figure 6.2.: a) Synthetic input, b) perspective, and c) orthographic reconstructions without depth regularizer at $t = 104$. Each image shows a front (left) and side view (right). The side views of b) and c) highlight the undesirable depth expansion and the 'C'-shaped distortion for the perspective camera view in b).

directions. With our axis-aligned voxelized volume, the depth motions are exactly the z-components of the velocity. First, we penalize large depth velocities more strongly by increasing the weight $\beta_{\Delta\mathbf{u}}$ for our Tikhonov regularizer described in Section 4.2.2 in Equation (4.1), i.e., we introduce $\beta_{\Delta\mathbf{u}_z}$. We set this weight depending on the voxel's position, i.e., we choose larger penalties for the top and bottom domain compared to the central domain, since in these cases, the rays are not directed horizontally but upwards or downwards:

$$\beta_{\Delta\mathbf{u}_z} = 1 + 10 * (|(Y/2.) - 1. - j|/(Y/2.))^2, \tag{6.1}$$

where Y and j are the domain size and the current voxel's index in the y-direction.

Furthermore, in order to avoid a 'C'-shaped drift of the densities, we introduce a novel regularizer minimizing the sum of depth velocities in one voxel row. The sum of z-velocity components along one row is described by the matrix S. Here, we only consider velocity components inside the smoke volume. This regularizer is added to the the systems matrix A_f of Equation (4.6) with a separate weighting term λ_{sum}:

$$E_{\text{sum}}(\Delta\mathbf{u}) = \tfrac{1}{2}\,||S\Delta\mathbf{u}||^2\,. \tag{6.2}$$

6.2. Evaluation Based on Synthetic Input

In order to evaluate our reconstruction method, we reconstruct various different synthetic simulation, which we first project onto the 2D image space for each time step t. The image resolution is 480×640 while our volumetric grid is of size $120 \times 160 \times 120$ if not stated otherwise. We use the following regularizing weights for synthetic reconstructions: $(\alpha_{\mathbf{u}}, \beta_{\mathbf{u}}, \alpha_{\Phi}, \beta_{\Phi}, \beta_{\Delta\mathbf{u}_z}, \lambda_{\text{sum}}) = (0.1, 1 \times 10^{-4}, 0.001, 1 \times 10^{-4}, 1 \times 10^{-3}, 10)$, which we found to obtain the fastest and most accurate results on average. Real-world reconstructions require slightly higher smoothness weights.

Plume with Single View Our first example is a simple plume. The synthetic input simulation and its perspective reconstructions are demonstrated in Figure 6.3a) - d) for front and side views. The side view is the most difficult to reconstruct, as it is orthogonal to the single input direction. As observable in

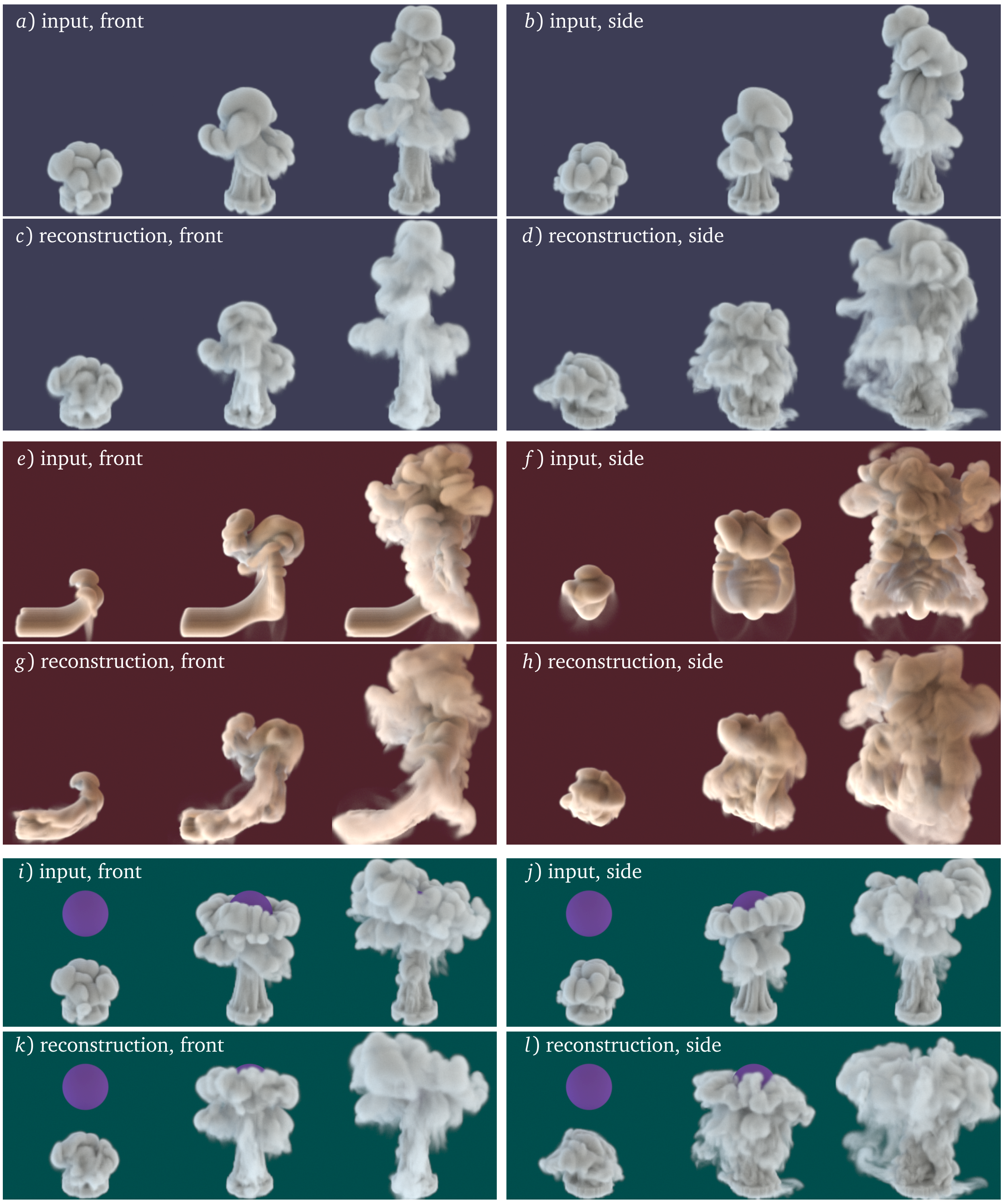

a) input, front
b) input, side
c) reconstruction, front
d) reconstruction, side
e) input, front
f) input, side
g) reconstruction, front
h) reconstruction, side
i) input, front
j) input, side
k) reconstruction, front
l) reconstruction, side

Figure 6.3.: Front and side views of input a,b) and reconstruction c,d) of a plume at $t = 30, 60, 96$, a jet stream e) - h) at $t = 30, 60, 96$, and a plume with solid obstacle i) - l) at $t = 30, 80, 112$.

Figure 6.3a) and c), the given input view is matched very well. The side view motion still expands in the depth direction, see Figure 6.3b) and d), as it is inherently under-constrained. However, considering

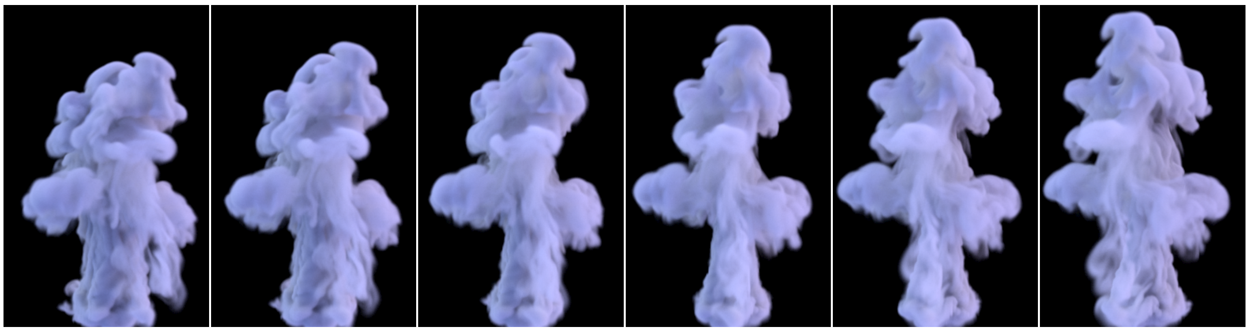


Figure 6.4.: Re-simulated plume reconstruction from a rotating viewpoint at $\gamma = 83, 85, 87, 89, 91, 93$ and $t = 84, 86, 88, 90, 92, 94$ with a $240 \times 320 \times 240$ domain.

that we are not explicitly constraining and hence limiting this motion, our reconstructions sill exhibit very natural flow behavior. Increasing the depth regularizing weights would in fact limit the depth motion further but at the expense of complying to our physical constraints and thus decrease the overall quality of the reconstructed motion.

Although the advected density distributions implicitly give insights into the underlying motion, we additionally present the velocity's central slice as seen from the front and side views for both ground truth and reconstructed motion in Figure 6.7 a) - d). The velocities in both central slices display natural and plausible motions, although the side view reveals a larger depth motion compared to the ground truth. In Figure 6.4, we demonstrate the flexible usage of our reconstructed 3D velocity fields applied to re-simulating the simple plume in Figure 6.3c) - d) on a higher resolution, where we advect a two times finer density distribution with our reconstructed velocity. Figure 6.4 shows the re-simulated fluid with both varying time and viewing angle, where $\gamma denotes the viewing angle$. These renderings point out the amount of detail captured by our velocity fields, which could be refined further with procedural turbulence methods.

Multiple View Constraints We are able to re-use the information of the input image to further constrain our plume from other views. The first approach is to use the mirrored input image from angle 0° as additional input view for 180°, which we call *front-as-back*. In doing so, we counteract the undesired tilting motion resulting from perspective camera rays without applying any depth reg-

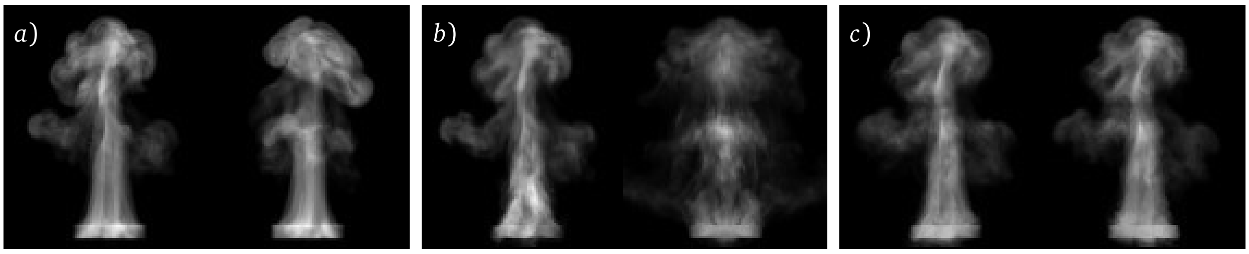


Figure 6.5.: a) Synthetic input, b,c) reconstructions with front view as constraint for back and side view, respectively, at $t = 76$ without using any depth regularizers.

ularizers, as demonstrated in Figure 6.5b). However, we obtain a symmetric side view. The second approach uses the front image as side constraint, which hence assumes partial rotational invariance of the plume. This assumption is overly limiting in a general setting although we use a decreased weight of 10 % for the side view compared to the front view, which is still matched with highest priority. As demonstrated in Figure 6.5c), the side view closely matched the front view when using the approach *front-as-side*. Hence, the depth expansion is eliminated even when not applying any depth regularizer. Both approaches increase the reconstruction quality drastically and demonstrate the large null-space of single-view reconstructions.

Jet Stream and Obstacle Scene In order to validate that our reconstruction method is fully capable of handling rotationally asymmetric flow and flows with interior boundaries, we generate two more synthetic cased of a jet stream and of a rising plume meeting a solid sphere. Figure 6.3e) - h) show a jet stream flowing in from the left side of the domain and rising upwards due to buoyancy while the plume with sphere in the center of the domain is shown in i) - l). The input view is matched very well for both scenarios. Our jet reconstruction plausibly recreates the depth motion as shown in the side view. For this scene, imposing artificial novel view constraints as mentioned before is clearly inadequate. The jet scene is a good example that specifically benefits from a single-view reconstruction.

Our plume reconstructions including the solid sphere likewise match the input view closely and exhibits a realistic motion for the unconstrained behaviors throughout the volumetric density. Our solver has prior knowledge of the size and position of the sphere. However, we could use techniques for single-view geometry estimation for a more generic reconstruction, which we suggest as extension for future work.

Comparison to Previous Work We now compare our reconstruction technique with a tomography plus divergence-free OF by Gregson et al. [GITH14] considering the simple rising plume from Figure 6.3a) - b). Although our method works well with a single input, we make use of an additional, orthogonal view for better comparison to previous work. Our reconstruction match the ground truth densities very well and much better than previous work, see Figure 6.6 and ground truth synthetic input in Figure 6.3a) and b). Our method produced less volume expansion and is closer to the ground truth views. The reconstructed motion for both approaches is illustrated in Figure 6.7 e) - h). Our velocities match the input in a) and b) accurately while Gregson's approach exhibits undesired velocities outside the plume's shape since it works best for reliable density estimations, which in turn correct the estimated motion at each time step. These results demonstrate again that decoupled velocity and density estimation leaves too many degrees of freedom undefined when using only a sparse number of input views degrading the reconstruction's quality.

Okabe et al. [ODAO15] present single-view reconstructions as well. However, their work primarily concerns with volumetric appearance where the captured motions are naturally less reliable. Hence, we do not compare our algorithms, but it would be an interesting future extension to combine both approaches.

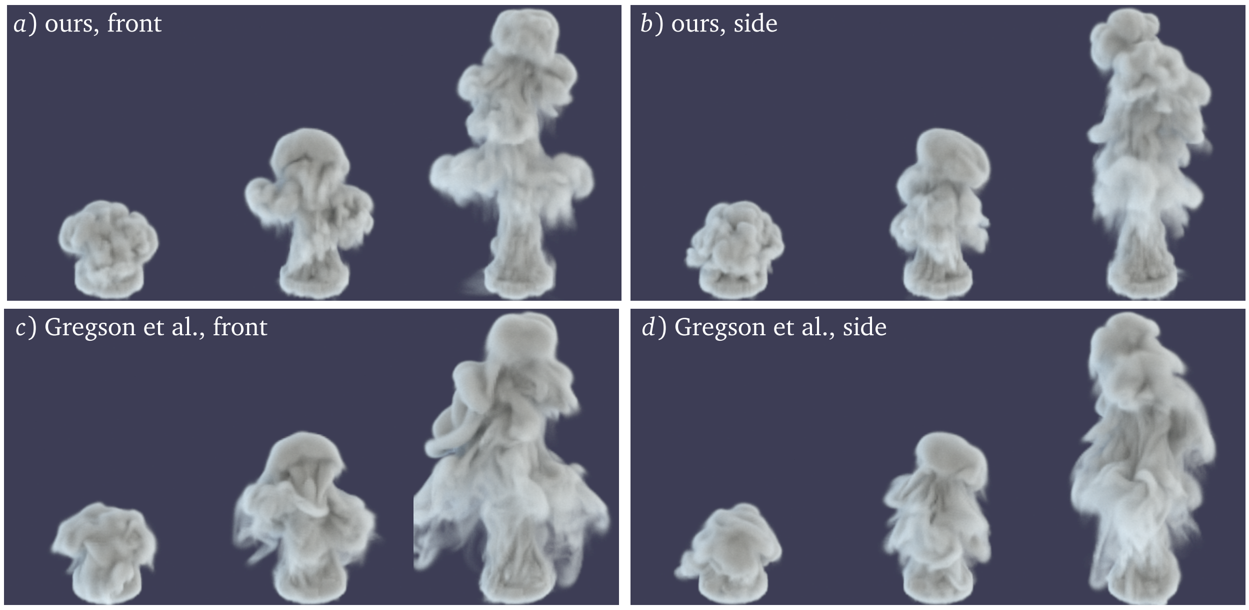


Figure 6.6.: Our reconstruction a,b) and the approach by Gregson et al. [GITH14] c,d) for a synthetic rising plume scene.

Time Extrapolation by Simulation In order to investigate the robustness of our method, we reconstruct the first half of a simulation until $t = 48$ and continue with a standard forward simulation in the second half. We show an example of the rising plume in Figure 6.8 with ground truth input, reconstructed density, and time extrapolated density. The extrapolated version follows the reconstruction tightly within the first ∼25 frames and is hence close to the input simulation, which validates the reliability of our reconstructed density and velocity. However, it would be very interesting to automatically adapt simulation parameters, such as buoyancy, to match the ground truth simulation for a larger period of time.

6.3. Real-World Results

With our capturing setup described in Section 4.1, we record rising hot smoke. Two examples of such raw captures without our post-processing technique are demonstrated in Figure 6.9a) and d). The invisible and temporally changing smoke inflow from the real-world smoke plumes poses a challenge for our single-view reconstructions, especially as we are using a simplified inflow estimation as outlined above. We estimate the inflow velocity c roughly from the captured images and use the same for both our real-world plumes. Our resulting reconstructions exhibit realistic volumetric motion, which leads to a good match between rendered density and input images and realistic swirls in the side view, which we demonstrate in Figure 6.9. The rendered density is created with a similar visual style but slightly different camera properties are shown below each raw input row. These results demonstrate that with our powerful reconstruction technique, we are able to reconstruct 3D density and motion from a single-view, which match the input image sequence well while exhibiting plausible fluid behavior throughout

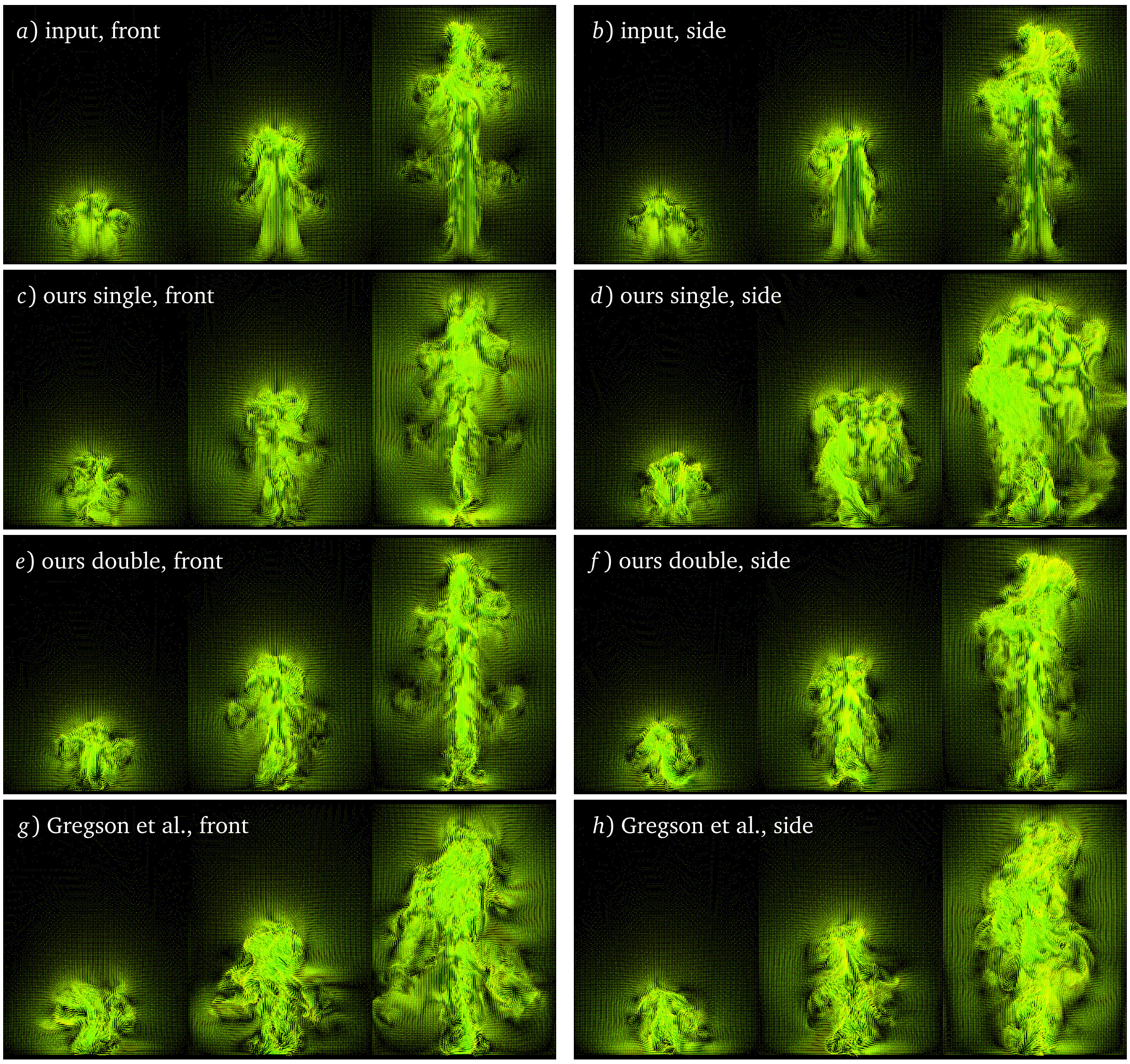


Figure 6.7.: Central velocity slices of ground truth, our single-view, our double-view, and Gregson et al.'s reconstructions at $t = 30, 60, 96$.

the whole volume.

Performance An overview over reconstruction run times and grid sizes is presented in Table 6.1. In fact, later time steps with larger visual hulls require much more computational resources compared to earlier time steps. As such, the run time is especially increased for the jet, plume, and first real capture reconstructions, where the density volume becomes largest. As mentioned earlier and in Section 4.2.3, our CG implementation calculates and stores the image formation matrix $P^T P$ explicitly which facilitates implementation but is suboptimal in terms of run time or memory consumption.

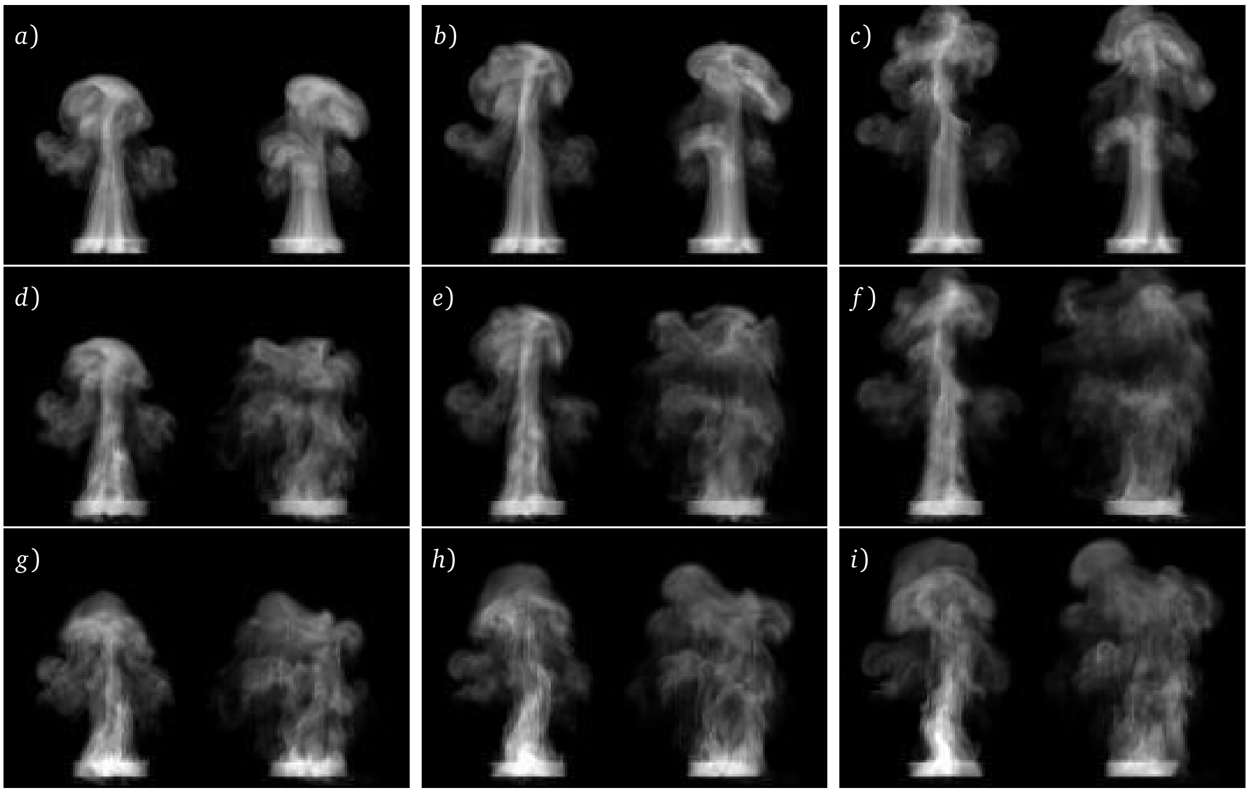


Figure 6.8.: Ground truth input in a,b,c), reconstruction in d,e,f), and extrapolation by simulation in g,h,i) after 12, 22, and 32 frames.

Scene	Grid Size	#Frames	Avg. T
Plume, Figure 6.3	120 × 160 × 120	96	61m
Re-sim., Figure 6.4	240 × 320 × 240	96	08m
Front = Side, Figure 6.5	120 × 160 × 120	76	52m
Jet, Figure 6.3	120 × 160 × 120	96	68m
Obstacle, Figure 6.3	120 × 160 × 120	112	45m
2 views, Figure 6.6	120 × 160 × 120	107	44m
Real, Figure 6.9 a)	120 × 200 × 120	120	62m
Real, Figure 6.9 d)	120 × 200 × 120	120	54m

Table 6.1.: Performance details for all reconstructed scenes. The time is given as an average over the whole sequence.

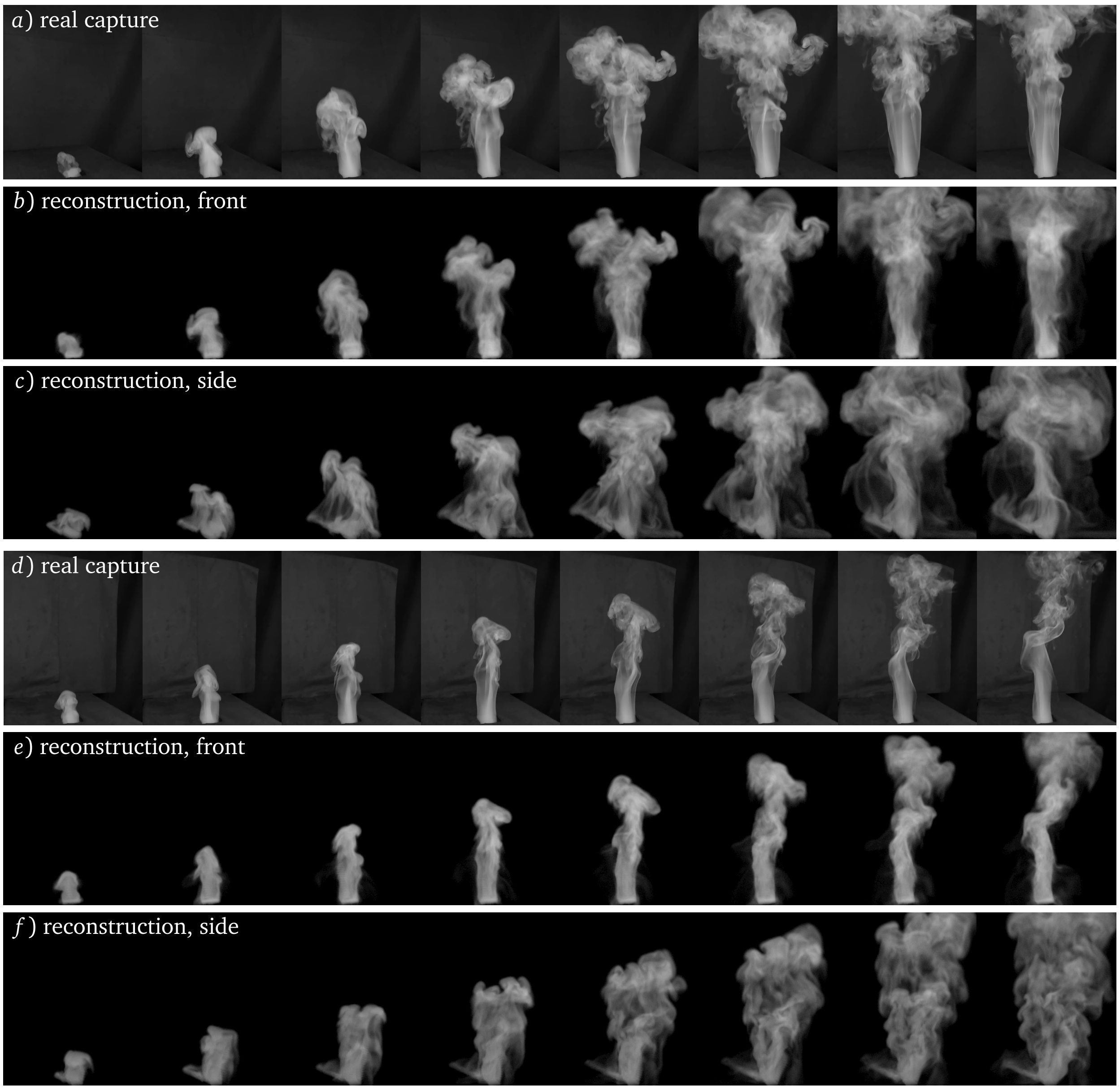


Figure 6.9.: Two captures of real fluids a,d) and reconstructions with front d,e) and side views c,f). Frames are shown in frame intervals of 15.

7 Fluid Guiding

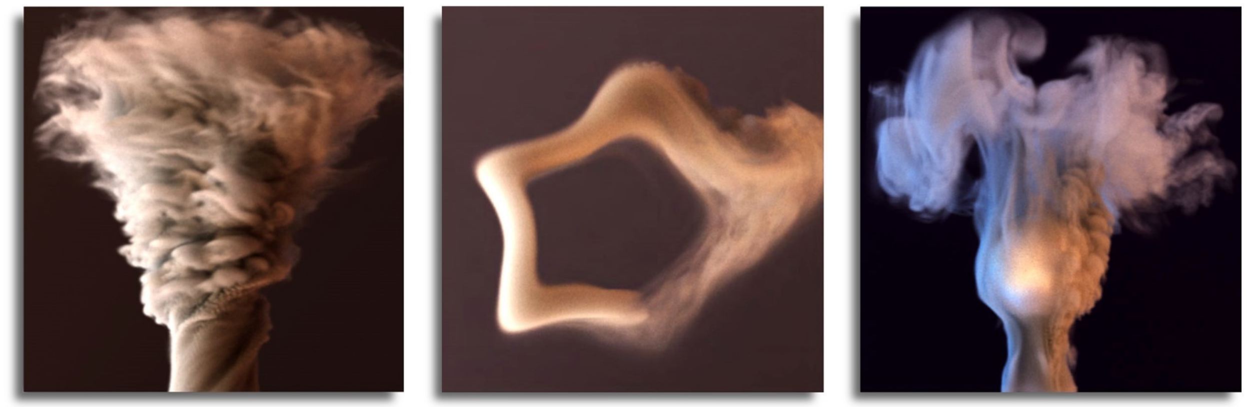


Figure 7.1.: Guiding smoke to form a tornado, to create a star shape, and to follow a low-resolution simulation with spatially varying guiding and a static obstacle.

We first describe our flexible fluid guiding method with several acceleration schemes, evaluate our results and performance on 2D examples, and show 3D results. We found PD to converge faster than ADMM. IOP is not applicable for fluid guiding, as not both proximal operators are orthogonal projections. The results for both fluid guiding and separating solid-wall BCs can be found in our accompanying video[1].

7.1. Method

Our objective is to guide the coarse motion of a fluid simulation to follow a target velocity while permitting the creation of small-scale fluid details. Therefore, we minimize the blurred difference between the guided velocity field $\mathbf{x}$ and the target velocity field $\mathbf{u}_{\text{tar}}$, where the blurred difference is exactly the coarse motion. We additionally constrain the change of the current velocity field $\mathbf{u}_{\text{cur}}$ to be small. The current velocity field is the simulation's velocity after advection and adding forces, but before the pressure projection, see the typical simulator steps in Figure 3.1. Our fluid guiding problem is hence

[1]`https://youtu.be/Pgbat5MXo8Q`

formulated as following:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} \quad f(\mathbf{x}) = \|G(\mathbf{x}-\mathbf{u}_{\text{tar}})\|^2 + \|W(\mathbf{x}-\mathbf{u}_{\text{cur}})\|^2 \\ &\text{subject to} \quad \mathbf{x} \in C_{\text{DIV}}, \end{aligned} \tag{7.1}$$

where W is the guiding weights matrix, G is the Gaussian blur matrix, and C_{DIV} is the space of divergence-free velocity fields. Both matrices are spatially varying, which allows guiding to be applied differently to distinct positions in space. Nielsen and Christensen [NC10] also realized spatially varying fluid guiding, but not with spatially varying blur. $W = W(\mathbf{x})$ is a diagonal matrix that contains the weights for controlling the guiding strength where larger entries imply weaker guiding. The Gaussian blur matrix $G = G(r_\beta, \mathbf{x})$ applies a blur of radius r_β to fluid cells. Boundaries and obstacles are left untouched. The first part of our objective function f is minimal for a velocity field that matches the target velocity when blurred with G. The second part penalizes velocities far away from the current velocity field.

As our objective function f is quadratic, we follow the derivation in Section 3.2 and formulate the proximal operator as

$$\mathbf{prox}_{f,\sigma}(\xi) = (A + \sigma I)^{-1}(\sigma\xi - b),$$

where

$$\begin{aligned} A &= 2(G^T G + W^2) \\ b &= -2(G^T G\mathbf{u}_{\text{tar}} + W^2\mathbf{u}_{\text{cur}}) \\ r &= \mathbf{u}_{\text{tar}}^T G^T G\mathbf{u}_{\text{tar}} + \mathbf{u}_{\text{cur}}^T W^2\mathbf{u}_{\text{cur}}. \end{aligned}$$

Since W is symmetric, the following holds: $W^T W = W^2$. When we factor out $\mathbf{u}_{\text{cur}}$, our proximal operator for f simplifies to

$$\mathbf{prox}_{f,\sigma}(\xi) = \mathbf{u}_{\text{cur}} + M^{-1}(\sigma\xi + q), \tag{7.2}$$

where

$$M = 2G^T G + 2W^2 + \sigma I \quad \text{and} \quad q = 2G^T G(\mathbf{u}_{\text{tar}} - \mathbf{u}_{\text{cur}}) - \sigma\mathbf{u}_{\text{cur}}.$$

The incompressibility constraint contained in g is solved as described in Section 3.2 with the proximal operator $\mathbf{prox}_{g,\rho}(\xi)$ given in Equation (3.16).

Instead of applying PD, we could solve both f and g in one large system as both are quadratic energies. More precisely, we can combine $f'(\mathbf{x}) = \mathbf{0}$ with three equations per cell in 3D and the incompressibility constraint with one equation per cell in one linear system $L\mathbf{x} = \mathbf{d}$. Since this system is over-constrained, we solve it with least squares by considering

$$L^T L\mathbf{x} = L^T \mathbf{d}, \tag{7.3}$$

where $L = \left(\begin{smallmatrix} A \\ \nabla\cdot \end{smallmatrix}\right)$ and $\mathbf{d} = \left(\begin{smallmatrix} \mathbf{b} \\ 0 \end{smallmatrix}\right)$. However, solving the large system, e.g., with an iterative solver, such as the CG, is infeasible since the number of elements in the matrix grows with $\mathcal{O}(N)$. N denotes the total number of cells in the simulation domain. Hence, we apply PD for our guiding scheme but compare the performance of a direct CG solver in Section 7.2.1.

Inverse Matrix Approximation for Fluid Guiding Computing $M^{-1}(\sigma\xi + q)$ for every iteration is very slow. The inversion of M is expensive, as M is large with the dimensions $N \times N$. Since G, W, and σ are constant for each simulation, matrix inversion takes place only once. In order to speed up our guiding scheme, we approximate the inverse of M. q can also be precomputed for each time step, as it does not rely on previous iterations.

The diagonal entries of M are orders of magnitude larger than its off-diagonal entries. We can express our matrix as $M = D + (2G^T G)$, where $D = 2W^2 + \sigma I$ contains the large diagonal terms and $2G^T G$ includes small off-diagonal terms, which are smaller than 1. By using the Sherman-Morrison-Woodbury formula, M^{-1} can be approximated as

$$M^{-1} = D^{-1} - 2D^{-1}G^T(I + 2GD^{-1}G^T)GD^{-1}. \tag{7.4}$$

Now, since G and D^{-1} both contains small value entries, $2GD^{-1}G^T$ is approximately zero. Hence

$$\begin{aligned} M^{-1} &\approx D^{-1} - 2D^{-1}G^T GD^{-1} && (7.5)\\ &= (2W^2 + \sigma I)^{-1} - 2(2W^2 + \sigma I)^{-1}G^T G(2W^2 + \sigma I)^{-1}. && (7.6) \end{aligned}$$

The inverse of D is trivial, as $D = 2W^2 + \sigma I$ is diagonal.

Separable Gaussian Another simplification can be applied to $\mathbf{prox}_{f,\sigma}(\xi)$, as spatially invariant Gaussian blurs are symmetric, which implies that G is a symmetric matrix and therefore $G^T G = G^2$. Furthermore, they are separable as convolution is associative. Foe example, a 2D convolution can be expressed as two 1D convolutions and as such, separable Gaussians can be applied independently in each dimension. A Gaussian blur applied to a 2D image or a 3D volume is equivalent to applying two or three independent 1D Gaussian blurs, which speeds up computation substantially.

Considering spatially varying Gaussian blurs, we still use symmetric and hence separable Gaussian blurs for each cell. Yet, as blur radii differ among cells, G is no longer symmetric. If the spatial variation is limited, we use $G^T G$ to approximate G^2, as $G^T G \approx G^2$ for regions of constant blur. An example for such limited spatial variation are different blur radii for the left and right parts of the simulation domain. Our approximation works well in practice as shown in Section 7.3. However, our PD guiding scheme is independent of the particular instance and realization of the blur kernel. It would be straightforward to replace our current approximation with a fast and full implementation of calculating $G^T G$.

For solid obstacle cells, we use a blur radius of zero and therefore allow for interior boundaries in contrast to the guiding scheme based on the fast Fourier transform [GITH14]. Fast Fourier transform schemes usually require periodic domains and are not able to handle internal boundaries.

Summary Applying the separable Gaussians and approximating the inverse of M greatly speeds up the $\mathbf{x}$-update for fluid guiding. The new PD variable updates are given by

$$\begin{aligned}
\mathbf{x}^{k+1} &= \mathbf{x}^k + \sigma\mathbf{y}^k - \sigma\left(\mathbf{u}_{\text{cur}} + D(\mathbf{x}^k + \sigma\mathbf{y}^k + q) - 2G^T G D^2(\mathbf{x}^k + \sigma\mathbf{y}^k + q)\right)\\
\mathbf{z}^{k+1} &= \Pi_{\text{DIV}}(\mathbf{z}^k - \tau\mathbf{x}^{k+1}) \\
\mathbf{y}^{k+1} &= \mathbf{z}^{k+1} + \theta(\mathbf{z}^{k+1} - \mathbf{z}^k).
\end{aligned} \tag{7.7}$$

Algorithm 9 outlines the proximal operator for f with the given guiding specific parameters W and r_β.

Algorithm 9 Approximation for $\mathbf{prox}_{f,\sigma}(\xi)$ in fluid guiding

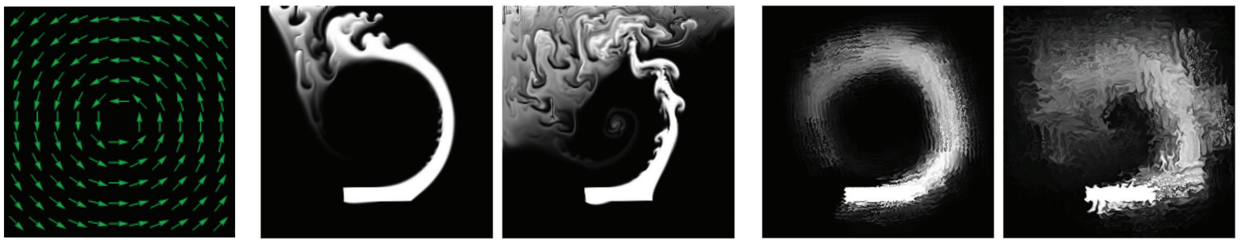
```
1: procedure PROXF($\sigma$, $\xi$, $W$, $r_\beta$)
2:     $q = 2G^T G(\mathbf{u}_{\text{tar}} - \mathbf{u}_{\text{cur}}) - \sigma\mathbf{u}_{\text{cur}}$        ▷ can be precomputed
3:     $D = (2W^2 + \sigma I)^{-1}$        ▷ inverse of diagonal matrix
4:     return $\mathbf{u}_{\text{cur}} + D(\sigma\xi + q) - 2G^T G D^2(\sigma\xi + q)$
5: end procedure
```

7.2. Evaluation

We first evaluate our guiding methods based on 2D examples and then show more sophisticated 3D results. To simplify notation, we write $W = c$ for spatially invariant guiding strengths c and $W = (c_1, c_2)$ for spatially varying strengths c_1 and c_2, e.g., on the left and right side of the domain. Spatially varying blur radii are written analogously as $r_\beta = (r_{\beta 1}, r_{\beta 2})$.

A comparison of our method and two naïve guiding methods is shown in Figure 7.2a, where we guide simulations with a spiral, counterclockwise circular velocity field. After guiding, the velocity is projected to be divergence-free for both alternative methods. A simple approach to fluid guiding is linear velocity blend, where the new velocity is a linear combination of the current and the target velocity: $\mathbf{u}_{\text{new}} = r_{\text{blend}}\mathbf{u}_{\text{cur}} + (1 - r_{\text{blend}})\mathbf{u}_{\text{tar}}$, where $0 \leq r_{\text{blend}} \leq 1$. Smaller values for r_{blend} lead to stronger guiding. The drawback of linear velocity blend is that strong guiding smoothes out small-scale fluid characteristics. In order to obtain more details, guiding must be weaker, which leads to reduced trajectory control, see Figure 7.2b.

Another alternative is detail-preserving guiding [TKPR06] where the large-scale, i.e., blurred, velocity parts are subtracted from the current velocity, resulting in preserving small-scale details from the



(a) Target velocity **(b)** Linear blend with $r_{\text{blend}} = 0.85, 0.95$ **(c)** Detail-preserving blend with $r_\beta = 3$, $r_\beta = 5$

Figure 7.2.: Naïve guiding methods with a spiral velocity field as target.

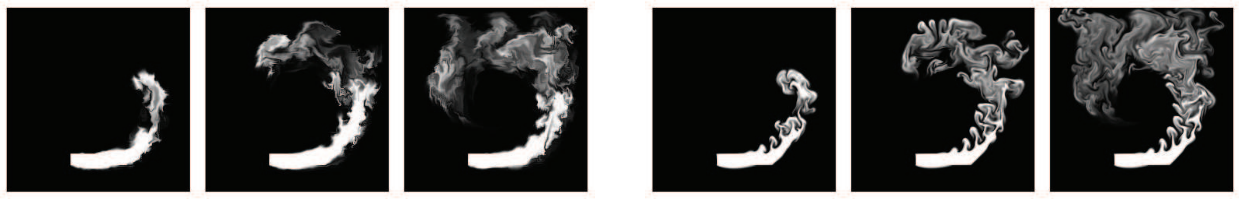


Figure 7.3.: Up-res of 64^2 simulations to a resolution of 256^2 using wavelet turbulence (left) and our method (right). Our resulting fluid motion contains significantly more coherent vortices.

simulated, current velocity field $\mathbf{u}_{\text{cur}}$. Then, the target velocity $\mathbf{u}_{\text{tar}}$ is added, accounting for the desired large-scale motion: $\mathbf{u}_{\text{new}} = \mathbf{u}_{\text{cur}} - G\mathbf{u}_{\text{cur}} + \mathbf{u}_{\text{tar}}$. As such, large-scale motions are guided by a target velocity field while small-scale details are preserved. However, it is challenging to control these details, which furthermore exhibit an unnatural frosted glass appearance, see Figure 7.2c.

In Figure 7.3, we compare our algorithm to wavelet turbulence [KTJG08], which is a fast and purely post-processing technique adding small-scale details without influencing the coarse-scale motion. The resulting vortices do not couple as realistically and tightly as with our technique.

Our method allows for much more flexible control over the fluid's motion compared to the presented alternative methods. W determines the large-scale guiding strength and the blur radius r_β affects the creation of small-scale details. An example of the influence of guiding strength and blur radius is given in Figure 7.4, based on the circular target field of Figure 7.2a. Larger guiding strength values W allow for more deviation from the prescribed velocity field, while larger blur radii r_β encourage the formation of small-scale details.

7.2.1. Performance

We compare the convergence of PD to the proximal methods ADMM and IOP, see Sections 3.2.1 and 3.2.3. While ADMM is closely related to PD, see how PD reduces to ADMM Section 3.2.2, IOP is a variant of the Projection onto Convex Sets (PoCS) algorithm [BB96], which is suited for convex feasibility problems and not for the more general problem solved by ADMM and PD. Convex feasibility problems locate intersection points of convex sets. When employing IOP for our fluid guiding problem, we alternate between the minimum of f and its closest divergence-free neighbor, which is not the divergence-free minimizer of f, as visualized in Figure 7.5. A divergent target velocity field leads to

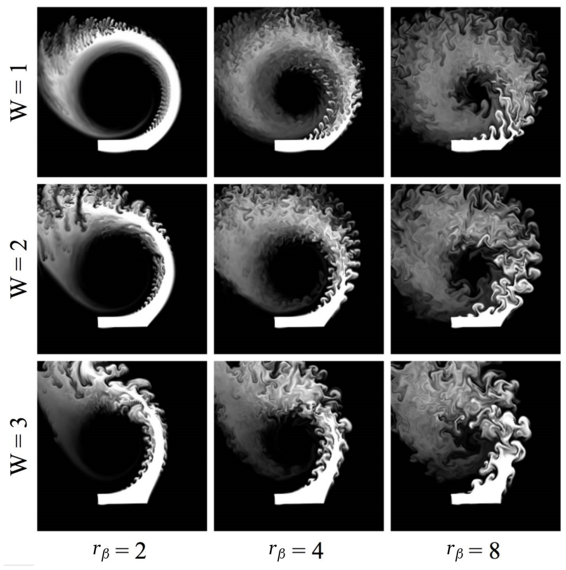


Figure 7.4.: Our guiding technique with a circular target velocity, varying W and r_β to allow for multi-scale motion control.

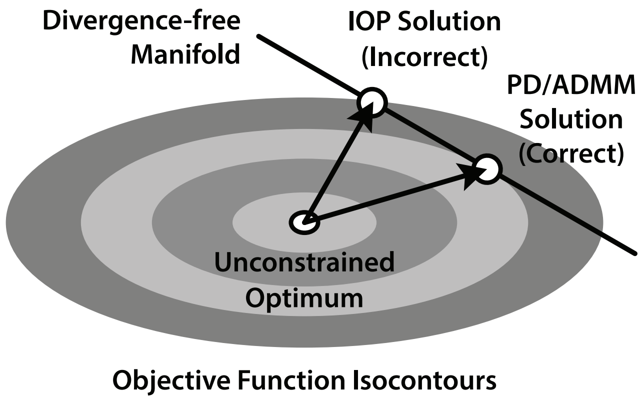


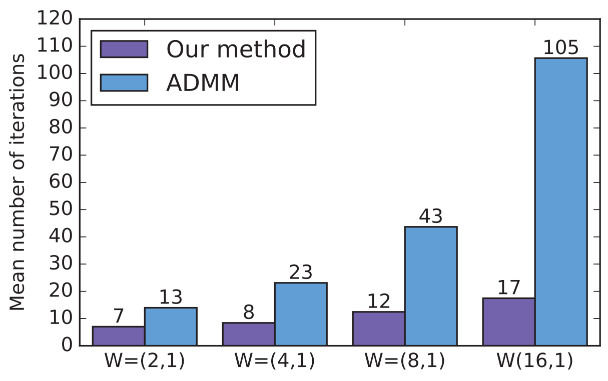
Figure 7.5.: Failure case for IOP given a divergent target velocity. The IOP's solution is on an isocontour farther from the center compared to the solution of PD/ADMM, which is the true minimum.

IOP failure, while ADMM and PD successfully converge. While IOP finds plausible approximate solutions for some instances, its reliability is limited in the general guiding case. Hence, we focus here on comparing PD with ADMM where we refer to PD with *our method*.

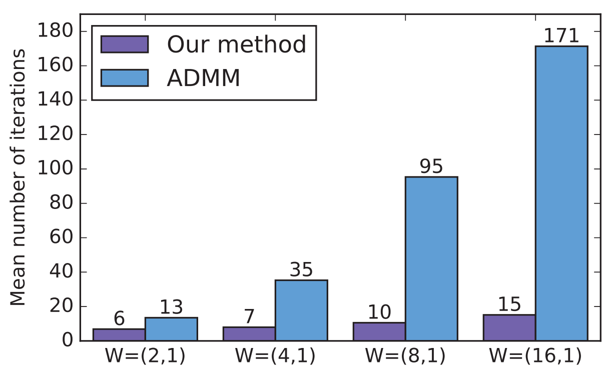
To ensure fair comparisons, we apply our acceleration schemes of matrix inversion and separable Gaussians to both methods. Furthermore, we deduce optimal parameter sets for both ADMM and our method through experiments with varying test scenes. On average, these parameters lead to the fastest convergence rates influenced by guiding weights and blur radii. We found that the optimal convergence parameters of both ADMM and PD are correlated to the average guiding weight $\bar{W}$, which averages the diagonal terms of W. For PD, we set $(\tau, \sigma, \theta) = (0.58/\bar{W}, 2.44/\tau, 0.3)$, and we use $\rho = 1.4\bar{W}^2$ for ADMM.

First, we compare the number of iterations for both ADMM and PD for a 2D fluid guiding on a 256^2 grid with our circular target velocity in Figure 7.2a. We use spatially varying weights, where $c_1 = \{2, 4, 8, 16\}$ for the left part and $c_2 = 1$ for the right part of the domain. We set $r_\beta = 1$. The mean number of iterations required to each convergence for both methods is shown in Figure 7.6a. Both methods perform decently for low spatial variance. However, PD clearly outperforms ADMM for larger spatial variance, e.g., for $W = (16, 1)$, PD needs only 16 % of the number of iterations of ADMM.

The same guiding parameters are used for our 3D scene, where we guide a smoke plume in a domain with a solid obstacle, as the one in Figure 7.14. The simulation's resolution is 120^3. As shown in Figure 7.6b, our method repeatedly outperforms ADMM, where the difference in convergence rates is even more apparent. For the largest spatial variance, PD needs only a small percentage of the iterations ADMM requires for convergence, namely less than 9 %. The run times for both methods are closely related to the mean number of iterations, as demonstrated in Table 7.7.



(a) 2D circular target

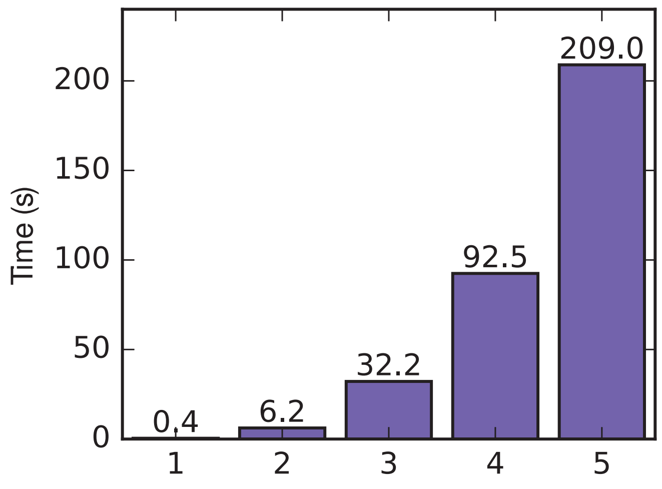


(b) 3D obstacle plume

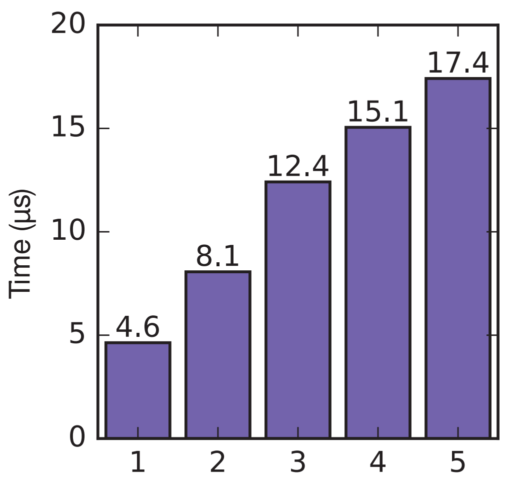
Figure 7.6.: Mean number of iterations per time step for guiding.

		Guiding weight *W*			
		(2,1)	**(4,1)**	**(8,1)**	**(16,1)**
2D	**Our method**	0.14	0.17	0.26	0.38
	ADMM	0.26	0.41	0.76	1.89
3D	**Our method**	16.6	19.5	26.3	38.4
	ADMM	30.4	67.8	176.9	325.6

Table 7.7.: Mean run time per time step (in seconds).



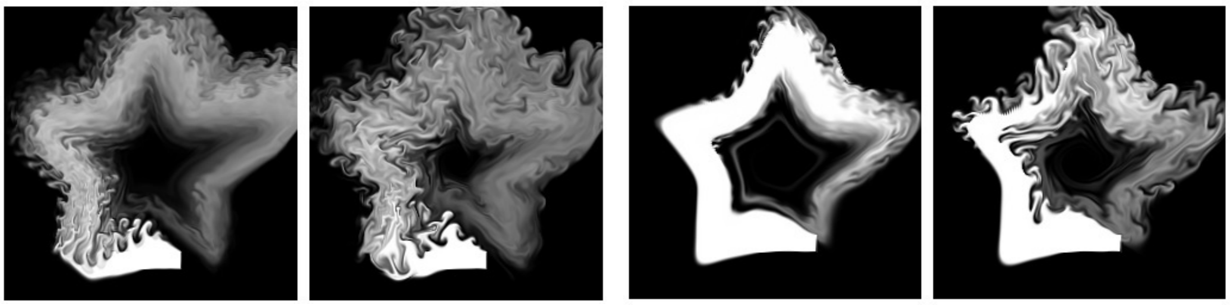
(a) Mean run time.



(b) Mean run time per grid cell.

Figure 7.8.: Scaling factor of our method's run time per time step for the 3D tornado scene at various resolutions.

Furthermore, we analyze how our guiding method scales with grid resolution based on the guided 3D tornado scene in Figure 7.1. We use five different grid resolutions, where the base resolution is $40^2 \times 60$. Hence, a scaling factor of three corresponds to the resolution $120^2 \times 180$. The mean run time



(a) without up-res on 64^2 **(b)** with up-res on 256^2

Figure 7.9.: Simulations guided by a star-shaped low-resolution simulation on 64^2 with $r_\beta = 5$ and $r_\beta = 9$.

per time steps are shown in Figure 7.8a, which appear to scale exponentially with increasing scaling factor. However, the mean run time per grid cell shows that in fact, the scaling is linear, see Figure 7.8b. Our guided simulation at the largest resolution $200^2 \times 300$ takes less than four minutes per time step.

As mentioned before, we could possibly solve the system of combined guiding and divergence-free constraints in Equation (7.1) directly with a CG solver. Solving the large system features poor performance leading to infeasible run times. For a small 128^2 example of our circular target velocity, the direct CG solver is 4000 slower than our PD-based method.

While the performance tests above all use our generic method of spatially varying guiding weights and blur radii, the special case of spatially invariant weights can be handled with a non-iterative method. In that case, we only use the non-divergent components of the current and target velocity fields $\mathbf{u}_{\text{cur}}t$ and $\mathbf{u}_{\text{tar}}t$ in $f(\mathbf{x})$ and omit the additional divergence-free constraint. The result of minimizing f with spatially invariant weights is divergence-free, as differentiation commutes with convolution. This non-iterative method greatly speeds up guiding, but we focus here on the more general guiding scheme with spatially varying operators that make iterating indispensable.

7.3. Results

We found an iterative up-res method to produce the most interesting guided motions, which is in line with previous work [KTJG08, YCZ11, HMK11, RLL*13, HK13]. In up-res methods, coarse flows are refined to yield the final, high-resolution result. As an example, consider Figure 7.9 of 2D star-shaped simulations. The base input is a star-guided simulation on a 64^2 grid with $W = 1$ and $r_\beta = 1$. If we use the base velocity field as input to guide a simulation with the same resolution but different blur radii, i.e., $r_\beta = 5$ and $r_\beta = 9$, the resulting flow has difficulties to follow the given shape while producing interesting small-scale details, see Figure 7.9a. However, when using the base velocity as input for a high-resolution simulation on a 256^2 grid, interesting fluid characteristics are created while preserving a sharp star shape, see Figure 7.9b.

We apply this up-res procedure for a simple 3D plume as shown in Figure 7.10. The base simulation is run on $50^2 \times 100$. Then, we use the upsampled base velocity field to guide two re-simulations on

$200^2 \times 400$ – one with spatially invariant blur $r_\beta = 5$ and one with spatially varying blur $r_\beta = (1, 10)$ in order to create different effects. In general, a larger blur radius permits the creation of more small-scale details. Our spatially varying example exhibits a sharp transition between the two blur radii in order to demonstrate the contrast between the two guided flow parts. The obvious seem can be damped by applying a gradual transition between both radii. This example validates that the approximation of $G^T G = G^2$ works well in practice.

We illustrate varying guiding strengths in Figure 7.11, where the simulations are guided by star-shaped input velocities. The base simulation's resolution is 50^3, while the high-resolution re-simulations take place on 250^3 grids. The blur radius is set to $r_\beta = 2$ for all examples. The guiding weight for the base resolution is spatially invariant $W = 1$, but varies on the right side for both re-simulations, namely $W = 5$ and $W = 7$. Compared to the overall computational costs, the costs for pre-computation are negligible, as one time step for a 250^3 simulation takes approximately 1.5 s.

We repeat the tornado simulation from Figure 7.1 with different blur radii leading to different levels of turbulence, as visualized in Figure 7.12. The base simulation is run on a $40^2 \times 60$ grid with a cylindrical target velocity with a small upward component. The re-simulations take place on a $200^2 \times 300$ grid.

As last example, we demonstrate our method's capability of handling interior boundaries in form of a solid sphere. Such interior boundaries cannot be handled by Fourier-domain based guiding methods, such as [GITH14]. The base simple plume simulation is run on 40^3 and is upsampled to 200^3. The guided high-resolution simulation with $W = 1$ and $r_\beta = 5$ follows the coarse input motion, but additionally creates visually appealing, fluid-specific details, as demonstrated in Figure 7.13. Spatially varying weights $W = (1, 100)$ are illustrated in Figure 7.14 with $r_\beta = 1$. The resulting motion on the left side of the plume features little additional small-scale details due to tightly following the base velocity. The right region is weakly guided and hence exhibits interesting smaller vortices. This example verifies that our separable, efficient, and easily parallelizable fluid guiding solver for Equation (7.1) easily handles arbitrary boundaries.

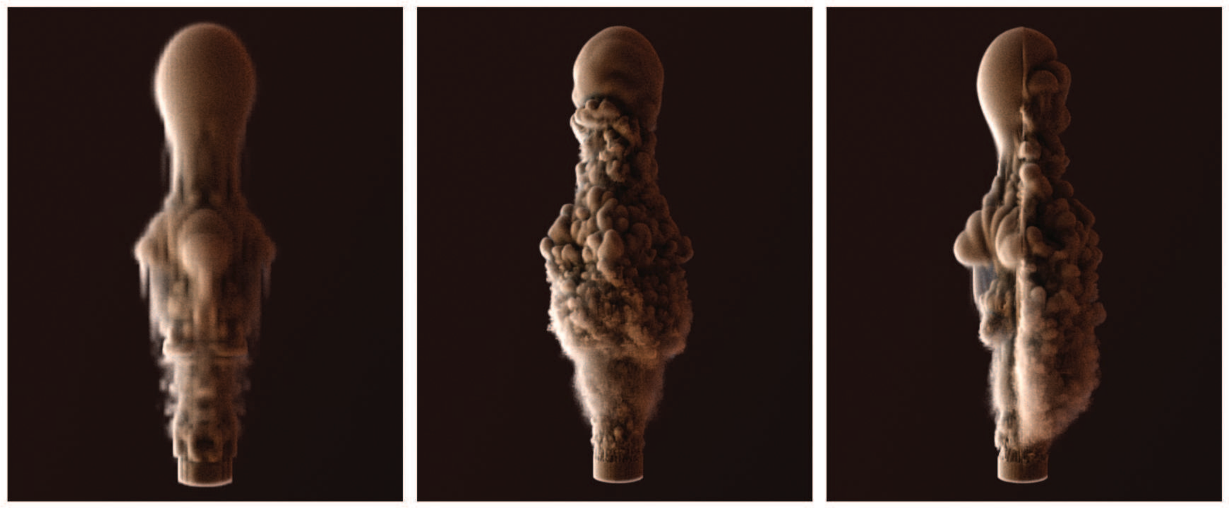


Figure 7.10.: Base simple plume simulation on $50^2 \times 100$ (left), guided re-simulations on $200^2 \times 400$ with constant blur (middle, $r_\beta = 5$) and spatially varying blur (right, $r_\beta = (1, 10)$).

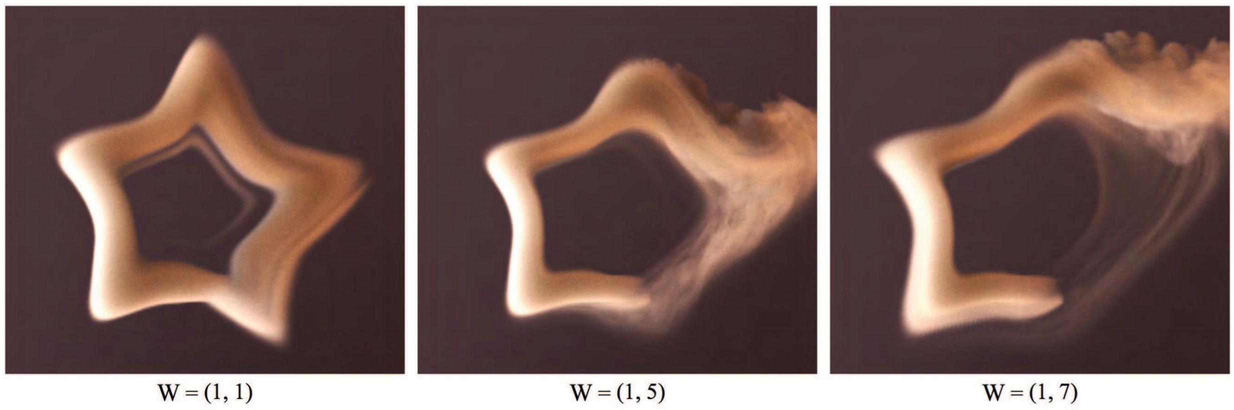


Figure 7.11.: Base star-shaped simulation on 50^3, guided re-simulations on 250^3 with spatially varying weights.

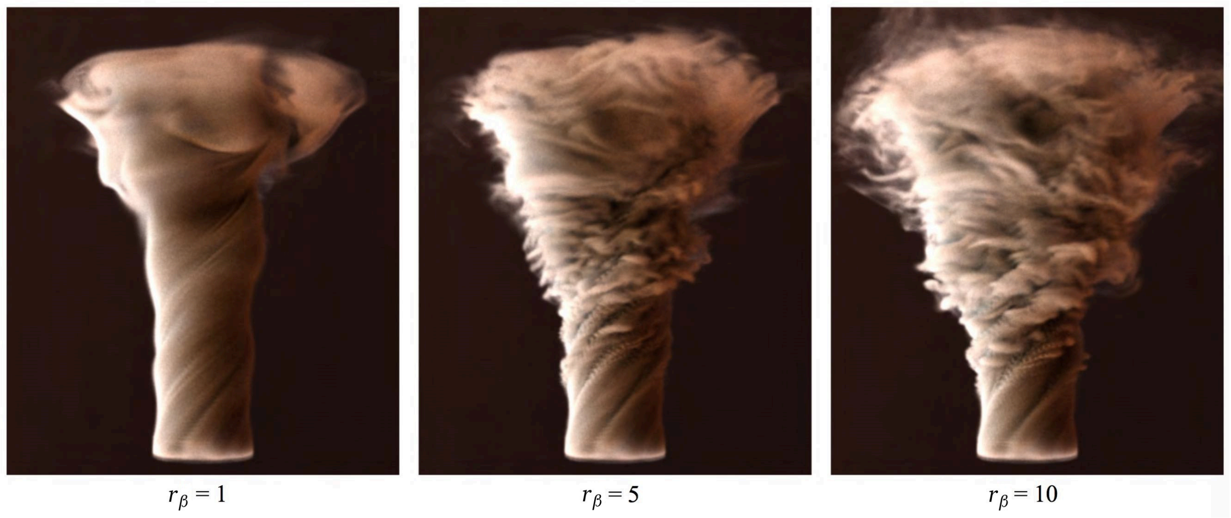


Figure 7.12.: Base tornado simulation on $40^2 \times 60$, re-simulations o $200^2 \times 300$ with varying r_β.

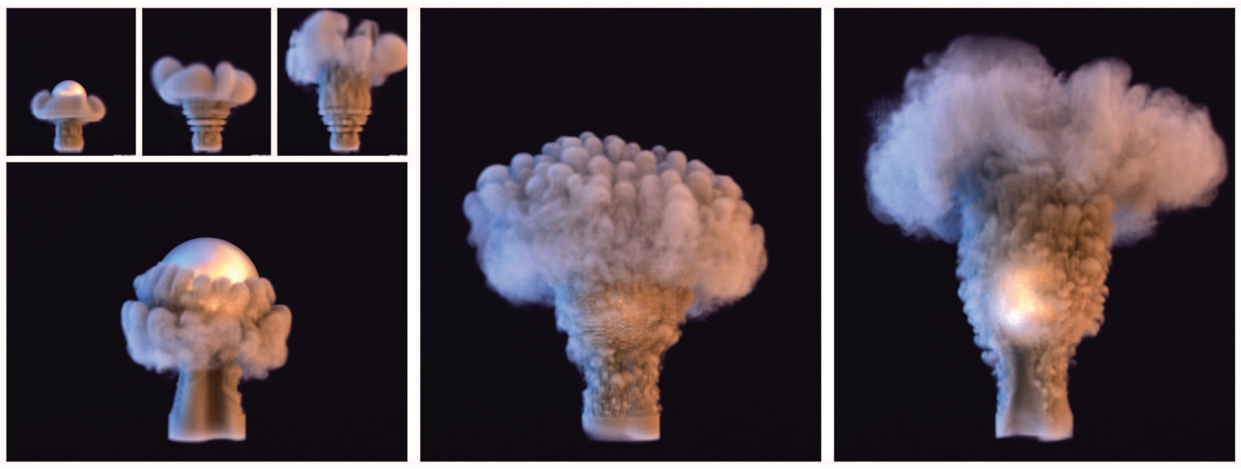


Figure 7.13.: Re-simulation on 200^3 of a base simulation on 40^3 with an obstacle (inset) with $W = 1$ and $r_\beta = 5$.

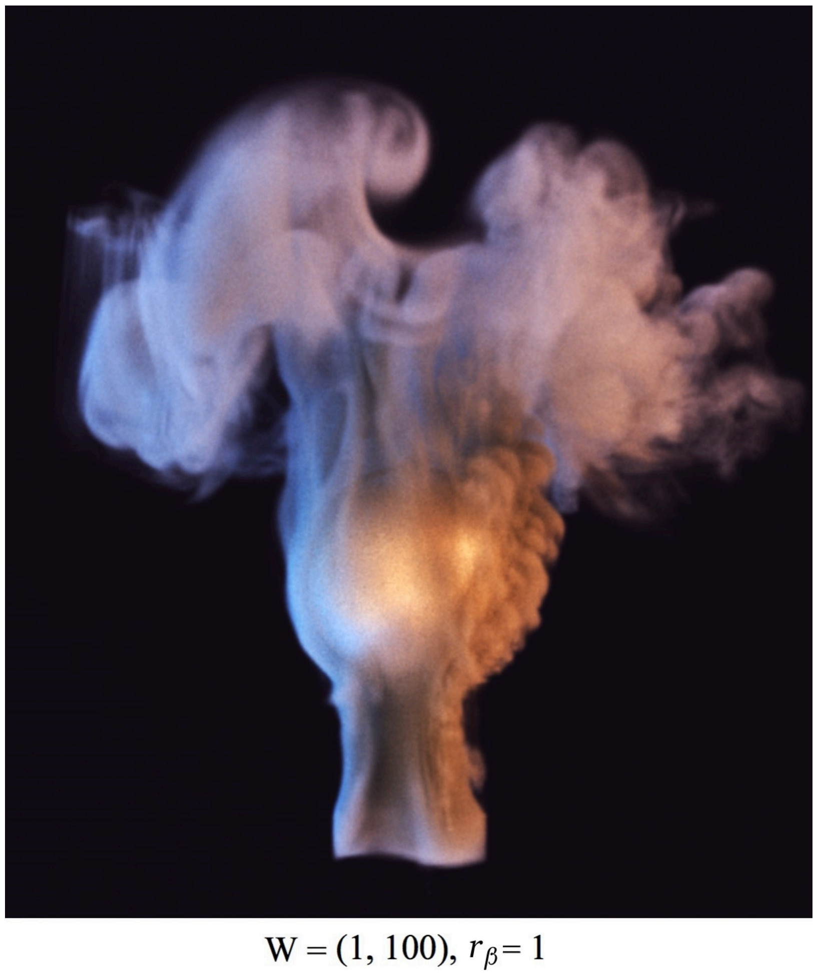


Figure 7.14.: Guided simple plume with sphere as obstacle and with spatially varying weights.

8 Separating Solid-Wall Boundary Conditions

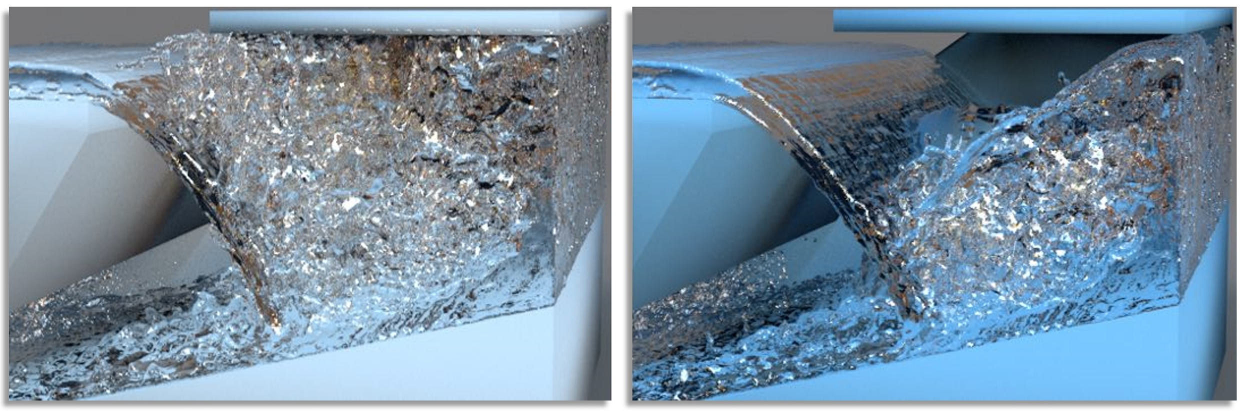


Figure 8.1.: Non-separating and separating solid-wall boundary conditions in 3D with multiple static obstacles on a $256 \times 230 \times 256$ grid. Regular, non- BCs lead to the liquid sticking to walls (left), while our solver allows for naturally separating liquids (right).

In this chapter, we present our adapted CG solver accounting for separating solid-wall BCs in its standard and accelerated variants. Considering the standard approach, PD converges faster than ADMM and IOP. However, IOP outperforms PD in the accelerated version, as only very few iterations are required. Hence, PD is not able to fully exploit its advanced variable updates. In Figure 8.1 and Figure 8.2, we illustrate the unwanted artifacts of fluid sticking to solid obstacles and domain boundaries for regular FLIP liquid simulations and demonstrate the results of our adapted CG solver accounting for separating solid-wall BCs leading to clean separation. When simulating large-scale fluid flow, the separating effect is more realistic and hence favored. The results for both fluid guiding and separating solid-wall BCs can be found in our accompanying video[1].

8.1. Method

We realize separating BCs with the inequality constraint $\mathbf{u} \cdot \hat{\mathbf{n}} \geq 0$, where $\mathbf{u}$ and $\hat{\mathbf{n}}$ are fluid velocity and the normal at fluid-solid faces, as outlined in Section 1.4. Our goal is to keep the system of equations in the pressure solver linear. In order to achieve both separating BCs while retaining the linear system

[1] `https://youtu.be/Pgbat5MXo8Q`

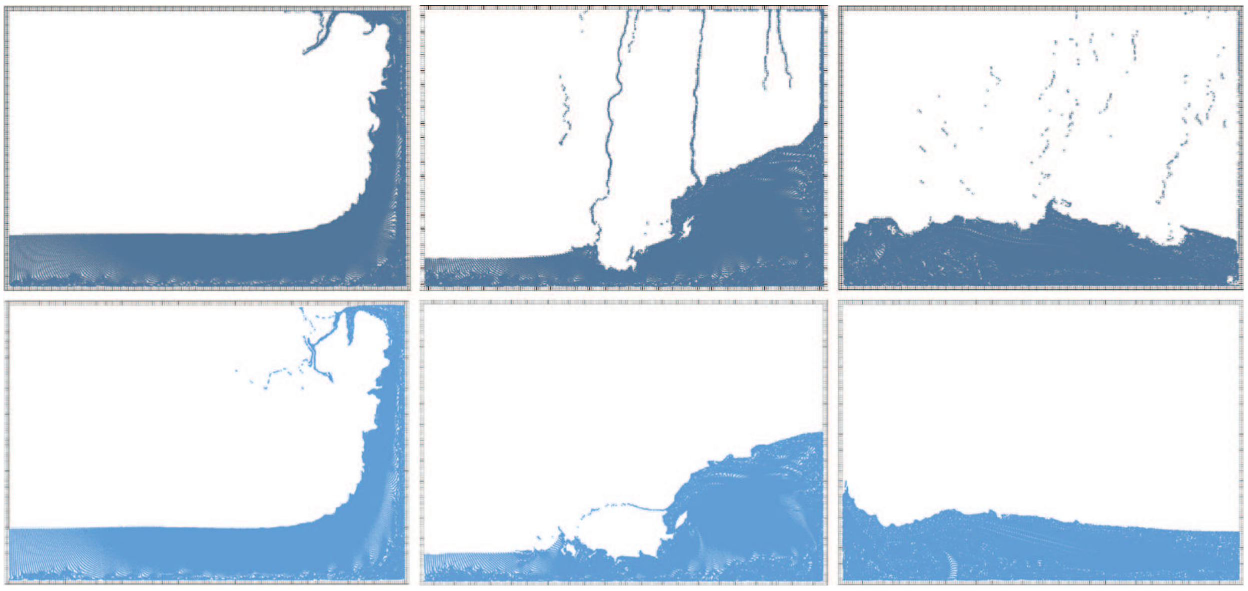


Figure 8.2.: 2D breaking dam with non-separating (top) vs. separating BCs (bottom) on a 200×140 grid at $t = 75, 112, 150$.

in the pressure solver, we detach the handling of solid-wall BCs from the pressure solver making the velocity field divergence-free. Hence, we make use of PD, see Section 3.2, where f controls solid-wall BCs and g enforces free-surface BCs and zero divergence with a regular CG solver, similar to Equation (3.16). To handle solid-wall BCs in f, we classify fluid-solid faces as separating or non-separating. Separating faces are solid walls where the fluid is not pushing against but rather moving away from. Hence, separating faces should allow for positive normal velocity components. Non-separating faces specify that fluid is pushing into the obstacle. We do not need to account for separating motions but project the velocity at non-separating faces to the space of velocities that do not flow into solid obstacles. We assume static obstacles, which have zero velocity on their surface. However, it is straightforward to implement our method for moving obstacles.

Separating Solid-Wall BCs Our proximal operator $\mathbf{prox}_{f,\sigma}(\xi)$ is responsible for velocities at fluid-solid faces and hence, to allow fluid to detach from detach from obstacles while preventing the fluid to flow into them, i.e., negative normal velocity components are prohibited: $\mathbf{u} \cdot \hat{\mathbf{n}} \geq 0$. BCs forming a linear constraint, here on the velocity, can be realized as orthogonal projections [MCPN08]. As we require normal velocity components to be zero, our proximal operator is an orthogonal projection onto the space of velocity fields with non-negative normal velocity components at fluid-solid faces $C_{\text{NN normal}}$. We denote this orthogonal projection with $\Pi_{\text{NN normal}}$, which is easily implemented by setting negative normal components to zero. Our proximal operator handling solid-wall BCs is given by

$$\mathbf{prox}_{f,\sigma}(\xi) = \Pi_{\text{NN normal}}(\xi).$$

The second proximal operator is the common $\mathbf{prox}_{g,1/\tau}(\xi)$ in Equation (3.16), which is, however,

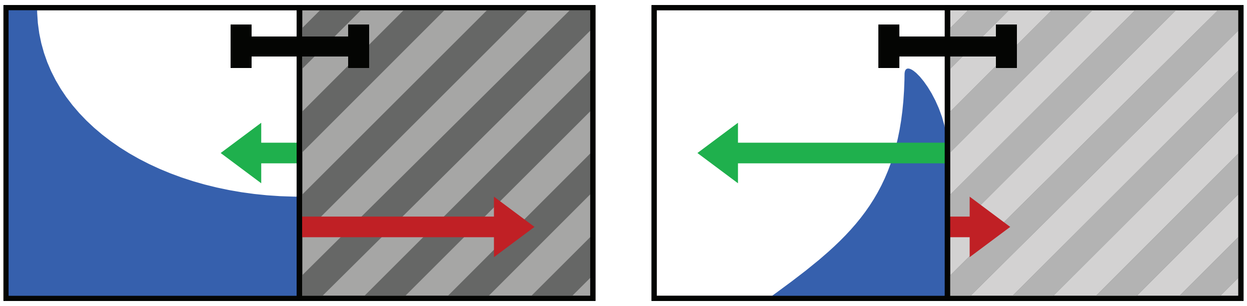


Figure 8.3.: Classification of solid faces into non-separating (left) and separating (right). The green and red arrows denote $\mathbf{u}_i \cdot \hat{\mathbf{n}}_i$ and $\mathbf{m}_i$ respectively, while the black bars indicate ϵ_{CG}.

not aware of any solid obstacle in this case. The pressure solver treats solid-wall BCs as free-surface with $p = 0$ in solid cells. In Section 8.1.1, we propose an accelerated version where the pressure solver includes the handling of solid-wall BCs at non-separating faces. Yet, in the standard version, we cleanly separate the handling of solid-wall BCs from the pressure solver.

While these two projections in $\mathbf{prox}_f$ and $\mathbf{prox}_g$ would work for exact solutions, small positive velocities are accumulated during PD iterations, as the pressure solver exhibits only finite accuracy. These accumulated velocities cause small separating motions where the fluid should only move tangentially or stand still, such as for hydrostatic cases with exactly zero velocities. To circumvent such small motions, we present a classification step with temporal coherence.

We first assume that all our fluid-solid faces i are separating, and create a list of non-separating faces S_{nsep}. The classification of separating and non-separating faces is only altered, if the absolute value of the normal velocity component at the given face is larger than the accuracy of the CG solver, i.e., $|\mathbf{u}_i \cdot \hat{\mathbf{n}}_i| > \epsilon_{\text{CG}}$. A separating face is classified as non-separating, if the fluid has a negative or zero normal velocity at its face, i.e., $\mathbf{u}_i \cdot \hat{\mathbf{n}}_i \leq 0$. At non-separating faces, zero normal velocity components are enforced, as positive normal velocity components are prohibited. In order to ensure temporal coherence, i.e., prevent small numerical errors to affect classification, we accumulate negative normal velocity components created by the pressure solver in each PD iteration. We store these accumulated velocities in our memory field $\mathbf{m}_i$. A non-separating face is classified as separating if its normal velocity component is positive and if its absolute value exceeds the magnitude of the memory field, i.e., $\mathbf{u}_i \cdot \hat{\mathbf{n}}_i > 0$ and $|\mathbf{u}_i \cdot \hat{\mathbf{n}}_i| \geq \mathbf{m}_i$. Conditioning classification with the memory field yields a hysteresis preventing inaccuracies to change classification states and therefore leads to reliable convergence.

The classification procedure is portrayed in Figure 8.3 by means of two fluid-solid faces with positive normal velocity. In both cases, the faces have the potential to changes state, as the absolute value of the normal velocity (green) exceeds the CG's accuracy threshold, i.e. ϵ_{CG}, $|\mathbf{u}_i \cdot \hat{\mathbf{n}}_i| > \epsilon_{\text{CG}}$. On the left, fluid is pushing against a solid cell. Although the normal velocity is positive, the non-separating classification of the face is not changed, as the previously accumulated negative normal velocities in $\mathbf{m}_i$ (red) are larger. The classification as a non-separating face leads to setting the green positive normal velocity to zero later on. As visualized, the fluid on the right side of Figure 8.3 is not pushing against the obstacle but rather flowing tangentially or even separating from it. This is indicated by the large positive normal

Algorithm 10 Classification of fluid-solid faces

```
1: procedure CLASSIFY($\mathbf{u}$, $\mathbf{m}$, $\epsilon_{CG}$, $S_{nsep}$)
2:     for all boundary faces $i$, with $|\mathbf{u}_i \cdot \hat{\mathbf{n}}_i| \geq \epsilon_{CG}$ do
3:         if $\mathbf{u}_i \cdot \hat{\mathbf{n}}_i \leq 0$ then
4:             $S_{nsep} \leftarrow S_{nsep} \cup \{i\}$                    ▷ non-separating
5:             $\mathbf{m}_i \leftarrow \mathbf{m}_i + \mathbf{u}_i \cdot \hat{\mathbf{n}}_i$
6:         else if ( $|\mathbf{u}_i \cdot \hat{\mathbf{n}}_i| \geq |\mathbf{m}_i|$ ) then
7:             $S_{nsep} \leftarrow S_{nsep} \setminus \{i\}$                    ▷ make separating
8:             $\mathbf{m}_i \leftarrow 0$
9:         end if
10:    end for
11: end procedure
```

Algorithm 11 $\mathbf{prox}_{f,\sigma}$ for separating solid-wall BCs

```
1: procedure PROXF($\xi$, $S_{nsep}$)        ▷ implements $\Pi_{\text{NN normal}}(\xi, S_{nsep})$ for non-separating faces
2:     for all $i \in S_{nsep}$ do $\xi_i \leftarrow \xi_i - (\xi_i \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}$
3:     return $\xi$
4: end procedure
```

velocity, which is greater than the accumulated memory field. Hence, the liquid boundary is treated as free surface and the velocity is allowed to move freely.

We outline our classification procedure in Algorithm 10. After initializing all fluid-solid faces to be separating, we perform classification on $\mathbf{z}$, which is the solution of PD. This means that we call CLASSIFY($\mathbf{u}$, $\mathbf{m}$, ϵ_{CG}, S_{nsep}) in Algorithm 1 after line 4. As described above, the classification procedure updates the list on non-separating faces S_{nsep} based on the current velocity $\mathbf{u}$ and the memory field $\mathbf{m}$. In the proximal operator $\mathbf{prox}_{f,\sigma}(\xi)$, we set the normal velocity components of non-separating fluid-solid faces to zero, as specified in Algorithm 11.

Convergence Speedup In order to improve convergence, we adapt a Krylov method for IOP [MCPN08], as we found it to speed up our BCs solver. The corresponding pseudocode can be found in Section 3.2.3. Furthermore, we apply an adaptive parameter scheme [CP11] to dynamically adapt the values of σ, τ, θ based on a novel parameter γ. We choose the following initial values $(\sigma^0, \tau^0, \gamma^0) = (1/\tau^0, 150, 200)$. The parameter updates are given by

$$\begin{aligned}\theta^k &\leftarrow 1/\sqrt{1+2\tau^{k-1}\gamma}\\ \sigma^k &\leftarrow \sigma^{k-1}/\theta^k\\ \tau^k &\leftarrow \tau^{k-1}\theta^k.\end{aligned}$$

Both extensions greatly speed up convergence, but are specific for proximal operators defined as orthogonal projections.

Algorithm 12 Accelerated separating BCs solver

```
1: procedure SOLVEPRESSUREWITHSEPARATINGBCS($\mathbf{u}$, $\epsilon_{CG}$)
2:     $S_{nsep} = 0$                                          ▷ no face is classified as non-separating yet
3:     CLASSIFY($\mathbf{u}$, $\cdot$, $\epsilon$, $S_{nsep}$)                         ▷ Algorithm 10
4:     while $S_{nsep}$ did not change in last call of CLASSIFY do
5:         $\mathbf{u} = $ PROXF($\mathbf{u}$, $S_{nsep}$)                        ▷ Algorithm 11
6:         $\mathbf{u} = \Pi_{DIV}(\mathbf{u}, \epsilon_{CG})$                        ▷ Equation (3.16)
7:         CLASSIFY($\mathbf{u}$, $\cdot$, $\epsilon$, $S_{nsep}$)                     ▷ Algorithm 10
8:     end while
9: end procedure
```

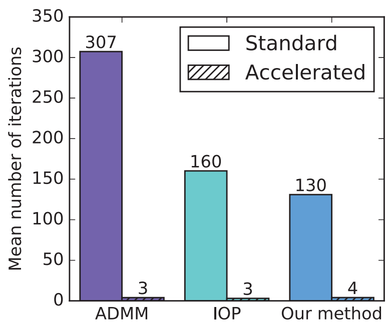
8.1.1. Accelerated Solver

Our presented variant of BCs solver follows the standard splitting approach as in other works [MCPN08, Hen12]. However, we can reduce the total number of iterations significantly if we allow the pressure solver to partially handle solid-wall BCs as well, namely, the ones it is capable of enforcing. Hence, we include the solid-wall BCs at non-separating faces, where $\mathbf{u} \cdot \hat{\mathbf{n}} = 0$, in the pressure solver instead of in f. The pressure solver thus uses Neumann BCs for non-separating fluid-solid faces and, as before, Dirichlet (free-surface) BCs for separating faces.

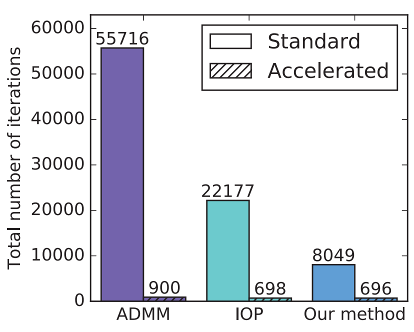
We outline the pseudocode for our accelerated variant in Algorithm 12. We first classify fluid-solid faces into separating and non-separating ones depending on the initial velocity field. Then, we iterate over setting normal velocities at non-separating faces to zero in Algorithm 11 (line 5), which implements $\Pi_{\text{NN normal}}(\xi, S_{\text{nsep}})$, using a regular CG solver to enforce incompressibility (line 6), and reclassifying fluid-solid faces with Algorithm 10 (line 7). As the pressure solver enforces zero velocities at non-separating fluid-solid faces, such faces cannot change back to separating. This is a difference to the standard variant leading to slight deviations in the resulting velocity. In practice, we found these differences to be negligible. Hence, classification is locked. Therefore, the accelerated BCs solver does not require the memory field $\mathbf{m}$, where we instead use the placeholder "$\cdot$". As classification of non-separating faces is locked, our the accelerated version converges to a stable solution within few iterations.

8.2. Evaluation and Results

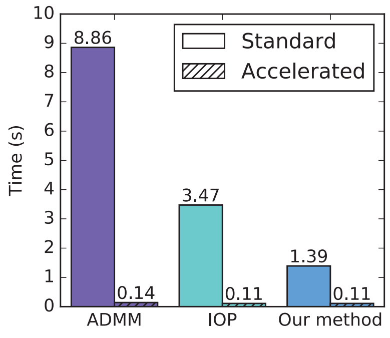
We first validate our separating solid-wall BCs solver on a 200×140 breaking dam scene by classifying all fluid-solid faces as non-separating, i.e., enforcing regular, non-separating BCs. We found our solver to produce the same results as a common CG pressure solver obtaining arbitrary levels of accuracy depending on the choice of ϵ_{CG} and PD stopping criterion threshold ϵ. Hence, our solver converges to the correct solution for non-separating BCs. In regard to separating BCs, we only compare our PD-based solver to ADMM- and IOP-based solvers from previous convex optimization works, as the regular



(a) IOP/ADMM/PD iterations



(b) Sum of CG iterations



(c) Run time in seconds

Figure 8.4.: 2D breaking dam with mean quantities per time step.

CG solver is unable to produce such separating effects. In all our experiments, we choose $\epsilon_{CG} = 10^{-5}$ and $\epsilon^{abs} = \epsilon^{rel} = 10^{-3}$.

When evaluating performance, we compare both PD/ADMM/IOP and total CG iteration numbers. Both influence run time, but the number of CG iterations accumulated over PD/ADMM/IOP iterations has stronger impact as shown in Figure 8.4 for a 2D breaking dam scene. The number of PD iterations is slightly higher compared to ADMM and IOP due to our adaptive CG accuracy scheme. However, our method is six and more than two times faster than ADMM and IOP, respectively, as shown in Figure 8.4c. We need less PD iterations than ADMM and IOP iterations, but even less total CG iterations, as presented in Figure 8.4a and Figure 8.4b.

Performance measurements for a complex 3D breaking dam simulation with multiple solid obstacles on a $256 \times 230 \times 256$ grid lead to similar results as shown in Figure 8.5. Due to ADMM's poor performance and hence infeasible run times, we omit such a comparison in the 3D case. The accelerated variant of our BCs solver seriously speeds up each method, such that PD performs on par with IOP. As the accelerated version requires only very few iterations, namely 3-4, the higher-order convergence of PD is not fully exploited.

A visualization of our complex breaking dam is presented in Figure 8.1 where the importance of liquid separating smoothly from solids becomes obvious. As illustrated, non-separating solid-walls yield undesirable fluid behavior of large amounts of liquid crawling up walls and sticking to obstacles. Our method produces much more plausible large-scale splashy motions with liquid separating naturally from solids.

Our accelerated separating BCs solver takes 38.7 s on average per time step. Considering the overall run time for one frame, including surface generation, our method requires only 12 % more computational time compared to a regular pressure solver while allowing for separating large-scale motion. Such increase of run time is a small effort considering the use of a simple CG solver without the need for specialized techniques, such as LCP solvers [GB13]. Even though LCPs have been researched on in detail and well-performing solvers exists, such as [ANE17], these problems still belong to complexity classes making high-resolution simulations challenging. For example, Gerszewski and Bargteil [GB13]

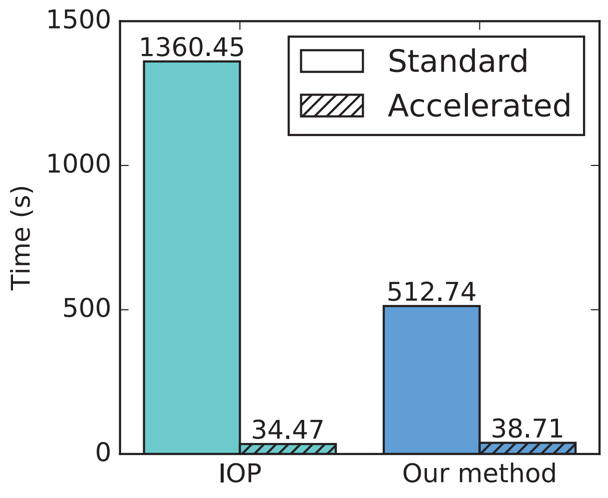
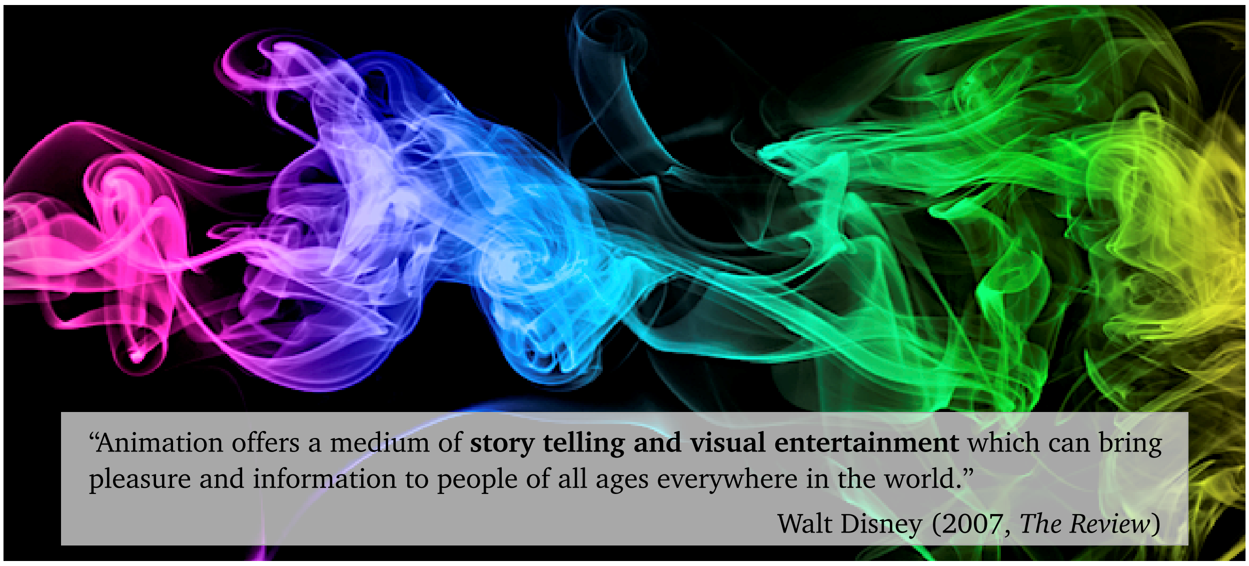


Figure 8.5.: 3D complex breaking dam with mean run time in seconds per time step.

require approximately 25 s for each LCP solve on a 100^3 grid while the solver by Andersen et al. [ANE17] needs 46.92 s using an NVIDIA GeForce GTX 1080 and a 100^3 grid.

9 Conclusion and Outlook



“Animation offers a medium of **story telling and visual entertainment** which can bring pleasure and information to people of all ages everywhere in the world.”

Walt Disney (2007, *The Review*)

In this chapter, we summarize and conclude the methods we developed and the results we achieved. Then, we discuss limitations and outline ideas for future work.

Summary and Conclusion In this thesis, we have introduced PD, a *fast first-order Primal-Dual method* proposed by Pock et al. [PCBC09], to fluid simulation for solving several problems targeting more controllable, realistic, and artifact-free fluid flows for animation. This fast convergent optimization framework is applicable to a large variety of problems in fluid simulation and other research areas. We have developed powerful techniques for sparse- or single-view reconstruction of real-world phenomena, flexible fluid guiding, and separating solid-wall BCs, which altogether enable more controllable and realistic fluid effects.

We have presented a novel, simple, easy-to-use, and affordable hardware setup with inexpensive capturing sensors, controllable and influenceable smoke generation, and an automated, accurate multi-plane calibration procedure used to obtain pixel-voxel correspondences. For inexpensive optical sensors, we have used cameras mounted on Raspberry Pis. Manipulable smoke motions in terms of speed, volume, and turbulence have been achieved by controlling the temperature of the smoke and the amount of in- and outflow of air. Compared to high-quality, professional capturing setups, our hardware setup is easily reproducible and available to a wide range of people. This setup has enabled us to capture repeatable and controllable real fluid phenomena from different viewing angles, which can be used for digital reconstructions.

In order to reconstruct volumetric and complex real-world fluid flows from only sparse views and with inexpensive sensors, we have constrained a forward fluid simulation to match the given input videos by using powerful physical constraints, such as tying the residual density to the velocity. We have also proposed a novel inflow solver for estimating the unseen density inflow and an efficient approach of incorporating regularizers into a fast tomography solver. The resulting dense and realistic flow fields can be used for re-simulation, novel visualizations, further editing, incorporation into visual effects scenes, or guiding operations.

With multiple input video streams, i.e., five, we have reconstructed real-world smoke plumes accurately and created a large-scale data set *ScalarFlow* of realistic and turbulent scalar flows. Each reconstruction comprises reconstructed volumetric density and velocity, captured videos, calibration data, and the corresponding algorithmic framework. Our data set is suitable for many application, e.g., as real-world reference data for evaluating simulation methods or for training neural networks. A first perceptual evaluation study has demonstrated *ScalarFlow*'s usefulness. The study concludes that the simulation methods *advection-reflection* and *wavelet turbulence* represent real flows best among other simulation methods. Further, our reconstructions yield dynamics that are comparable to finely resolved forward simulations. To enable the use of much more general input streams including smartphone recordings, we have proposed a method for solving the strongly under-determined reconstruction problem of using just one input video. By making use of strong physical constraints and regularizers, our results outperform the reconstruction quality of previous work.

For exerting both large-scale and small-scale control over a fluid simulation while permitting the creation of physically-based, interesting fluid characteristics, we have presented a novel formulation for fluid guiding. Our guiding scheme can easily be integrated into existing solvers without the need for complicated extensions. In particular, a simulation is guided to be arbitrarily close to a target velocity field prescribed by an artist or a low-resolution simulation allowing for artistic modulations as well as up-res effects. Furthermore, we have efficiently adapted the commonly used CG pressure solver to account for separating solid-wall BCs when simulating liquids. Here, we have classified faces between fluid and solids into separating and non-separating faces, and split the tasks of enforcing incompressibility and BCs over two proximal operators. The resulting simulation produces visually appealing large-scale liquid behavior with an increase of only 12 % of run time.

We have shown that it can be appropriate to use IOP instead of PD if all constraints can be realized as orthogonal projections and if only very few iterations are required for convergence like for the case of accelerated separating solid-wall BCs.

Discussion and Outlook Regarding our real-world fluid capturing setup, we have found it difficult to tune the direction of the smoke plumes due to ambient air motions. We plan to either limit such motions by installing damping cloth or to make such motions more controllable by influencing them actively, e.g., through directed air flow. Furthermore, we intend to optimize for initial and boundary conditions for our reconstructions, e.g., position, size, shape, and amount of density source and amount of velocity inflow over time, to further reduce discrepancies between re-projections and input images. It would also be very beneficial to automate reconstruction parameters, e.g., PD convergence parameters for practical settings or weights for the smoothness and kinetic energy regularizers. The latter need to be increased, e.g., for higher image intensities, as those lead to larger density values. Since we are solving a non-linear optimization problem, our algorithm is not guaranteed to converge, which is also the case for related approaches. However, we found our technique to work well in practice.

As future research it will be interesting to evaluate the effect of increasing the reconstruction resolution, i.e., to verify that even higher resolutions do not capture essentially more features from natural phenomena compared to moderate reconstruction resolutions. However, as storing the tomography matrix requires large parts of memory, the tomography part poses a memory bottleneck, especially when increasing the reconstruction resolution. It would be beneficial in terms of computational costs to replace our tomography solver with SART or another matrix-free method like *Stochastic Tomography* from [GKHH12]. While our linear image formation model works well with thin smoke captures, denser phenomena require more complex image formation models, which account for, e.g., scattering and absorption. Since our tomography solver is a separate module, we plan on evaluating its replacement with a data-driven approach, such as an efficient neural network trained for non-linear image formation models.

As our data set contains varying smoke plumes in terms of direction, average motion speed, or amount of smoke, we could create a multitude of reconstructions for each of plume variant and sort the plumes accordingly. In order to extend our data set further, we aim for creating a completely different effect, such as a smoke jet flowing in horizontally and rising due to buoyancy. We are confident that our captured reference data in *ScalarFlow* enables exciting future applications, such as benchmark tests, evaluation metrics, accuracy measurements, and evaluations of realism of different simulation methods. We will also employ this data to train neural networks, e.g., in order to develop new and faster tomographic reconstruction methods or velocity reconstructions, to speed up forward simulations in general, to create sparse parameter representations of simulations, or to up-res low-resolution simulations based on real-world data [SDKN18, XFCT18]. Since we do not have access to reliable information about the depth motion when reconstructing from just a single view, we intend to use data-driven regularizers to further reduce degrees of freedom along the depth direction.

With our flexible velocity-based fluid guiding method, we are not yet capable of controlling the shape itself. Therefore, it is difficult to guide liquids, as they require constraints targeting the desired shape. For such constraints, we need to take the position of the surface into account, which should integrate well into our optimization pipeline. We could further improve an artist's workflow by providing automatic generation of target velocity fields, i.e., such that artistic visuals are created more intuitively. Furthermore, we plan to combine our guiding scheme with our real-world reconstructions, e.g., to use the reconstructed velocity field as prescribed velocity field and guide a high-resolution simulation to investigate whether structures that are even more turbulent will be created. Our accelerated solid-wall BC solver deviates slightly from the standard version but we did not observe any visual artifacts from this highly efficient variant. Multigrid solvers such as [CMF12] could possibly outperform our solver, but the advantage of our approach is that it is modular, converges fast, and requires only small implementation changes for regular CG solvers.

In the future, we will make use of our modular, efficient, and versatile optimization scheme PD for solving many more challenging fluid optimization problems, such as highly complex inverse problems. Moreover, with our real-world fluid data set, we enable further investigation of advanced fluid behavior and aim for gaining new insights into the mysteries of turbulence. We intend to train neural networks to learn the difference between simulated fluid data and real-world fluids, which could not only benefit fluid animation due to reduced run time and increased realism but also lead to better understanding of fluid behavior in general.

A
List of Hardware Components for our Capturing Setup

The prices listed for each component are the ones valid when buying the items. The sum in Euros is 986 €, which makes up less than 1100 $.

1. **Black molton cloth:** dimensions 3 m × 2 m, 10 €
   `https://www.amazon.de/dp/B0099352F2/ref=cm_sw_em_r_mt_dp_U_SaWpDbRP0H28D`
2. **Clips for cloth:** 10 €
   `https://www.amazon.de/dp/B00SFJ1EHG/ref=cm_sw_em_r_mt_dp_U_xdWpDbE6FAQS6`
3. **Raspberry Pi:** version 1, 2, or 3 ≈ 35 €, SD card 10 €, power cable 10 €, seven pieces
   `https://www.conrad.de/de/p/raspberry-pi-3-b-1-gb-4-x-1-2-ghz-raspberry-pi-1419716.html`
4. **Raspberry Pi camera module:** V1 35 €, microphone stands 10 €, five pieces
   `https://www.conrad.de/de/p/raspberry-pi-camera-module-v1-5mp-cmos-farb-kameramodul-passend-fuer-raspberry-pi-622959.html`
5. **Fog machine:** Eurolite N-10, dimensions 5.7 cm × 10.8 cm × 6.7 cm, 37 €
   `https://www.amazon.de/dp/B005DQF9NY/ref=cm_sw_em_r_mt_dp_U_njWpDbMVN4QS1`
6. **Items for driving remote control of fog machine:** 10 €
   a) relais: `https://www.amazon.de/dp/B07CNR7K9B/ref=cm_sw_em_r_mt_dp_U_8jWpDbS84HM7E`
   b) strip board: `https://www.conrad.de/de/p/phoenix-contact-eh-45f-16-dev-pcb-lochrasterplatine-gruen-1-st-1833475.html`
   c) optical coupler: `https://www.conrad.de/de/p/vossloh-schwabe-optokoppler-phototransistor-ltv827-dip-8-transistor-dc-187003.html`
7. **Fog fuid:** EUROLITE Smoke Fluid – P, 7.5 €
   `https://www.conrad.de/de/p/nebelfluid-eurolite-p2d-profi-extrem-1-l-1432431.html`

8. **Machine cleaner:** Nebelmaschinen-Reiniger 590298, 16 €
   `https://www.conrad.de/de/p/nebelmaschinen-reiniger-590298-1-l-590298.html`
9. **Servo motor:** Absima Standard-Servo S60MH Analog-Servo, 14.5 € , two pieces
   `https://www.conrad.de/de/p/absima-standard-servo-s60mh-analog-servo-getriebe-material-metall-stecksystem-jr-1677457.html`
   a) previous version, performing sufficiently: 4er MG90S Mini Metal Gear Analog Servo, 5 €
      `https://www.conrad.de/de/p/modelcraft-standard-servo-bms-410c-analog-servo-getriebe-material-kunststoff-stecksystem-jr-404753.html`
   b) previously used servo for side opening, not needed anymore: Modelcraft Mini-Servo MC1811 Analog-Servo, 4 €
      `https://www.conrad.de/de/p/modelcraft-mini-servo-mc1811-analog-servo-getriebe-material-kunststoff-stecksystem-jr-275460.html`
10. **Silicone hose:** Schlauchland, 4 €
    `https://www.amazon.de/dp/B0180PZR5E/ref=cm_sw_em_r_mt_dp_U_SrWpDbBX51Y2Y`
11. **Styrofoam box:** Thermobox, expanded polystyrene foam (EPS), dimensions 54.5 cm × 35 cm × 30 cm, 9 €
    `https://www.hornbach.de/shop/Thermobox-mit-Deckel-gross/5991349/artikel.html`
12. **Heating cable:** Terrarium heating cable, 17.5 €
    `https://www.amazon.de/dp/B01N95UWEO/ref=cm_sw_em_r_mt_dp_U_TsWpDbY9KZCYK`
13. **Metallic fence:** hadra G12500V5I verzinktes, punktgeschweißtes Gitter, 16 €
    `https://www.amazon.de/dp/B00GR0XS92/ref=cm_sw_em_r_mt_dp_U_hxWpDb6RW43FV`
14. **Controller thermostat:** KKmoon Thermostat Controller, 14 €
    `https://www.amazon.de/dp/B01EFSD04U/ref=cm_sw_em_r_mt_dp_U_cyWpDbNFY1BVR`
15. **Timer:** GAO Mechanischer Countdown Timer, 10 €
    `https://www.amazon.de/dp/B00KACQ00W/ref=cm_sw_em_r_mt_dp_U_AyWpDbZZSKNHE`
16. **Stepping motor:** Quimat Nema 17 Schrittmotor, 24 €
    `https://www.amazon.de/dp/B06XT3HKRX/ref=cm_sw_em_r_mt_dp_U_KuWpDbA75RDVT`
17. **Rail:** 1204 Ball Screw Linear Rail Stroke Long Stage Actuator with Stepper Motor 400MM, 75 €
    `https://www.ebay.de/itm/1204-Ball-Screw-Linear-Rail-Stroke-Long-Stage-Actuator-w-Stepper-Motor-400MM/372110386346?hash=item56a381e4aa:g:fQsAAOSwcXVZ6aSZ` (not available any more, a comparable item is `https://www.ebay.de/itm/DE-200mm-Stroke-Linear-Guide-Rail-Stage-1605-Ball-Screw-Motion-Actuator-Stepper/264383115222?hash=item3d8e762fd6:g:BuAAAOSwfjdbaUkI`)
18. **Smaller items:** network cables, wood, glue, 100 €

B Derivation of Reconstruction Optimization Problem

Here, we derive our optimization problem in Figure 4.2 step-by-step starting with the brightness constancy assumption plus constraints based on the full quantities $\mathbf{u}^t$ and Φ^t.

$$
\begin{aligned}
& \underset{\mathbf{u}^t,\Phi^t}{\text{minimize}} \quad && \left\| \frac{\partial \Phi^t}{\partial t} + \nabla\Phi^t \cdot \mathbf{u}^t \right\|^2 \\
& && \qquad\qquad \left| \begin{array}{l} \triangleright \Phi^t = \mathcal{A}(\Phi^{t-1}+\Phi_{\mathrm{I}}^t, \mathbf{u}^t), \\ \triangleright \mathbf{u}^t = \tilde{\mathbf{u}}^t + \Delta\mathbf{u}^t \end{array}\right. \\
\equiv\quad & \underset{\Delta\mathbf{u}^t,\Phi_{\mathrm{I}}^t}{\text{minimize}} \quad && \left\| \frac{\partial \mathcal{A}(\Phi^{t-1}+\Phi_{\mathrm{I}}^t, \tilde{\mathbf{u}}^t+\Delta\mathbf{u}^t)}{\partial t} + \nabla\mathcal{A}(\Phi^{t-1}+\Phi_{\mathrm{I}}^t, \tilde{\mathbf{u}}^t+\Delta\mathbf{u}^t)\cdot(\tilde{\mathbf{u}}^t+\Delta\mathbf{u}^t) \right\|^2 \\
& && \qquad\qquad \left| \begin{array}{l} \triangleright \text{linearize advection :} \\ \triangleright \mathcal{A}(\Phi^{t-1}+\Phi_{\mathrm{I}}^t, \tilde{\mathbf{u}}^t+\Delta\mathbf{u}^t) \approx \mathcal{A}(\mathcal{A}(\Phi^{t-1}+\Phi_{\mathrm{I}}^t, \tilde{\mathbf{u}}^t), \Delta\mathbf{u}^t) \end{array}\right. \\
\equiv\quad & \underset{\Delta\mathbf{u}^t,\Phi_{\mathrm{I}}^t}{\text{minimize}} \quad && \left\| \frac{\partial \mathcal{A}(\mathcal{A}(\Phi^{t-1}+\Phi_{\mathrm{I}}^t, \tilde{\mathbf{u}}^t), \Delta\mathbf{u}^t)}{\partial t} + \nabla\mathcal{A}(\mathcal{A}(\Phi^{t-1}+\Phi_{\mathrm{I}}^t, \tilde{\mathbf{u}}^t), \Delta\mathbf{u}^t)\cdot(\tilde{\mathbf{u}}^t+\Delta\mathbf{u}^t) \right\|^2 \\
& && \qquad\qquad \left| \begin{array}{l} \triangleright \Phi_{\mathrm{I}}^t \approx \tilde{\Phi}_{\mathrm{I}}^t, \\ \triangleright \nabla\Phi^t \approx \nabla\tilde{\Phi}^t \end{array}\right. \\
\equiv\quad & \underset{\Delta\mathbf{u}^t,\Phi^t}{\text{minimize}} \quad && \left\| \frac{\partial \mathcal{A}(\mathcal{A}(\Phi^{t-1}+\tilde{\Phi}_{\mathrm{I}}^t, \tilde{\mathbf{u}}^t), \Delta\mathbf{u}^t)}{\partial t} + \nabla\mathcal{A}(\Phi^{t-1}+\tilde{\Phi}_{\mathrm{I}}^t, \tilde{\mathbf{u}}^t)\cdot(\tilde{\mathbf{u}}^t+\Delta\mathbf{u}^t) \right\|^2 \\
& && \qquad\qquad \left| \begin{array}{l} \triangleright \tilde{\Phi}^t = \mathcal{A}(\Phi^{t-1}+\tilde{\Phi}_{\mathrm{I}}^t, \tilde{\mathbf{u}}^t), \\ \triangleright \text{remove constant terms} \end{array}\right. \\
\equiv\quad & \underset{\Delta\mathbf{u}^t,\Phi^t}{\text{minimize}} \quad && \left\| \frac{\partial \mathcal{A}(\tilde{\Phi}^t, \Delta\mathbf{u}^t)}{\partial t} + \nabla\tilde{\Phi}^t \cdot \Delta\mathbf{u}^t \right\|^2 \\
& && \qquad\qquad \left| \begin{array}{l} \triangleright \Delta\Phi^t = \frac{\partial\mathcal{A}(\tilde{\Phi}^t,\Delta\mathbf{u}^t)}{\partial t} = \frac{\tilde{\Phi}^t+\Delta\Phi^t-\tilde{\Phi}^t}{\Delta t}, \Delta t = 1 \end{array}\right. \\
\equiv\quad & \underset{\Delta\mathbf{u}^t,\Delta\Phi^t}{\text{minimize}} \quad && \left\| \Delta\Phi^t + \nabla\tilde{\Phi}^t \cdot \Delta\mathbf{u}^t \right\|^2 \qquad\qquad \text{(B.1)}
\end{aligned}
$$

$$
\begin{aligned}
\text{subject to}\quad & \mathbf{u}^t|_I = c && \Big|\triangleright\, \mathbf{u}^t = \tilde{\mathbf{u}}^t + \Delta\mathbf{u}^t,\ \text{where } c \text{ is inflow speed} \\
\equiv\quad & \tilde{\mathbf{u}}^t|_I + \Delta\mathbf{u}^t|_I = c \\
\equiv\quad & \Delta\mathbf{u}^t|_I = c - \tilde{\mathbf{u}}^t|_I, && \qquad\qquad\qquad \text{(B.2)} \\
& \nabla\cdot\mathbf{u}^t = 0 && \Big|\triangleright\, \mathbf{u}^t = \tilde{\mathbf{u}}^t + \Delta\mathbf{u}^t \\
\equiv\quad & \nabla\cdot(\tilde{\mathbf{u}}^t + \Delta\mathbf{u}^t) = 0 && \Big|\triangleright\, \tilde{\mathbf{u}}^t \text{ is div-free} \\
\equiv\quad & \nabla\cdot\Delta\mathbf{u}^t = 0, && \qquad\qquad\qquad \text{(B.3)} \\
& P\Phi^t - i^t = 0 && \Big|\triangleright\, \Phi^t = \mathcal{A}(\Phi^{t-1} + \Phi^t_{\mathrm{I}}, \mathbf{u}^t) \\
\equiv\quad & P\mathcal{A}(\Phi^{t-1} + \Phi^t_{\mathrm{I}}, \mathbf{u}^t) - i^t = 0 && \Big|\triangleright\, \mathbf{u}^t = \tilde{\mathbf{u}}^t + \Delta\mathbf{u}^t, \tilde{\Phi}^t_{\mathrm{I}} \approx \Phi^t_{\mathrm{I}},\ \text{linearize advection} \\
\equiv\quad & P\mathcal{A}(\mathcal{A}(\Phi^{t-1} + \tilde{\Phi}^t_{\mathrm{I}}, \tilde{\mathbf{u}}^t), \Delta\mathbf{u}^t) - i^t = 0 && \Big|\triangleright\, \tilde{\Phi}^t = \mathcal{A}(\Phi^{t-1} + \tilde{\Phi}^t_{\mathrm{I}}, \tilde{\mathbf{u}}^t) \\
\equiv\quad & P\mathcal{A}(\tilde{\Phi}^t, \Delta\mathbf{u}^t) - i^t = 0 && \Big|\triangleright\, \mathcal{A}(\tilde{\Phi}^t, \Delta\mathbf{u}^t) = \tilde{\Phi}^t + \Delta\Phi^t \Rightarrow \Phi^t \approx \tilde{\Phi}^t + \Delta\Phi^t \\
\equiv\quad & P\tilde{\Phi}^t + P\Delta\Phi^t - i^t = 0, && \Big|\triangleright\, \tilde{i}^t = P\tilde{\Phi}^t \\
\equiv\quad & P\Delta\Phi^t - (i^t - \tilde{i}^t) = 0, && \Big|\triangleright\, \Delta i^t = i^t - \tilde{i}^t \\
\equiv\quad & P\Delta\Phi^t - \Delta i^t = 0, && \qquad\qquad\qquad \text{(B.4)} \\
& \Phi^t \geq 0 && \Big|\triangleright\, \Phi^t \approx \tilde{\Phi}^t + \Delta\Phi^t \\
\equiv\quad & \tilde{\Phi}^t + \Delta\Phi^t \geq 0 \\
\equiv\quad & \Delta\Phi^t \geq -\tilde{\Phi}^t\ . && \qquad\qquad\qquad \text{(B.5)}
\end{aligned}
$$

Bibliography

[AEHKL*16] ABU-EL-HAIJA S., KOTHARI N., LEE J., NATSEV P., TODERICI G., VARADARAJAN B., VIJAYANARASIMHAN S.: Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675* (2016). 16

[AHOGG14] AVSARKISOV V., HOYAS S., OBERLACK M., GARCIA-GALACHE J. P.: Turbulent plane Couette flow at moderately high Reynolds number. *Journal of Fluid Mechanics 751* (2014). 16

[AIH*08] ATCHESON B., IHRKE I., HEIDRICH W., TEVS A., BRADLEY D., MAGNOR M., SEIDEL H.-P.: Time-resolved 3D capture of non-stationary gas flows. In *ACM Transactions on Graphics* (2008), vol. 27, ACM, p. 132. 5, 13

[AK84] ANDERSEN A. H., KAK A. C.: Simultaneous algebraic reconstruction technique (SART): a superior implementation of the ART algorithm. *Ultrasonic Imaging 6*, 1 (1984), 81–94. 14

[ANE17] ANDERSEN M., NIEBE S., ERLEBEN K.: A fast linear complementarity problem solver for fluid animation using high level algebra interfaces for gpu libraries. *Computers & Graphics 69* (2017), 36–48. 18, 108, 109

[ANSN06] ANGELIDIS A., NEYRET F., SINGH K., NOWROUZEZAHRAI D.: A controllable, fast and stable basis for vortex based smoke simulation. In *Symposium on Computer Animation* (2006), Eurographics Association, pp. 25–32. 12

[ARL*09] ALEXANDER O., ROGERS M., LAMBETH W., CHIANG M., DEBEVEC P.: The digital emily project: photoreal facial modeling and animation. In *ACM SIGGRAPH Courses* (2009), ACM, p. 12. 12

[BB96] BAUSCHKE H. H., BORWEIN J. M.: On projection algorithms for solving convex feasibility problems. *SIAM review 38*, 3 (1996), 367–426. 95

[BB01] BEAUCHEMIN S. S., BAJCSY R.: Modelling and removing radial and tangential distortions in spherical lenses. In *Multi-Image Analysis*. Springer, 2001, pp. 1–21. 39

[BBA*07] BICKEL B., BOTSCH M., ANGST R., MATUSIK W., OTADUY M., PFISTER H., GROSS M.: Multi-scale capture of facial geometry and motion. In *ACM Transactions on Graphics* (2007), vol. 26, ACM, p. 33. 12

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. In *ACM Transactions on Graphics* (2007), vol. 26, ACM, p. 100. 9, 18

[BCM11] BUADES A., COLL B., MOREL J.-M.: Non-local means denoising. *Image Processing On Line 1* (2011), 208–212. 49

[BHN07] BRIDSON R., HOURIHAM J., NORDENSTAM M.: Curl-noise for procedural fluid flow. In *ACM Transactions on Graphics* (2007), vol. 26, ACM, p. 46. 17

[BPC*11] BOYD S., PARIKH N., CHU E., PELEATO B., ECKSTEIN J., ET AL.: Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning 3*, 1 (2011), 1–122. 18, 29, 30, 33

[BPS*08] BRADLEY D., POPA T., SHEFFER A., HEIDRICH W., BOUBEKEUR T.: Markerless garment capture. In *ACM Transactions on Graphics* (2008), vol. 27, ACM, p. 99. 12

[Bri15] BRIDSON R.: *Fluid simulation for computer graphics*. AK Peters/CRC Press, 2015. 12, 17, 22

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for Lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (2009), 493–503. 18

[BWS05] BRUHN A., WEICKERT J., SCHNÖRR C.: Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision 61*, 3 (2005), 211–231. 15

[CCL02] CATER K., CHALMERS A., LEDDA P.: Selective quality rendering by exploiting human inattentional blindness: Looking but not seeing. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology* (2002), VRST '02, ACM, pp. 17–24. 16

[CDF*18] CHANG A., DAI A., FUNKHOUSER T. A., HALBER M., NIESSNER M., SAVVA M., SONG S., ZENG A., ZHANG Y.: Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision* (2018), Institute of Electrical and Electronics Engineers Inc., pp. 667–676. 16

[Cho68] CHORIN A. J.: Numerical solution of the Navier-Stokes equations. *Mathematics of Computation 22*, 104 (1968), 745–762. 26

[Cie11] CIERNIAK R.: *X-ray computed tomography in biomedical engineering*. Springer Science & Business Media, 2011. 4, 14

[CLH16] CHEN D., LI W., HALL P.: Dense motion estimation for smoke. In *Asian Conference on Computer Vision* (2016), Springer, pp. 225–239. 15

[CMF12] CHENTANEZ N., MUELLER-FISCHER M.: A multigrid fluid pressure solver handling separating solid boundary conditions. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1191–1201. 18, 114

[CMP02] CORPETTI T., MÉMIN É., PÉREZ P.: Dense estimation of fluid flows. *Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 3 (2002), 365–380. 15

[CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: animating the interplay between rigid bodies and fluid. In *ACM Transactions on Graphics* (2004), vol. 23, ACM, pp. 377–384. 18

[CP11] CHAMBOLLE A., POCK T.: A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision 40*, 1 (2011), 120–145. 19, 32, 106

[CT17] CHU M., THUEREY N.: Data-driven synthesis of smoke flows with CNN-based feature descriptors. *ACM Transactions on Graphics 36*, 4 (2017). 16

[DAST*08] DE AGUIAR E., STOLL C., THEOBALT C., AHMED N., SEIDEL H.-P., THRUN S.: Performance capture from sparse multi-view video. vol. 27, ACM. 12

[DATSS07] DE AGUIAR E., THEOBALT C., STOLL C., SEIDEL H.-P.: Marker-less deformable mesh tracking for human shape and motion capture. In *Proceedings of IEEE Computer Vision and Pattern Recognition* (2007), IEEE, pp. 1–8. 12

[DDS*09] DENG J., DONG W., SOCHER R., LI L.-J., LI K., FEI-FEI L.: Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Computer Vision and Pattern Recognition* (2009), IEEE, pp. 248–255. 6, 16

[DFI*15] DOSOVITSKIY A., FISCHER P., ILG E., HÄUSSER P., HAZIRBAŞ C., GOLKOV V., V.D. SMAGT P., CREMERS D., BROX T.: FlowNet: Learning optical flow with convolutional networks. In *Proceedings of IEEE International Conference on Computer Vision* (2015). 15

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation*. Springer, 1996, pp. 61–76. 12

[EHT18] ECKERT M.-L., HEIDRICH W., THÜREY N.: Coupled fluid density and motion from single views. *Computer Graphics Forum 37* (2018), 47–58. 10

[Enga] Engineers Edge: Viscosity of air, dynamic and kinematic. `https://www.engineersedge.com/physics/viscosity_of_air_dynamic_and_kinematic_14483.htm`. Accessed: 2019-07-23. 45

[Engb] The Engineering ToolBox. `https://www.engineeringtoolbox.com/thermal-conductivity-d_429.html`. Accessed: 2019-07-23. 46

[Erl13] ERLEBEN K.: Numerical methods for linear complementarity problems in physics-based animation. In *ACM SIGGRAPH Courses* (2013), ACM, p. 8. 18

[ESWvO06] ELSINGA G. E., SCARANO F., WIENEKE B., VAN OUDHEUSDEN B. W.: Tomographic particle image velocimetry. *Experiments in Fluids 41*, 6 (2006), 933–947. 13

[EUT19] ECKERT M.-L., UM K., THUEREY N.: ScalarFlow: A large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *Forthcoming in ACM Transactions on Graphics* (2019). 10

[FAW*16] FERSTL F., ANDO R., WOJTAN C., WESTERMANN R., THUEREY N.: Narrow band flip for liquid simulations. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 225–232. 12

[FCG*06] FUCHS C., CHEN T., GOESELE M., THEISEL H., SEIDEL H.-P.: Volumetric density capture from a single image. In *Proceedings of International Workshop on Volume Graphics* (2006), pp. 17–22. 15

[Fec60] FECHNER G. T.: *Elemente der psychophysik*. Breitkopf & Härtel, Leipzig, 1860. 76

[Fee10] FEEMAN T. G.: *The mathematics of medical imaging*. Springer, 2010. 13

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proceedings of ACM Annual Conference on Computer Graphics and Interactive Techniques* (2001), ACM, pp. 23–30. 12, 18

[FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. In *ACM Transactions on Graphics* (2004), vol. 23, ACM, pp. 441–448. 17

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5 (1996), 471–483. 11, 17

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *Proceedings of ACM Annual Conference on Computer Graphics and Interactive Techniques* (2001), ACM, pp. 15–22. 5, 6, 12

[GB13] GERSZEWSKI D., BARGTEIL A. W.: Physics-based animation of large-scale splashing liquids. *ACM Transactions on Graphics 32*, 6 (2013), 185–1. 18, 108

[GHD03] GÉNEVAUX O., HABIBI A., DISCHLER J.-M.: Simulating fluid-solid interaction. In *Graphics Interface* (2003), vol. 2003, pp. 31–38. 17

[GITH14] GREGSON J., IHRKE I., THUEREY N., HEIDRICH W.: From capture to simulation: connecting forward and inverse problems in fluids. *ACM Transactions on Graphics 33*, 4 (2014), 139. 4, 6, 13, 15, 16, 17, 18, 34, 35, 56, 62, 73, 86, 87, 94, 99

[GKHH12] GREGSON J., KRIMERMAN M., HULLIN M. B., HEIDRICH W.: Stochastic tomography and its applications in 3D imaging of mixing fluids. *ACM Transactions on Graphics 31*, 4 (2012), 52–1. 5, 14, 30, 113

[GNS*12] GOLAS A., NARAIN R., SEWALL J., KRAJCEVSKI P., DUBEY P., LIN M.: Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Transactions on Graphics 31*, 6 (2012), 148. 12

[GOSB14] GOLDSTEIN T., O'DONOGHUE B., SETZER S., BARANIUK R.: Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences 7*, 3 (2014), 1588–1623. 30

[GP00] GILES M. B., PIERCE N. A.: An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion 65*, 3-4 (2000), 393–415. 17

[GW99] GERING D. T., WELLS W.: Object modeling using tomography and photography. In *Proceedings of IEEE Workshop on Multi-View Modeling and Analysis of Visual Scenes* (1999), IEEE, pp. 11–18. 12

[Har62] HARLOW F. H.: *The particle-in-cell method for numerical solution of problems in fluid dynamics*. Tech. rep., Los Alamos Scientific Lab., N. Mex., 1962. 11, 12

[HDN*16] HEIDE F., DIAMOND S., NIESSNER M., RAGAN-KELLEY J., HEIDRICH W., WETZSTEIN G.: ProxImaL: Efficient image optimization using proximal algorithms. *ACM Transactions on Graphics 35*, 4 (2016), 84. 19

[HED05] HAWKINS T., EINARSSON P., DEBEVEC P.: Acquisition of time-varying participating media. *ACM Transactions on Graphics 24*, 3 (July 2005), 812–815. 12, 43

[Hen12] HENDERSON R. D.: Scalable fluid simulation in linear time on shared memory multiprocessors. In *Proceedings of the Digital Production Symposium* (2012), ACM, pp. 43–52. 18, 107

[HK03] HASINOFF S. W., KUTULAKOS K. N.: Photo-Consistent 3D Fire by Flame-Sheet Decomposition. In *Proceedings of IEEE International Conference on Computer Vision* (2003), vol. 3, p. 1184. 13

[HK07] HASINOFF S. W., KUTULAKOS K. N.: Photo-consistent Reconstruction of Semitransparent Scenes by Density-sheet Decomposition. *Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 5 (2007), 870–885. 13

[HK13] HUANG R., KEYSER J.: Automated sampling and control of gaseous simulations. *The Visual Computer 29*, 6-8 (2013), 751–760. 17, 98

[HMK11] HUANG R., MELEK Z., KEYSER J.: Preview-based sampling for controlling gaseous simulations. In *Symposium on Computer Animation* (2011), ACM, pp. 177–186. 17, 98

[HRH*13] HEIDE F., ROUF M., HULLIN M. B., LABITZKE B., HEIDRICH W., KOLB A.: High-quality computational imaging through simple lenses. *ACM Transactions on Graphics 32*, 5 (2013), 149–1. 19

[HRZ*13] HOYET L., RYALL K., ZIBREK K., PARK H., LEE J., HODGINS J., O'SULLIVAN C.: Evaluating the distinctiveness and attractiveness of human motions on realistic virtual bodies. *ACM Transactions on Graphics 32*, 6 (2013), 204:1–204:11. 16

[HS81] HORN B. K., SCHUNCK B. G.: Determining optical flow. *Artificial Intelligence 17*, 1-3 (1981), 185–203. 15

[Hun04] HUNTER D. R.: MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics 32*, 1 (2004), 384–406. 77

[HW65] HARLOW F. H., WELCH J. E.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids 8*, 12 (1965), 2182–2189. 11, 27

[IEGT17] INGLIS T., ECKERT M.-L., GREGSON J., THUEREY N.: Primal-dual optimization for fluids. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 354–368. 10

[IGM05] IHRKE I., GOIDLUECKE B., MAGNOR M.: Reconstructing the geometry of flowing water. In *Proceedings of IEEE International Conference on Computer Vision* (2005), vol. 2, IEEE, pp. 1055–1060. 14

[IM04] IHRKE I., MAGNOR M.: Image-based tomographic reconstruction of flames. In *Symposium on Computer Animation* (2004), Eurographics Association, pp. 365–373. 5, 14, 37, 39, 40, 53, 59, 62

[IM06] IHRKE I., MAGNOR M.: Adaptive grid optical tomography. *Graphical Models 68*, 5-6 (2006), 484–495. 14

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH fluids in computer graphics. In *Eurographics 2014 - State of the Art Reports* (2014), Lefebvre S., Spagnuolo M., (Eds.), The Eurographics Association. 12

[JST*16] JIANG C., SCHROEDER C., TERAN J., STOMAKHIN A., SELLE A.: The material point method for simulating continuum materials. In *ACM SIGGRAPH Courses* (2016), ACM, p. 24. 12

[KCAT*19] KIM B., C. AZEVEDO V., THUEREY N., KIM T., GROSS M., SOLENTHALER B.: Deep Fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum 38*, 2 (2019). 16

[KCG*90] KAVANDI J., CALLIS J., GOUTERMAN M., KHALIL G., WRIGHT D., GREEN E., BURNS D., MCLACHLAN B.: Luminescent barometry in wind tunnels. *Review of Scientific Instruments 61*, 11 (1990), 3340–3347. 12

[KLLR05] KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: FlowFixer: Using BFECC for fluid simulation. In *Proceedings of the First Eurographics conference on Natural Phenomena* (2005), pp. 51–56. 12

[KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH* (1990), vol. 24, ACM, pp. 49–57. 11

[KNH14] KRIZHEVSKY A., NAIR V., HINTON G.: The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html 55* (2014). 16

[KPZK17] KNAPITSCH A., PARK J., ZHOU Q.-Y., KOLTUN V.: Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics 36*, 4 (2017), 78. 16

[KSW02] KAK A. C., SLANEY M., WANG G.: Principles of computerized tomographic imaging. *Medical Physics 29*, 1 (2002), 107–107. 14

[KTJG08] KIM T., THÜREY N., JAMES D., GROSS M.: Wavelet turbulence for fluid simulation. In *ACM Transactions on Graphics* (2008), vol. 27, ACM, p. 50. 12, 17, 77, 95, 98

[Lau94] LAURENTINI A.: The visual hull concept for silhouette-based image understanding. *Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 2 (1994), 150–162. 38

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *ACM Transactions on Graphics* (2004), vol. 23, ACM, pp. 457–462. 12

[LJS*15] LADICKY L., JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M.: Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics 34*, 6 (2015), 199. 16

[LLM*07] LINTU A., LENSCH H. P., MAGNOR M. A., EL-ABED S., SEIDEL H.-P.: 3D reconstruction of emission and absorption in planetary nebulae. In *Proceedings of International Workshop on Volume Graphics* (2007), pp. 9–16. 14

[LPW*08] LI Y., PERLMAN E., WAN M., YANG Y., MENEVEAU C., BURNS R., CHEN S., SZALAY A., EYINK G.: A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, 9 (2008), N31. 16

[LS08] LIU T., SHEN L.: Fluid flow and optical flow. *Journal of Fluid Mechanics 614* (2008), 253–291. 15

[MAF*09] MASIA B., AGUSTIN S., FLEMING R. W., SORKINE O., GUTIERREZ D.: Evaluation of reverse tone mapping through varying exposure conditions. *ACM Transactions on Graphics 28*, 5 (2009), 160:1–160:8. 16

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108. 39

[MBK81] MARTINS H., BIRK J. R., KELLEY R. B.: Camera models based on data from two calibration planes. *Computer Graphics and Image Processing 17*, 2 (1981), 173–180. 51

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Symposium on Computer Animation* (2003), pp. 154–159. 12

[MCPN08] MOLEMAKER J., COHEN J. M., PATEL S., NOH J.: Low viscosity flow simulations for animation. In *Symposium on Computer Animation* (2008), Eurographics Association, pp. 9–18. 4, 19, 33, 34, 104, 106, 107

[MK90] MYONG H. K., KASAGI N.: A new approach to the improvement of k-ε turbulence model for wall-bounded shear flows. *JSME International Journal 33*, 1 (1990), 63–72. 16

[MK11] MORRIS N. J., KUTULAKOS K. N.: Dynamic refraction stereo. *Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 8 (2011), 1518–1531. 14

[MLPK13] MEINHARDT-LLOPIS E., PÉREZ J. S., KONDERMANN D.: Horn-Schunck optical flow with a multi-scale strategy. *Image Processing On Line 2013* (2013), 151–172. 15, 59

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Transactions on Graphics 32*, 4 (2013), 104. 19

[MNvTK*16] MORRIS P., NARRACOTT A., VON TENGG-KOBLIGK H., SOTO D., HSIAO S., LUNGU A., ET AL.: Computational fluid dynamics modelling in cardiovascular medicine. *Heart 102*, 1 (2016), 18–28. 6

[MSCL91] MOIN P., SQUIRES K., CABOT W., LEE S.: A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Physics of Fluids A: Fluid Dynamics 3*, 11 (1991), 2746–2757. 6

[MST10] MCADAMS A., SIFAKIS E., TERAN J.: A parallel multigrid Poisson solver for fluids simulation on large grids. In *Symposium on Computer Animation* (2010), pp. 65–74. 12

[MTP*18] MA P., TIAN Y., PAN Z., REN B., MANOCHA D.: Fluid directed rigid body control using deep reinforcement learning. *ACM Transactions on Graphics 37*, 4 (2018), 96. 16

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. In *ACM Transactions on Graphics* (2004), vol. 23, ACM, pp. 449–456. 17

[NB11] NIELSEN M. B., BRIDSON R.: Guide shapes for high resolution naturalistic liquid simulation. In *ACM Transactions on Graphics* (2011), vol. 30, ACM, p. 83. 17

[NC10] NIELSEN M. B., CHRISTENSEN B. B.: Improved variational guiding of smoke animations. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 705–712. 17, 92

[NCZ*09] NIELSEN M. B., CHRISTENSEN B. B., ZAFAR N. B., ROBLE D., MUSETH K.: Guiding of smoke animations through variational coupling of simulations at different resolutions. In *Symposium on Computer Animation* (2009), ACM, pp. 217–226. 17

[New99] NEWTON I.: *The Principia: mathematical principles of natural philosophy*. University of California Press, 1999. 22

[NGCL09] NARAIN R., GOLAS A., CURTIS S., LIN M. C.: Aggregate dynamics for dense crowd simulation. In *ACM Transactions on Graphics* (2009), vol. 28, ACM, p. 122. 18

[NGF02] NGUYEN D., GIBOU F., FEDKIW R.: A fully conservative ghost fluid method and stiff detonation waves. In *International Detonation Symposium* (2002). 17

[NGL10] NARAIN R., GOLAS A., LIN M. C.: Free-flowing granular materials with two-way solid coupling. In *ACM Transactions on Graphics* (2010), vol. 29, ACM, p. 173. 18

[NOB16] NARAIN R., OVERBY M., BROWN G. E.: ADMM $\supseteq$ projective dynamics: fast simulation of general constitutive models. In *Symposium on Computer Animation* (2016), pp. 21–28. 19

[NSCL08] NARAIN R., SEWALL J., CARLSON M., LIN M. C.: Fast animation of turbulence using energy transport and procedural synthesis. In *ACM Transactions on Graphics* (2008), vol. 27, ACM, p. 166. 12

[ODAO15] OKABE M., DOBASHI Y., ANJYO K., ONAI R.: Fluid volume modeling from sparse multi-view images by appearance transfer. *ACM Transactions on Graphics 34*, 4 (2015), 93. 14, 15, 86

[OV14] O'CONNOR D., VANDENBERGHE L.: Primal-dual decomposition by operator splitting and applications to image deblurring. *SIAM Journal on Imaging Sciences 7*, 3 (2014), 1724–1754. 19

[Par12] PARENT R.: *Computer animation: Algorithms and techniques*. Newnes, 2012. 1

[PB*14] PARIKH N., BOYD S., ET AL.: Proximal algorithms. *Foundations and Trends® in Optimization 1*, 3 (2014), 127–239. 26, 29, 58

[PBT19] PRANTL L., BONEV B., THUEREY N.: Generating liquid simulations with deformation-aware neural networks. In *International Conference on Learning Representations* (2019). 16, 82

[PCBC09] POCK T., CREMERS D., BISCHOF H., CHAMBOLLE A.: An algorithm for minimizing the mumford-shah functional. In *Proceedings of IEEE International Conference on Computer Vision* (2009), IEEE, pp. 1133–1140. xxi, 3, 19, 31, 111

[PHT*13] PAN Z., HUANG J., TONG Y., ZHENG C., BAO H.: Interactive localized liquid motion editing. *ACM Transactions on Graphics 32*, 6 (2013), 184. 17

[PJH16] PHARR M., JAKOB W., HUMPHREYS G.: *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016. 39

[PM17] PAN Z., MANOCHA D.: Efficient solver for spacetime control of smoke. *ACM Transactions on Graphics 36*, 5 (2017), 162. 17, 19

[PTC*10] PFAFF T., THUEREY N., COHEN J., TARIQ S., GROSS M.: Scalable fluid simulation using anisotropic turbulence particles. vol. 29, ACM, p. 174. 12

[PTSG09] PFAFF T., THUEREY N., SELLE A., GROSS M.: Synthetic turbulence using artificial boundary layers. In *ACM Transactions on Graphics* (2009), vol. 28, ACM, p. 121. 17

[QZG*19] QU Z., ZHANG X., GAO M., JIANG C., CHEN B.: Efficient and conservative fluids using bidirectional mapping. *ACM Transactions on Graphics 38*, 4 (2019), article 128. 12

[Rad86] RADON J.: On the determination of functions from their integral values along certain manifolds. *IEEE Transactions on Medical Imaging 5*, 4 (1986), 170–176. 4, 14

[Rad13] RADKE R. J.: *Computer vision for visual effects*. Cambridge University Press, 2013. 15

[REN*04] RASMUSSEN N., ENRIGHT D., NGUYEN D., MARINO S., SUMNER N., GEIGER W., HOON S., FEDKIW R.: Directable photorealistic liquids. In *Symposium on Computer Animation* (2004), Eurographics Association, pp. 193–202. 17

[RLL*13] REN B., LI C.-F., LIN M. C., KIM T., HU S.-M.: Flow field modulation. *IEEE Transactions on Visualization and Computer Graphics 19*, 10 (2013), 1708–1719. 17, 98

[SABS14] SETALURI R., AANJANEYA M., BAUER S., SIFAKIS E.: SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics 33*, 6 (2014), 205. 12

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Transactions on Graphics 31*, 4 (2012), 61. 18

[SDKN18] SATO S., DOBASHI Y., KIM T., NISHITA T.: Example-based turbulence style transfer. *ACM Transactions on Graphics 37*, 4 (2018), 84. 6, 113

[SFK*08] SELLE A., FEDKIW R., KIM B., LIU Y., ROSSIGNAC J.: An unconditionally stable MacCormack method. *Journal of Scientific Computing 35*, 2-3 (2008), 350–371. 12, 77

[SO10] SCHLATTER P., ORLU R.: Assessment of direct numerical simulation data of turbulent boundary layers. *Journal of Fluid Mechanics 659* (2010), 116–126. 16

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. In *ACM Transactions on Graphics* (2005), vol. 24, ACM, pp. 910–914. 12

[SS16] SCHNEIDERS J. F., SCARANO F.: Dense velocity reconstruction from tomographic PTV with material derivatives. *Experiments in Fluids 57*, 9 (2016), 139. 13

[SSC*13] STOMAKHIN A., SCHROEDER C., CHAI L., TERAN J., SELLE A.: A material point method for snow simulation. *ACM Transactions on Graphics 32*, 4 (2013), article 102. 12

[SSE17] SCHNEIDERS J. F., SCARANO F., ELSINGA G. E.: Resolving vorticity and dissipation in a turbulent boundary layer by tomographic PTV and VIC+. *Experiments in Fluids 58*, 4 (2017), 27. 13

[Sta99] STAM J.: Stable fluids. In *SIGGRAPH* (1999), vol. 99, pp. 121–128. 2, 11, 24, 77

[Sta01] STARN J.: A simple fluid solver based on the FFT. *Journal of Graphics Tools 6*, 2 (2001), 43–52. 12

[Sut94] SUTER D.: Motion estimation and vector splines. In *Proceedings of IEEE Computer Vision and Pattern Recognition* (1994), vol. 94, pp. 939–942. 15

[SY05a] SHI L., YU Y.: Controllable smoke animation with guiding objects. *ACM Transactions on Graphics 24*, 1 (2005), 140–164. 17

[SY05b] SHI L., YU Y.: Taming liquids for rapidly changing targets. In *Symposium on Computer Animation* (2005), ACM, pp. 229–236. 17

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. In *Computer Graphics Forum* (2003), vol. 22, Wiley Online Library, pp. 391–400. 17

[TGK*17] TAMPUBOLON A. P., GAST T., KLÁR G., FU C., TERAN J., JIANG C., MUSETH K.: Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics 36*, 4 (2017), 105. 12

[Thu17] THUEREY N.: Interpolations of smoke and liquid simulations. *ACM Transactions on Graphics 36*, 1 (2017), 3. 82

[TKPR06] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-preserving fluid control. In *Symposium on Computer Animation* (2006), Eurographics Association, pp. 7–12. 17, 94

[TP19] THUEREY N., PFAFF T.: MantaFlow, 2019. http://mantaflow.com. 28, 77

[TSSP17] TOMPSON J., SCHLACHTER K., SPRECHMANN P., PERLIN K.: Accelerating Eulerian fluid simulation with convolutional networks. In *Proceedings of International Conference on Machine Learning* (2017), vol. 70, JMLR.org, pp. 3424–3433. 16

[UHT17] UM K., HU X., THUEREY N.: Perceptual evaluation of liquid simulation methods. *ACM Transactions on Graphics 36*, 4 (2017), 143. 16, 77

[UHT18] UM K., HU X., THUEREY N.: Liquid splash modeling with neural networks. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 171–182. 16

[WBAW12] WAHLBERG B., BOYD S., ANNERGREN M., WANG Y.: An ADMM algorithm for a class of total variation regularized estimation problems. *IFAC Proceedings Volumes 45*, 16 (2012), 83–88. 31

[WBT19] WIEWEL S., BECHER M., THUEREY N.: Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 71–82. 16

[WC11] WEDEL A., CREMERS D.: Stereoscopic scene flow for 3D motion analysis, 2011. 58

[WCF07] WHITE R., CRANE K., FORSYTH D. A.: Capturing and animating occluded cloth. In *ACM Transactions on Graphics* (2007), vol. 26, ACM, p. 34. 12

[WLM13] WENGER S., LORENZ D., MAGNOR M.: Fast image-based modeling of astronomical nebulae. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 93–100. 14

[WLZ*09] WANG H., LIAO M., ZHANG Q., YANG R., TURK G.: Physically guided liquid surface modeling from videos. *ACM Transactions on Graphics 28*, 3 (2009), 90. 13, 14

[WM93] WEI G.-Q., MA S.: A complete two-plane camera calibration method and experimental comparisons. In *Proceedings of IEEE International Conference on Computer Vision* (1993), IEEE, pp. 439–446. 51

[WP10] WEISSMANN S., PINKALL U.: Filament-based smoke with vortex shedding and variational reconnection. In *ACM Transactions on Graphics* (2010), vol. 29, Citeseer, p. 115. 12

[XFCT18] XIE Y., FRANZ E., CHU M., THUEREY N.: tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Transactions on Graphics 37*, 4 (2018). 16, 113

[XIAP*17] XIONG J., IDOUGHI R., AGUIRRE-PABLO A. A., ALJEDAANI A. B., DUN X., FU Q., THORODDSEN S. T., HEIDRICH W.: Rainbow particle imaging velocimetry for dense 3D fluid velocity imaging. *ACM Transactions on Graphics 36*, 4 (2017), 36. 13, 19, 43

[YCZ11] YUAN Z., CHEN F., ZHAO Y.: Pattern-guided smoke animation with Lagrangian coherent structure. In *ACM Transactions on Graphics* (2011), vol. 30, ACM, p. 136. 17, 98

[YRK18] YAZDANI A., RAISSI M., KARNIADAKIS G.: Hidden fluid mechanics: Navier-Stokes informed deep learning from the passive scalar transport. *Bulletin of the American Physical Society* (2018). 6

[ZAI*18] ZANG G., ALY M., IDOUGHI R., WONKA P., HEIDRICH W.: Super-resolution and sparse view CT reconstruction. In *Proceedings of the European Conference on Computer Vision* (2018), pp. 137–153. 14

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *ACM Transactions on Graphics* (2005), vol. 24, ACM, pp. 965–972. 12

[ZIT*18] ZANG G., IDOUCHI R., TAO R., LUBINEAU G., WONKA P., HEIDRICH W.: Space-time tomography for continuously deforming objects. *ACM Transactions on Graphics 37*, 4 (2018), 100. 14

[ZIT*19] ZANG G., IDOUGHI R., TAO R., LUBINEAU G., WONKA P., HEIDRICH W.: Warp-and-project tomography for rapidly deforming objects. *ACM Transactions on Graphics 38*, 4 (2019). 13, 19

[ZNT18] ZEHNDER J., NARAIN R., THOMASZEWSKI B.: An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics 37*, 4 (2018), 85:1–85:8. 12, 77