

Physical Simulation and Deep Learning using Particle-Based Representations

VON DER FAKULTÄT FÜR INFORMATIK DER RWTH AACHEN UNIVERSITY ZUR ERLANGUNG DES
AKADEMISCHEN GRADES EINES DOKTORS DER NATURWISSENSCHAFTEN GENEHMIGTE
DISSERTATION.

vorgelegt von

Stefan Rhys Jeske, M.Sc. RWTH

Berichter: Prof. Dr. rer. nat. Jan Bender
Prof. Dr.-Ing. Matthias Teschner

Tag der mündlichen Prüfung: 12.08.2026

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek verfügbar.

Acknowledgments

First, I would like to thank Jorja. You listened to my complaints, offered advice, and supported me in more ways than I can express. I am eternally grateful.

Next, I thank my parents, Antje and Claus, who supported me throughout my academic pursuits, both financially and by giving me stability during one of the most chaotic periods of my life. You have always valued knowledge and education, and you have encouraged me to make my own way and find my own solutions.

I also thank my brother Philip. With your outstanding moral values and character, you have been more of a role model to me than you can imagine. I think very fondly of our childhood together and always look forward to spending time with you and your family.

I also thank my friends and colleagues. Theron Guo and Fabian Löschner, whom I met on the first day of university almost 13 years ago. I don't think any of us would have imagined then that we would all end up doing (and finishing) a PhD. We have been through so many late nights studying for seemingly useless courses and spent far too much time gaming to make it this far. I look forward to seeing what life will have in store for us in the future. I especially thank Fabian, with whom I was able to walk much of the academic path together. I will miss being on the other side of the desk from you. I thank José Antonio Fernández-Fernández, with whom I could thoroughly trash-talk many topics, such as academic publishing processes, job prospects, and overall life goals. I look forward to exchanging more audio podcasts. I thank Lukas Westhofen, Timna Böttcher, Tassilo Kugelstadt, Andreas Longva and Marcel Weiler for the productive discussions, paper collaborations, and coffee breaks.

Finally, I would like to thank my supervisor, Prof. Jan Bender, who saw potential in me during my master's degree and offered me the opportunity to pursue a PhD. He taught me a great deal about how to work, write, and deal with critique, and gave me the tools to improve both myself and my work.

Abstract

Points are simple and versatile primitives for representing geometry in computer graphics, with applications including physical simulation and geometric deep learning. Current approaches in these directions are already powerful, but they are still lacking in specific areas. For one, implicit particle-based fluid simulation is an active area of research that has shown improvements for a variety of effects, but, at the time of publication of the relevant work, had not been successfully explored for surface tension. This is an important force for realistic fluid motion. In addition, particle simulation has not been quantifiably established as a viable alternative to more traditional grid-based simulation methods, particularly for complex melt-driven processes. Finally, reconstructing closed surfaces from surface particles is an ill-posed problem without a clear or simple solution. Existing methods, particularly in deep learning, often fail to generate satisfying surfaces using direct inference and may rely on additional optimization passes at test-time to create a better fit. To address these shortcomings, this thesis investigates how the fidelity and efficiency of using particles can be improved in two primary directions.

First, for particle-based fluid simulation with Smoothed Particle Hydrodynamics (SPH), methods are developed to address challenging thermodynamic, melt-driven scenarios. An implicit cohesion-based surface tension model is introduced and strongly coupled with implicit viscosity to improve stability, especially in high-tension regimes. In addition, implicit formulations for melting and solidification, together with accurate particle-to-grid transfer, enable quantitative studies of welding and thermal spraying. These studies show good agreement between the SPH framework and established grid-based approaches while providing efficient high-resolution simulations for free-surface dynamics.

Second, for point-based deep learning, an efficient hybrid point-grid architecture is presented to reconstruct neural distance fields from point-cloud inputs. The method combines point processing and latent grid representations in a single forward-pass encoder-decoder design, enabling accurate and efficient shape reconstruction of closed surfaces. It is shown that this architecture is fast to evaluate and train, while outperforming other baselines on the reconstruction task.

Taken together, the presented works improve point-based methods both in terms of physical accuracy in simulation and efficient geometric representation learning.

Contents

Symbols	vii
Shorthands	viii
1 Introduction	1
1.1 Outline	3
1.2 List of Publications	4
1.3 Contributions	5
 I Particle-based Fluid Simulation	 7
2 Foundations of Smoothed Particle Hydrodynamics	9
2.1 Motivation and Derivation	10
2.2 SPH Algorithm	13
2.3 Summary	21
3 Implicit Surface Tension for SPH Fluid Simulation	23
3.1 Related Work	24
3.2 Method	27
3.3 Results	35
3.4 Conclusion	42
4 Quantitative Evaluation of SPH in TIG Spot Welding	45
4.1 Related Work	46
4.2 Method	48
4.3 Results	56
4.4 Discussion	65
4.5 Conclusion	66
5 Application and Benchmark of SPH for Modeling the Impact in Thermal Spraying	67
5.1 Related Work	68
5.2 Method	70
5.3 Results	76
5.4 Discussion	83
5.5 Conclusion	84
 II Particle-Based Deep Learning	 87
6 Foundations of Deep Learning	89

6.1	Multi-layer Perceptrons	90
6.2	Training	92
6.3	Backpropagation and Stochastic Gradient Descent	92
6.4	Deep Learning on Particles	94
6.5	Encoder-Decoder Networks	97
6.6	Summary	98
7	HYVE: Hybrid Vertex Encoder for Neural Distance Fields	99
7.1	Related Work	101
7.2	Zero-Level-Set Encoder	103
7.3	Results	108
7.4	Conclusion	113
III	Summary	115
8	Concluding Remarks	117
8.1	Summary	117
8.2	Outlook	118
	Bibliography	121
A	Appendix for <i>HYVE: Hybrid Vertex Encoder for Neural Distance Fields</i>	135
A.1	Ablation	135

List of Symbols

Symbols

$\mathbf{x}$	m	position
$\mathbf{v}$	m s^{-1}	velocity
$\mathbf{a}$	m s^{-2}	acceleration
p	N m^{-2}	pressure
μ	N s m^{-2}	dynamic viscosity
ν	$\text{m}^2 \text{s}^{-1}$	kinematic viscosity
$\mathbf{f}_{\text{ext}}$	N m^{-3}	external forces (volumetric)
$\mathbf{g}$	m s^{-2}	gravitational acceleration
t	s	time
m	kg	mass
V	m^3	volume
ρ	kg m^{-3}	density
ρ_0	kg m^{-3}	rest density (of incompressible fluid)
h	m	SPH smoothing length
σ	N m^{-1}	surface tension coefficient
d_{part}	m	particle diameter
r_{part}	m	particle radius
C	s^{-1}	morphological constant
T	K	temperature
c_p	$\text{J kg}^{-1} \text{K}^{-1}$	specific heat capacity
λ	$\text{W m}^{-1} \text{K}^{-1}$	heat conductivity
$\dot{q}'''$	W m^{-3}	volumetric heat source
$\hat{h}$	J kg^{-1}	specific enthalpy
$\mathbf{A}_m$	T m	magnetic vector potential
$\mathbf{J}$	A m^{-2}	current density
$\mathbf{B}$	T	magnetic induction
$\mathbf{F}_L$	N m^{-3}	volumetric Lorentz force
σ_e	S m^{-1}	electrical conductivity
φ	V	scalar electric potential
I_w	A	welding current

Shorthands

f_i	$f(\mathbf{x}_i)$	function value at particle i
f_{ij}	$f(\mathbf{x}_i) - f(\mathbf{x}_j)$	difference between function value at particles i and j
$\bar{f}_{ij}$	$\frac{1}{2} (f(\mathbf{x}_i) + f(\mathbf{x}_j))$	average of function value at particles i and j
W_{ij}	$W(\mathbf{x}_i - \mathbf{x}_j; h)$	SPH kernel evaluation of particle j at particle i
∇f_i	$\nabla_{\mathbf{x}_i} f(\mathbf{x}_i)$	derivative implicitly wrt. particle i
∇W_{ij}	$\nabla_{\mathbf{x}_i} W(\mathbf{x}_i - \mathbf{x}_j; h)$	function derivative implicitly wrt. particle i
$\nabla_i W_{ij}$	$\nabla_{\mathbf{x}_i} W(\mathbf{x}_i - \mathbf{x}_j; h)$	SPH kernel derivative explicitly wrt. particle i
$\nabla_j W_{ij}$	$\nabla_{\mathbf{x}_j} W(\mathbf{x}_i - \mathbf{x}_j; h)$	SPH kernel derivative explicitly wrt. particle j
$\mathcal{N}_{\mathbf{x}}$		neighboring particles of position $\mathbf{x}$
$\mathcal{N}_i$		neighboring particles of particle with index i

Introduction

Motion and movement are ever-present in our daily lives. As children, we quickly learn to understand and interact with our surroundings, as well as how our actions affect materials around us. For example, we learn how to throw and catch a ball, how to swish liquid inside a glass, or how to place our feet so that we do not slip on icy surfaces. While physical interactions just occur all around us, it takes a tremendous amount of research and mathematical modeling to emulate these interactions within virtual environments. In fact, doing so with any reasonable accuracy would massively exceed the entirety of all our compute capabilities combined. Therefore, the goal of physical simulation in computer graphics is largely to capture these interactions with a focus on visual plausibility—think “Does this look natural?”—instead of extreme physical accuracy—“Is this bouncing ball exerting a contact force equal to that of real-world measurements?”

Physical motion is formally described through differential equations that relate the change in an object’s state to the sum of outside influences and any internal changes. Restricting ourselves to motion on everyday scales—consider a coffee maker, water bottle, car, or frying pan—Newton’s laws of motion are sufficient. Furthermore, we will follow the continuum mechanics approach of assuming that all physical quantities are defined at every point in space. Then, in order to be able to run physical simulations on computer hardware, these “infinitely-fine” continuum models have to be discretized, i.e., they have to be made finite as computer algorithms only have finite computational capacity.

In practice, this can be done by using geometric shapes of varying complexity. Typically, points are the basis of any discretization and can be connected to form, e.g., triangles and quadrilaterals, or in 3D, tetrahedra, hexahedra, and more. Points are pivotal to these representations, i.e., the triangle cannot exist without the three distinct points where its edges intersect. In fact, a given shape could be entirely represented by points, without any connectivity, which simplifies the discretization process to simply placing points in space. The result is a fast, flexible, and versatile representation that can nonetheless capture intricate details. It further permits topological changes without having to reconnect parts of the geometry to avoid problems like element inversions and self-intersections. Using only points unfortunately comes at the cost of not defining clear material boundaries as well as connectivity within a material, such as strands or fibers. Nevertheless, as shown in Figure 1.1, points are used as an effective representation in a variety of computer graphics applications, such as 3D scanning, scene reconstruction with Gaussian splatting and, of course, physical simulation (often fluid simulation).

This thesis focuses on Lagrangian point-based discretizations. In this framework, physical space and materials are represented by a collection of unstructured points, commonly known as point clouds. In contrast to the Eulerian perspective, where the simulation frame is fixed in space, the Lagrangian perspective tracks points as they move with the material. This is the simplest possible representation,



Figure 1.1. Showing a variety of use cases for point-based representations in graphics. The left image shows a 3D lidar scan from the KITTI dataset [GEIGER et al. 2013] used in vision tasks for autonomous driving. The middle image shows scene reconstruction with Gaussian splatting¹ and the right image shows a particle-based fluid simulation [BENDER et al. 2023].

natively only containing its own position, even though it may be augmented by orientation and some kind of shape in practice. Particles also carry field quantities, such as mass or temperature (enthalpy), as they move through the domain, naturally conserving these quantities without numerical dissipation.

Within point-based discretizations there exist a variety of open problems, as there are with any methodological approach. In this thesis, we mainly aim to tackle two categories of these problems. First, we aim to *extend and improve the fidelity of point-based simulations* in complex scenarios that require modeling thermodynamic phase-change phenomena found in industrial manufacturing processes. In particular, we will focus on the particle-based simulation method Smoothed Particle Hydrodynamics (SPH). Despite its name implying simulation of fluids (where it has its origin), SPH is used to simulate a large variety of materials and physical effects, as we discussed in a recent state-of-the-art report [HOLZ et al. 2025]. Second, we want to *improve the efficiency of using point-based representations* and mitigate problems due to a lack of surface definition in point clouds in a data-driven way. For this, we will build on geometric deep learning to process point clouds in a permutation and translation invariant manner, while proposing an architecture that aim to maximize information extraction.

To start with, we will narrow down the scope of phase-change processes. In this thesis, we look specifically at tungsten inert gas (TIG) welding [JESKE et al. 2022b] and thermal spraying [JESKE et al. 2022a]. For the simulation of these melt-driven processes, purely grid-based methods (FEM, VOF) are the de facto standard [CADIOU et al. 2020, HERTEL et al. 2014, MOKROV et al. 2021, SEMYONOV 2014]. These simulations can be very costly in many ways. They require precise geometric meshes, which have to be recomputed as the geometry is adjusted. The simulation itself can be very expensive due to a high number of degrees of freedom and may fail to capture topology changes in the geometry without expensive remeshing operations. In addition, they are limited to the domain they define in advance, and often cannot capture anything outside this region of interest. Finally, the region of interest (such as a weld seam) often has to be “baked” into the geometric mesh, which makes application to dynamic processes much more difficult than to quasi-static processes. All of these drawbacks can be addressed by employing a particle-based simulation framework such as SPH, which simulates a number of particles that can freely move in an unbounded domain and can change topology without requiring any additional computation. This leads to the definition of our first questions:

RESEARCH QUESTION 1

How can we model complex melt-driven processes such as TIG welding and thermal spraying using the SPH method?

¹Centrale II viewed using [krpano](#).

RESEARCH QUESTION 2

What degree of quantitative accuracy can we expect the SPH method to achieve in comparison to established grid-based solvers in simulating melt-driven processes?

While the goal in graphics is often visual plausibility, the goal here is to bridge the gap and show that visual plausibility is often intertwined with quantitative accuracy. Simulating these processes accurately further involves high surface tension forces and phase changes. Previous methods simulating surface tension rely on explicit integration, which can struggle with stability in high-stiffness scenarios, limiting their overall applicability [JESKE et al. 2023]. In addition, there are already a number of implicit primal and dual solvers for the SPH framework. They show the benefits of implicit integration both in terms of simulation stability and efficiency, generally because larger time steps can be taken. A logical question is therefore:

RESEARCH QUESTION 3

How can surface tension be formulated implicitly to ensure simulation stability in high-tension scenarios? Can it be strongly coupled with other implicit solvers to further improve efficiency and stability?

Even though existing SPH simulations are effective, there remains a core challenge when representing an object using particles, namely reconstructing its surface. Trying to determine which surface resulted in the current particle sampling is a fundamentally ill-posed problem, i.e., a problem where small changes in the particle configuration could cause a large change in the underlying surface. Implicit surface representations that define the surface as an iso-contour of a volumetric field can be used to mitigate this problem. For example, rigid objects within particle simulations can be represented implicitly as volumetric (signed) distance fields (SDFs), greatly accelerating interactions. However, extending this geometric representation to deformable objects poses a challenge. Implicit representations, such as SDFs, are powerful for interactions of rigid objects, but computing them for dynamically changing geometry is computationally expensive. As deep learning is well suited to constructing priors from data and mapping patterns to specific outputs, we investigate whether it could offer a more efficient mapping from explicit point sets to implicit fields. We consider the final question:

RESEARCH QUESTION 4

How do we derive a deep learning architecture to efficiently construct and update implicit representations of deforming point-based objects?

All of these questions will be addressed in the course of this thesis. The next section will give an outline of the various parts and chapters.

1.1 Outline

This thesis is divided into two main parts, both of which fundamentally use point primitives as a basis. The first part concerns itself with the particle-based simulation method Smoothed Particle Hydrodynamics (SPH). To start, Chapter 2 introduces fundamentals about the SPH method, specifically how it is used to discretize continuum mechanics equations of motion. This includes a description of explicit and implicit fluid solvers. The following chapters will then be based on peer-reviewed publications. Chapter 3 describes an implicit cohesion-based surface tension model that shows improvements with

respect to simulation stability and computational efficiency. Then, it is shown how the SPH method can be used to simulate melt-driven fluid phenomena with an accuracy that is comparable to more established grid-based discretizations. First, for a high-frequency tungsten inert gas (HF-TIG) welding process, in Chapter 4, and second, for the simulation of a microscopic droplet impact that is observed when thermally spraying material coatings, in Chapter 5.

The second part of this thesis concerns deep learning approaches for point-based representations. While deep learning is very well established for grid-based representations, such as images, or sequential data, such as text, it is much harder to generalize these concepts to unstructured collections of points. Therefore, Chapter 6 will first introduce some of the general concepts of point-based deep learning. Subsequently, in Chapter 7, a recently proposed method for efficiently processing point cloud inputs within neural networks (NNs) will be outlined. This method uses points in combination with a background grid for efficient spatial aggregation and proposes consistent point-to-grid transfers that aim to preserve as much information about the points as possible. In the final chapter of this thesis, Chapter 8, the presented topics are summarized and possible future work is discussed.

Please take note that the content of this thesis is largely based on peer-reviewed works as discussed in the next section. These works resulted from a collaborative research effort. As such, the language in Chapter 3, Chapter 4, Chapter 5 and Chapter 7 is written from the perspective of a collective “we”.

1.2 List of Publications

Over the course of my employment as a researcher at the Computer Animation group of RWTH Aachen University I have contributed to a number of scientific articles. These articles are published in peer-reviewed scientific journals and were, in part, presented at international conferences. The following contains a list of publications for which I was the main author:

JESKE, STEFAN RHYS, JONATHAN KLEIN, DOMINIK MICHELS, and JAN BENDER [2026]. “HYVE: Hybrid Vertex Encoder for Neural Distance Fields”. In: *IEEE Transactions on Visualization and Computer Graphics* 32.4, pp. 2962–2973. DOI: [10.1109/TVCG.2026.3658870](https://doi.org/10.1109/TVCG.2026.3658870).

JESKE, STEFAN RHYS, LUKAS WESTHOFEN, FABIAN LÖSCHNER, JOSÉ ANTONIO FERNÁNDEZ-FERNÁNDEZ, and JAN BENDER [2023]. “Implicit Surface Tension for SPH Fluid Simulation”. In: *ACM Trans. Graph.* 43.1, 13:1–13:14. DOI: [10.1145/3631936](https://doi.org/10.1145/3631936).

JESKE, STEFAN RHYS, JAN BENDER, KIRSTEN BOBZIN, HENDRIK HEINEMANN, KEVIN JASUTYN, MAREK SIMON, OLEG MOKROV, RAHUL SHARMA, and UWE REISGEN [2022a]. “Application and Benchmark of SPH for Modeling the Impact in Thermal Spraying”. In: *Computational Particle Mechanics* 9.6, pp. 1137–1152. DOI: [10.1007/s40571-022-00459-9](https://doi.org/10.1007/s40571-022-00459-9).

JESKE, STEFAN RHYS, MAREK SEBASTIAN SIMON, OLEKSII SEMENOV, JAN KRUSKA, OLEG MOKROV, RAHUL SHARMA, UWE REISGEN, and JAN BENDER [2022b]. “Quantitative Evaluation of SPH in TIG Spot Welding”. In: *Computational Particle Mechanics* 10.1, pp. 1–18. DOI: [10.1007/s40571-022-00465-x](https://doi.org/10.1007/s40571-022-00465-x).

In addition, I have made contributions to a number of other peer-reviewed works:

BÖTTCHER, TIMNA, LUKAS WESTHOFEN, **STEFAN RHYS JESKE**, and JAN BENDER [2025]. “Implicit Incompressible Porous Flow Using SPH”. In: *ACM Trans. Graph.* 44.6, 268:1–268:13. DOI: [10.1145/3763325](https://doi.org/10.1145/3763325).

HOLZ, DANIEL, **STEFAN RHYS JESKE**, FABIAN LÖSCHNER, JAN BENDER, YIN YANG, and SHELDON ANDREWS [2025]. “Multiphysics Simulation Methods in Computer Graphics”. In: *Computer Graphics Forum* 44.2, e70082. DOI: [10.1111/cgf.70082](https://doi.org/10.1111/cgf.70082).

LÖSCHNER, F., J. A. FERNÁNDEZ-FERNÁNDEZ, **S. R. JESKE**, and J. BENDER [2024]. “Curved Three-Director Cosserat Shells with Strong Coupling”. In: *Computer Graphics Forum*, e15183. DOI: [10.1111/cgf.15183](https://doi.org/10.1111/cgf.15183).

- MOKROV, O., S. WARKENTIN, L. WESTHOFEN, **S. R. JESKE**, J. BENDER, R. SHARMA, and U. REISGEN [2024]. “Simulation of Wire Metal Transfer in the Cold Metal Transfer (CMT) Variant of Gas Metal Arc Welding Using the Smoothed Particle Hydrodynamics (SPH) Approach”. In: *Materialwissenschaft und Werkstofftechnik* 55.1, pp. 62–71. DOI: [10.1002/mawe.202300166](https://doi.org/10.1002/mawe.202300166).
- WESTHOFEN, L., J. A. FERNÁNDEZ-FERNÁNDEZ, **S. R. JESKE**, and J. BENDER [2024]. “Strongly Coupled Simulation of Magnetic Rigid Bodies”. In: *Computer Graphics Forum*, e15185. DOI: [10.1111/cgf.15185](https://doi.org/10.1111/cgf.15185).
- BENDER, JAN, LUKAS WESTHOFEN, and **STEFAN RHYS JESKE** [2023]. “Consistent SPH Rigid-Fluid Coupling”. In: *Vision, Modeling, and Visualization*, pp. 209–217. DOI: [10.2312/VMV.20231244](https://doi.org/10.2312/VMV.20231244).
- BOBZIN, K., H. HEINEMANN, K. JASUTYN, **S. R. JESKE**, J. BENDER, S. WARKENTIN, O. MOKROV, R. SHARMA, and U. REISGEN [2023]. “Modeling the Droplet Impact on the Substrate with Surface Preparation in Thermal Spraying with SPH”. In: *Journal of Thermal Spray Technology* 32.2, pp. 599–608. DOI: [10.1007/s11666-023-01534-0](https://doi.org/10.1007/s11666-023-01534-0).
- LÖSCHNER, FABIAN, TIMNA BÖTTCHER, **STEFAN RHYS JESKE**, and JAN BENDER [2023a]. “Weighted Laplacian Smoothing for Surface Reconstruction of Particle-based Fluids”. In: 9 pages. DOI: [10.2312/VMV.20231245](https://doi.org/10.2312/VMV.20231245).
- LÖSCHNER, FABIAN, JOSÉ ANTONIO FERNÁNDEZ-FERNÁNDEZ, **STEFAN RHYS JESKE**, ANDREAS LONGVA, and JAN BENDER [2023b]. “Micropolar Elasticity in Physically-Based Animation”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.3, 46:1–46:24. DOI: [10.1145/3606922](https://doi.org/10.1145/3606922).
- WESTHOFEN, LUKAS, **STEFAN RHYS JESKE**, and JAN BENDER [2023]. “A Comparison of Linear Consistent Correction Methods for First-Order SPH Derivatives”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.3, 48:1–48:20. DOI: [10.1145/3606933](https://doi.org/10.1145/3606933).
- FERNÁNDEZ-FERNÁNDEZ, JOSÉ ANTONIO, LUKAS WESTHOFEN, FABIAN LÖSCHNER, **STEFAN RHYS JESKE**, ANDREAS LONGVA, and JAN BENDER [2022]. “Fast Octree Neighborhood Search for SPH Simulations”. In: *ACM Transactions on Graphics* 41.6, 242:1–242:13. DOI: [10.1145/3550454.3555523](https://doi.org/10.1145/3550454.3555523).
- KUGELSTADT, TASSILO, JAN BENDER, JOSÉ ANTONIO FERNÁNDEZ-FERNÁNDEZ, **STEFAN RHYS JESKE**, FABIAN LÖSCHNER, and ANDREAS LONGVA [2021]. “Fast Corotated Elastic SPH Solids with Implicit Zero-Energy Mode Control”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4.3, 33:1–33:21. DOI: [10.1145/3480142](https://doi.org/10.1145/3480142).
- LÖSCHNER, FABIAN, ANDREAS LONGVA, **STEFAN RHYS JESKE**, TASSILO KUGELSTADT, and JAN BENDER [2020]. “Higher-Order Time Integration for Deformable Solids”. In: *Computer Graphics Forum* 39.8, pp. 157–169. DOI: [10.1111/cgf.14110](https://doi.org/10.1111/cgf.14110).

1.3 Contributions

The content of this thesis is mainly based on the four first-authored works listed above. The following will outline my contributions to these works, all of which were published using “RWTH Aachen University” affiliation according to §5 Article (4) of the doctoral degree regulations of Faculty 9 at RWTH Aachen University. For all first-authored publications, I personally researched related work, implemented the presented methods, performed comparisons with competing methods, and wrote most of the manuscript. Unless stated otherwise, co-authors assisted with optimization of the method, running experiments, rendering and visualization of results and feedback on the written manuscript. The following gives a brief outline of the work each paper entailed in chronological order.

- *Quantitative evaluation of SPH in TIG spot welding* [JESKE et al. 2022b]: The main motivation for this work was to establish a baseline for the quantitative accuracy of the Smoothed Particle Hydrodynamics method in welding simulations. To achieve this, existing SPH models were extended to account for heat transfer and phase change from solid to fluid and vice-versa by

means of a momentum sink. This momentum sink was implicitly integrated to guarantee stability. An auxiliary method was developed to transfer SPH quantities to a regular grid by solving a least-squares system for the coefficients of linear basis functions, which aimed to accurately capture physical quantities on the particles. Ultimately, the work was able to show the agreement between the SPH solution and that of the commercial solver COMSOL, while also validating both results with an additional custom FEM solver provided by one of the co-authors.

- *Application and benchmark of SPH for modeling the impact in thermal spraying* [JESKE et al. 2022a]: This work was motivated similarly to the work above, but instead of welding, the main focus was to evaluate the accuracy and efficiency of the SPH method in thermal spraying simulations. In particular, this concerned droplet impact and solidification at very small spatial scales. The melting and solidification model of the method above was used. However, as surface tension plays a much more important role in thermal spraying due to the small spatial scale and rapid solidification process, an explicit curvature-based surface tension model was used and augmented by an implicit cohesion model. The SPH implementation was compared with a preexisting solution using the Volume of Fluid method, showing remarkable agreement. Our results also aligned well with known analytical solutions. Overall, using SPH resulted in net performance gains for these kinds of thermal spraying simulations.

This work was jointly first-authored together with Kevin Jasutyn. This decision was made, as I was solely responsible for developing and implementing the simulation method, while Kevin Jasutyn conducted most of the experiments independently and provided me with valuable context about the simulation processes. This is the first time this work has been used in a dissertation to the best of my knowledge.

- *Implicit Surface Tension for SPH Fluid Simulation* [JESKE et al. 2023]: From the previous two works a lot of insight was gained into how to develop an implicit SPH model. The experience was used to derive and develop a novel fully implicit cohesion-based surface tension model. This was shown to outperform other state-of-the-art methods for computing surface tension in SPH both in terms of performance as well as stability, by being able to stably simulate very large surface tension coefficients. The work was published in the ACM Transactions on Graphics (TOG) journal and presented at the SIGGRAPH '24 conference in Denver, USA.
- *HYVE: Hybrid Vertex Encoder for Neural Distance Fields* [JESKE et al. 2026]: Finally, starting from a long-running cooperative project, a method was developed to quickly compute the signed distance field of arbitrary shapes using neural networks. The method was first intended to use tetrahedral and triangle meshes as an input and to use a learning-based approach. However, as robust analytical solutions already exist for this case, the project pivoted to using (oriented) point clouds, since the solution for these is less well established and can benefit from geometric priors, a task for which deep learning is well suited. The hybrid architecture of this work was motivated by extracting as much detail as possible from the positional information of the input point cloud while using efficient data processing on 3D voxel grids to propagate information across spatial scales. It was shown that the proposed method outperforms other shape encoding methods on the shape reconstruction task. The work was published as a journal paper in IEEE Transactions on Visualization and Computer Graphics.

Part I

Particle-based Fluid Simulation

Foundations of Smoothed Particle Hydrodynamics

Continuum mechanics is a well-researched field that deals with modeling real-life materials. As the name implies, it models the behavior of physical matter as a continuum, in other words, assuming that physical values are defined at every point in continuous space. These models are also known as macroscale models, since their continuum assumption typically only holds when considering materials at “observable” scales, i.e., visible with the naked eye. However, it is known that the continuum assumption does not hold at smaller scales, since, to our current understanding, there exist discrete structures at the smallest level of matter, e.g., atoms. At smaller scales, this is (partially) accounted for by other models, like the meso-scale Lattice-Boltzmann Method [W. LI et al. 2020, LYU et al. 2021] or the micro-scale Molecular Dynamics model [ADCOCK and MCCAMMON 2006]. For this thesis, we exclusively use macro-scale models for which the continuum assumption holds. The following will give a very brief introduction to continuum mechanics, focusing on fluid mechanics in particular. For more information on the topic, the reader is referred to the works of BATCHELOR [2010], CHORIN and MARSDEN [1990], and LANDAU and LIFSHITZ [2013].

There are a variety of proposed models for continuous materials, but all follow the conservation of momentum as described by the well-studied Cauchy momentum equation

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f}, \quad (2.1)$$

which relates the change in momentum to the divergence of the stress tensor $\boldsymbol{\sigma}$ and the external forces $\mathbf{f}$. The density is denoted by ρ . Continuum models then describe the internal forces of a given material using the stress tensor $\boldsymbol{\sigma}$. There is a large variety of well-studied materials, but for this thesis we will focus on incompressible viscous fluids. For this kind of material, the stress tensor can be derived to be

$$\boldsymbol{\sigma} = -p\mathbf{I} + \mu (\nabla \mathbf{v} + (\nabla \mathbf{v})^T), \quad (2.2)$$

where p represents an isotropic stress, called pressure, μ is the viscosity coefficient, $\mathbf{v}$ is the velocity and $\mathbf{I} \in \mathbb{R}^{3 \times 3}$ the identity matrix. Inserting the stress tensor, Equation (2.2), into the Cauchy momentum equation, Equation (2.1), will yield an equation describing the motion of incompressible viscous fluids. The resulting equation is the so-called Navier–Stokes equation. Writing it out in full form yields

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{v} + \frac{\mathbf{f}_{\text{ext}}}{\rho}, \quad (2.3)$$

where the first term on the right-hand side denotes the pressure force which counteracts compression. The second term denotes the viscosity force which acts as a kind of internal friction and $\mathbf{f}_{\text{ext}}$ denotes

external volumetric forces (force per unit volume). Note that the units of all terms are those of accelerations (force per unit mass). This equation is typically augmented by the continuity equation, which enforces the conservation of mass. In the Lagrangian frame, this follows directly from the fact that the mass $m = \rho V$ of an infinitesimal material element is constant,

$$\frac{Dm}{Dt} = \frac{D\rho}{Dt}V + \rho \frac{DV}{Dt} = 0 \iff \frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{v} = 0, \quad (2.4)$$

where $\frac{D}{Dt}$ denotes the material derivative. Since the element is transported with the material, its boundary moves with the local velocity, such that its volume changes at the rate $\frac{DV}{Dt} = \oint \mathbf{v} \cdot \mathbf{n} dS = V \nabla \cdot \mathbf{v}$, i.e., the velocity divergence measures the local rate of volumetric expansion [BATCHELOR 2010]. Expanding the material derivative as $\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \rho$ recovers the equivalent Eulerian form $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$. For incompressible fluids, the density of each material element additionally remains constant along its trajectory,

$$\frac{D\rho}{Dt} = 0. \quad (2.5)$$

Note that this does not require the density to be spatially uniform, only that it is not changed by the motion of the material. Inserting Equation (2.5) into Equation (2.4) and using $\rho > 0$ directly yields

$$\rho \nabla \cdot \mathbf{v} = 0 \iff \nabla \cdot \mathbf{v} = 0, \quad (2.6)$$

which results in the condition that the velocity field must have a divergence of zero. Note that this condition was also used to simplify Equation (2.3) when inserting the stress tensor given by Equation (2.2). More commonly, this is called a “divergence-free” velocity field, which can be thought of as the reduction of the continuity equation, Equation (2.4), for incompressible fluids, Equation (2.5).

As previously mentioned, the work in this thesis deals mainly with fluid simulations using particle-based representations. In other words, we use particles to represent discrete “chunks” of matter. This process is typically referred to as “discretization” and is necessary for us to be able to solve continuum models on computer hardware with finite resources. A common method in computer graphics using a particle-based representation is called Smoothed Particle Hydrodynamics, which is frequently used for the simulation of incompressible fluids. While SPH can also discretize other continuum models, its use in this thesis is mainly for fluid simulation. Generally speaking, the representation using points allows for a very versatile and flexible simulation method that is able to simulate free surface flows very well. Furthermore, attaching a finite mass to each particle perfectly conserves mass by construction. This type of discretization is classified as Lagrangian, where individual material points represent actual material and move as the material moves. It stands in contrast to so-called Eulerian types of discretization, in which discretization points are fixed in space and “observe” as the actual material moves through space. Figure 2.1 illustrates the conceptual difference between Eulerian and Lagrangian viewpoints. Both yield valid results and are used in practice for continuum mechanics simulations in computer graphics. They can even be used in combination in so-called “hybrid” discretizations.

This chapter will first cover the fundamentals of the Smoothed Particle Hydrodynamics method. Afterwards, the method itself will be derived and the discretization of differential operators shown. Using these tools we will then investigate how to construct an explicit simulator. Then we will discuss how to derive an implicit closure for the pressure forces and how to use Chorin splitting to enable implicit time integration, which improves stability in state-of-the-art simulation methods. Finally, the chapter will end with a brief summary.

2.1 Motivation and Derivation

Smoothed Particle Hydrodynamics is a particle-based Lagrangian discretization method that is used to discretize partial differential equations (PDEs). Initially, it was conceived for the simulation of

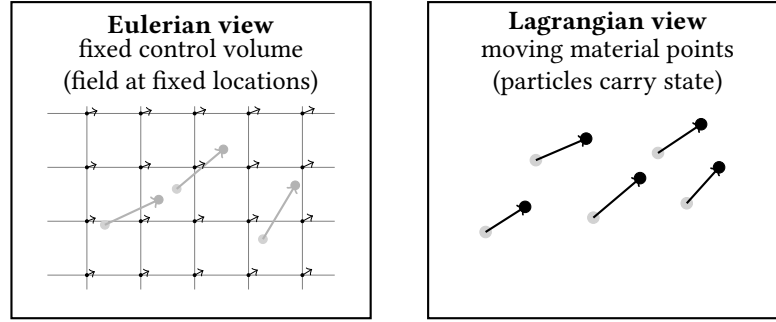


Figure 2.1. Comparison between the two perspectives commonly used in fluid simulation. In the Eulerian view, quantities are evaluated at fixed spatial locations, i.e., the material moves but the evaluation points remain fixed. In the Lagrangian view, material points are tracked as they move through space.

astrophysical phenomena [DESBRUN and GASCUEL 1996, GINGOLD and MONAGHAN 1977, LUCY 1977, MONAGHAN 1992]. In computer graphics, it is mainly used for the simulation of incompressible fluids by discretizing the Navier–Stokes equations, and it was adopted in this field in the early 2000s [M. MÜLLER et al. 2003]. To this day, SPH fluid simulation is an established method in a wide variety of fields [FARROKHPANAH et al. 2021, JESKE et al. 2022b, KOMEN et al. 2020]. Recent advances in graphics include the simulation of snow [GISSLER et al. 2020], the simulation of deformable objects [KUGELSTADT et al. 2021a, PEER et al. 2018], and the strong coupling between fluids and rigid bodies [GISSLER et al. 2019]. In the following, we derive how to construct differential operators on an unstructured collection of points using SPH. The derivations will mostly follow the works by IHMSEN et al. [2014b], KOSCHIER et al. [2019, 2022], M. B. LIU and G. R. LIU [2010], and PRICE [2012].

Typical derivations start with the integral identity using the δ distribution

$$f(\mathbf{x}^*) = (f * \delta)(\mathbf{x}^*) = \int f(\mathbf{x})\delta(\mathbf{x}^* - \mathbf{x})d\mathbf{x}, \quad (2.7)$$

which states that the convolution of function f with the delta distribution δ is simply equal to the evaluation of function f . Since the δ distribution, shown conceptually in Figure 2.2, is not possible to be discretized or computed in practice, it is approximated using a smoothed variant. One such function that approaches the δ distribution in the limit is the Gaussian bell function. However, the Gaussian is not an ideal choice for computational efficiency since its value is non-zero everywhere and only decays towards zero for values approaching infinity. More generally, the δ distribution is approximated using a kernel function W , such that the integral from above reads

$$f(\mathbf{x}^*) \approx (f * W)(\mathbf{x}^*) = \int f(\mathbf{x})W(\mathbf{x}^* - \mathbf{x}; h)d\mathbf{x}, \quad (2.8)$$

where the kernel function W is additionally parametrized by a so-called smoothing length h . As the name implies, increasing h increases the amount of “smoothing” introduced by the approximation of using W , by taking into account a larger area of influence surrounding evaluation point $\mathbf{x}^*$.

In order to be a sufficient approximation of the δ identity in Equation (2.7), the kernel function needs to fulfill the following properties:

$$\text{normalization condition} \quad \int W(\mathbf{x}; h) = 1, \quad (2.9)$$

$$\text{delta condition} \quad \lim_{h \rightarrow 0} W(\mathbf{x}; h) = \delta(\mathbf{x}), \quad (2.10)$$

$$\text{symmetry condition} \quad W(\mathbf{x}; h) = W(-\mathbf{x}; h), \quad (2.11)$$

$$\text{positivity condition} \quad W(\mathbf{x}; h) \geq 0, \quad (2.12)$$

$$\text{compact-support condition} \quad W(\mathbf{x}; h) = 0 \text{ for } \|\mathbf{x}\| > r, \quad (2.13)$$

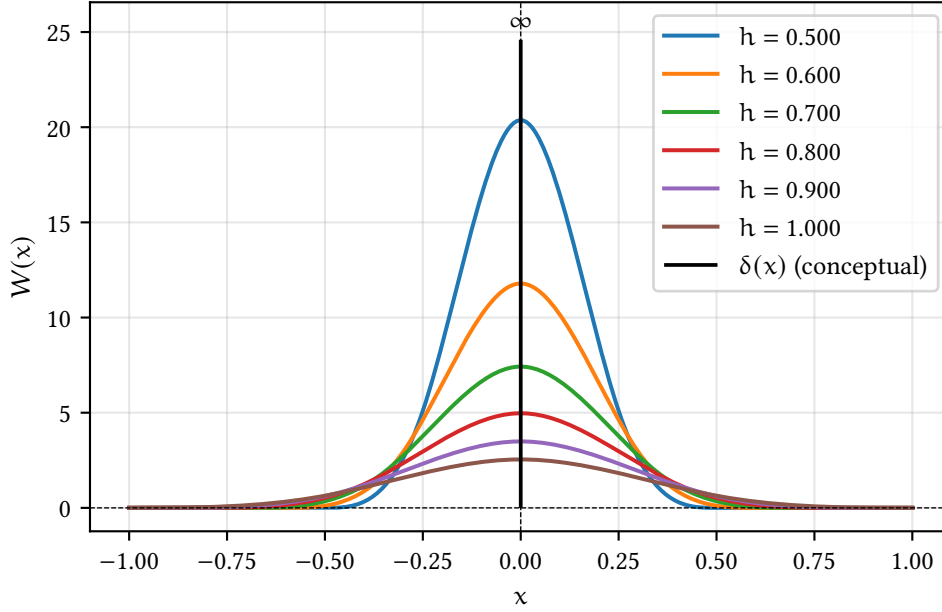


Figure 2.2. The cubic spline kernel as commonly used for SPH simulations in computer graphics. The smoothing length h is varied, which changes the width of the function and the compact support (non-zero region, Equation (2.13)).

where r denotes the compact support radius of kernel function W .

For Equation (2.8) to be a valid approximation of Equation (2.7), only the first two conditions *have* to be satisfied, i.e., Equation (2.9) and Equation (2.10). In other words, the kernel function has to be normalized (much like the δ distribution) and has to converge to the δ distribution as the smoothing length h approaches zero. The other conditions become important in the context of physical simulation. First, the symmetry condition, Equation (2.11), ensures that there is no bias regarding specific particle arrangements. Second, the positivity condition, Equation (2.12), ensures that estimations of physical quantities cannot result in negative values when dealing with incomplete integration domains in practice. This is particularly important when computing things such as mass, volume and density, where negative values can be particularly problematic (and unphysical). Finally, the compact-support condition, Equation (2.13), ensures that only a small subset of points contributes to integral approximations at a certain point. This avoids quadratic algorithmic complexity in the number of points in the simulation and allows SPH simulations to scale to large numbers of points. One such function that fulfills all of these conditions is the so-called cubic spline kernel which is shown in Figure 2.2. For the definition of this function, the reader is referred to the works of KOSCHIER et al. [2019] and MONAGHAN [1992].

Discretization Using a kernel function W that fulfills all of these conditions, we can discretize the integral in Equation (2.8) on a set of particles, which results in

SPH SCALAR APPROXIMATION

$$\begin{aligned} f(\mathbf{x}^*) &\approx \int f(\mathbf{x})W(\mathbf{x}^* - \mathbf{x}; h)d\mathbf{x}, \\ &\approx \sum_{j \in \mathcal{N}(\mathbf{x}^*)} V(\mathbf{x}_j)f(\mathbf{x}_j)W(\mathbf{x}^* - \mathbf{x}_j; h), \end{aligned} \quad (2.14)$$

where $j \in \mathcal{N}(\mathbf{x}^*)$ denotes all particles whose position $\mathbf{x}_j$ is located within the compact support of kernel W centered around position $\mathbf{x}^*$. Further, $V(\mathbf{x}_j)$ denotes the volume associated with particle j and $f(\mathbf{x}_j)$

the function value f at the location of particle j . Implicitly this equation assumes that the function value f is constant within the volume of particle j . Equation (2.14) is generally known as the main equation for SPH discretizations. It can trivially be extended to vectorial or matrix quantities, and allows quantities defined on a set of unordered particles to be evaluated at arbitrary positions in space. This evaluation, however, also assumes that the region of interest is evenly covered with particles, in which case SPH is first-order consistent [WESTHOFEN et al. 2023]. If there is a deficiency of particles, then the approximation accuracy will degrade. Nevertheless, this is rarely a problem in practice with countless state-of-the-art publications utilizing the SPH method for simulating a variety of different materials. It is also featured prominently as one of the main methods enabling multiphysical simulation in computer graphics [HOLZ et al. 2025].

Derivatives In order to use SPH to simulate physical phenomena, e.g., incompressible fluids, the Navier-Stokes equations which were introduced in Equation (2.3) have to be discretized. To that end, we need to be able to compute discrete derivatives, since Equation (2.14) only discretizes scalar (and vector) quantities. However, it can be trivially modified to compute (spatial) derivatives with respect to the current position by moving the derivative operator to the kernel function, i.e.,

SPH SCALAR DERIVATIVE

$$\begin{aligned}\nabla_{\mathbf{x}^*} f(\mathbf{x}^*) &\approx \nabla_{\mathbf{x}^*} \int f(\mathbf{x}) W(\mathbf{x}^* - \mathbf{x}; h) d\mathbf{x}, \\ &= \int f(\mathbf{x}) \nabla_{\mathbf{x}^*} W(\mathbf{x}^* - \mathbf{x}; h) d\mathbf{x}, \\ &\approx \sum_{j \in \mathcal{N}(\mathbf{x}^*)} V(\mathbf{x}_j) f(\mathbf{x}_j) \nabla_{\mathbf{x}^*} W(\mathbf{x}^* - \mathbf{x}_j; h).\end{aligned}\tag{2.15}$$

The form above is for scalar quantities, but it can also be formulated for vector derivatives as follows

$$\begin{aligned}\nabla_{\mathbf{x}^*} f(\mathbf{x}^*) &\approx \sum_{j \in \mathcal{N}(\mathbf{x}^*)} V(\mathbf{x}_j) f(\mathbf{x}_j) \nabla_{\mathbf{x}^*} W(\mathbf{x}^* - \mathbf{x}_j; h)^T, \\ \nabla_{\mathbf{x}^*} \times \mathbf{f}(\mathbf{x}^*) &\approx - \sum_{j \in \mathcal{N}(\mathbf{x}^*)} V(\mathbf{x}_j) f(\mathbf{x}_j) \times \nabla_{\mathbf{x}^*} W(\mathbf{x}^* - \mathbf{x}_j; h), \\ \nabla_{\mathbf{x}^*} \cdot \mathbf{f}(\mathbf{x}^*) &\approx \sum_{j \in \mathcal{N}(\mathbf{x}^*)} V(\mathbf{x}_j) f(\mathbf{x}_j) \cdot \nabla_{\mathbf{x}^*} W(\mathbf{x}^* - \mathbf{x}_j; h),\end{aligned}\tag{2.16}$$

where the negation in the cross product stems from its anti-commutative nature. Higher-order derivatives work analogously to the derivatives stated above, although a “trick” is commonly used to avoid computation of second derivatives of the kernel function W . For example, the following form for the Laplacian, as presented by BROOKSHAW [1985], can be used

$$\nabla^2 f(\mathbf{x}^*) \approx - \sum_{j \in \mathcal{N}(\mathbf{x}^*)} V(\mathbf{x}_j) [f(\mathbf{x}^*) - f(\mathbf{x}_j)] \frac{2 \|\nabla_{\mathbf{x}^*} W(\mathbf{x}^* - \mathbf{x}_j; h)\|}{\|\mathbf{x}^* - \mathbf{x}_j\|}.\tag{2.17}$$

This form of the Laplacian generally results in more stable simulations as opposed to using the second-order derivative of the cubic-spline kernel function. While using a kernel function of higher polynomial degree could also be an option, this has not been well-established in the SPH literature in computer graphics.

2.2 SPH Algorithm

Now that we have established how to discretize differential operators using the SPH framework, we can begin to construct a simple algorithm to simulate the motion of incompressible fluids by applying them

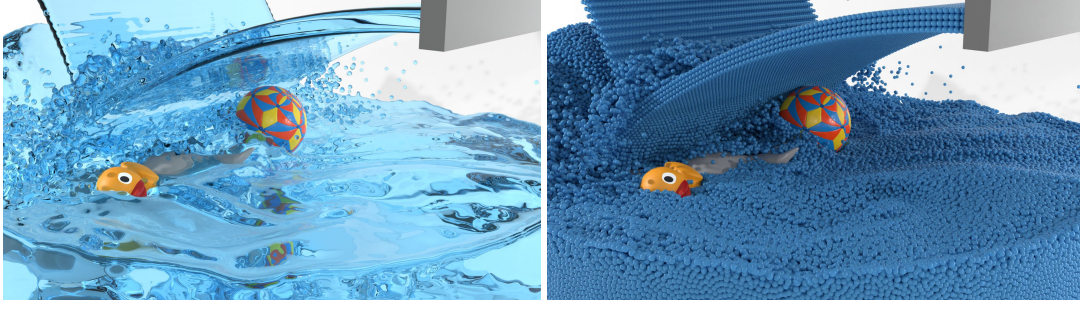


Figure 2.3. *Smoothed Particle Hydrodynamics discretization of continuous media into particles. The image on the left shows the continuous surface, while the image on the right shows the actual discretization as particles [LÖSCHNER et al. 2023].*

to the Navier-Stokes equations in Equation (2.3). To do so, we use the method of lines, which discretizes the spatial and time dimensions separately. This is common practice in almost all numerical PDE solvers, with the exception being methods using, e.g., space-time finite elements. For the spatial domain, i.e., for pressure, viscosity, and external forces, we use the SPH framework. This works by first discretizing our continuous domain $\Omega \subset \mathbb{R}^3$ using a finite set of particles with positions $\mathbf{x}_i \in \mathbb{R}^3$, $i \in [1, \dots, N]$ as seen in Figure 2.3. To use the SPH equations from the previous sections in practice, we will use a shorthand notation in which

$$f_i = f(\mathbf{x}_i) \quad (2.18)$$

denotes the evaluation of scalar quantity f at the position of particle $\mathbf{x}_i$. Furthermore, we will use double indices to denote differences between function evaluations at $\mathbf{x}_i$ and $\mathbf{x}_j$

$$f_{ij} = f(\mathbf{x}_i) - f(\mathbf{x}_j), \quad (2.19)$$

and the special case of

$$W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j; h), \quad (2.20)$$

for the evaluation of the kernel function using particles i and j . For the time domain we use ODE solution methods, of which forward Euler or backward Euler are most commonly used in SPH. To keep things simple, we will consider an explicitly integrated SPH algorithm that directly computes pressure forces using an equation of state (EOS) from density values. Furthermore, an explicit computation of viscosity will be used. While this is not the typically used state of the art in SPH simulation, it can be configured to yield acceptable simulation results and is much more illustrative of the SPH algorithm. The following will describe the individual methods for computing the pressure and viscosity terms of the Navier-Stokes equations. They are repeated here for clarity

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{v} + \mathbf{g},$$

where $\frac{f_{\text{ext}}}{\rho}$ has been replaced with the constant gravitational acceleration $\mathbf{g}$.

2.2.1 Explicit

The Navier-Stokes equations constitute three equations (in 3D), but contain four unknown variables, namely velocity (3 components) and pressure (1 component). Therefore, we need an additional equation to fully solve the PDE. We recall that incompressible fluid simulation also imposes an additional equation, called the continuity equation, Equation (2.4), which for incompressible fluids reduces to the requirement that the velocity field be “divergence-free”, Equation (2.6). It is common practice to define a closure for the unknown pressure by requiring that the pressure forces should act in a way

that counteracts any non-zero divergence that occurs in the fluid velocity field. In other words, the pressure values should counteract compression and expansion of the fluid and ensure that the density remains constant. Since velocity divergence and fluid density can be converted using the continuity equation, rewritten here in Lagrangian form

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v}, \quad (2.21)$$

it can be more suitable to directly use density deviations to prevent density drift rather than using velocity divergence. Note that although Equation (2.5) requires $\frac{D\rho}{Dt} = 0$ for a continuous incompressible fluid, the discrete density obtained from Equation (2.25) generally deviates from the rest density. The residual $\frac{D\rho}{Dt}$ therefore serves as a measure of the violation of the incompressibility constraint that the pressure forces have to remove. There is a large number of ways to derive such a relationship. A straightforward way is to define a so-called equation of state (EOS) that directly relates pressure values to the deviation from the fluid rest density ρ_0 . Examples of such equations are

$$p_i = \kappa \left(\frac{\rho_i}{\rho_0} - 1 \right), \quad (2.22)$$

$$p_i = \kappa (\rho_i - \rho_0), \quad (2.23)$$

$$p_i = \kappa \left(\left(\frac{\rho_i}{\rho_0} \right)^\gamma - 1 \right). \quad (2.24)$$

Here κ and γ denote constants that can be tuned to obtain the desired material behavior and ρ_i denotes the current density estimated by SPH. For a specific particle i at location $\mathbf{x}_i$ this is computed as

$$\rho_i \approx \sum_{j \in \mathcal{N}_i} m_j W_{ij}, \quad (2.25)$$

where we use the shorthands introduced in Equation (2.18) and Equation (2.20). The pressure force can be computed by directly evaluating the gradient of the pressure values obtained from the chosen EOS, e.g., Equation (2.22) by using

$$\nabla p_i \approx \rho_i \sum_{j \in \mathcal{N}_i} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla_i W_{ij}. \quad (2.26)$$

The astute observer will notice that this is not the general-purpose form of the gradient presented in Equation (2.15). Instead, it is another “custom” approximation which results in symmetric force contributions between all pairs of particles. The derivation of this form is presented by PRICE [2012].

The viscosity force can be directly computed using a specialized second-order derivative formulation

$$\mu \nabla^2 \mathbf{v}_i \approx \mu 2(d+2) \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} \frac{\mathbf{v}_{ij} \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \nabla W_{ij}, \quad (2.27)$$

where d is the number of spatial dimensions. This formulation avoids direct computation of the second derivative of the kernel W and additionally conserves linear and angular momentum.

Using Equation (2.25), Equation (2.26) and Equation (2.27) we can now construct a full SPH algorithm under the condition that we can determine the particle neighborhoods $\mathcal{N}_i$, $i \in [1, \dots, N]$. Since this is trivial to do in $\mathcal{O}(N^2)$ by looping over all other particles for each particle, we instead refer to KOSCHIER et al. [2019] for an introduction to the more efficient compact hashing algorithm. The only missing part is integrating the time derivative, for which the symplectic Euler scheme is frequently used

$$\begin{aligned} \mathbf{v}_i^{n+1} &= \mathbf{v}_i^n + \Delta t \mathbf{a}_i^n, \\ \mathbf{x}_i^{n+1} &= \mathbf{x}_i^n + \Delta t \mathbf{v}_i^{n+1}. \end{aligned} \quad (2.28)$$

Algorithm 1 Explicit SPH

```

1: procedure FLUIDSOLVE
2:   for all particles  $i$  do
3:      $\rho_i \leftarrow \text{COMPUTEDENSITY}(i)$  ▷ Equation (2.25)
4:      $p_i \leftarrow \text{COMPUTE PRESSUREVALUE}(i)$  ▷ Equation (2.22)
5:   end for
6:   for all particles  $i$  do
7:      $\mathbf{a}_i \leftarrow \mathbf{a}_i - \frac{1}{\rho_i} \text{COMPUTE PRESSUREFORCE}(i, \rho_i)$  ▷ Equation (2.26)
8:   end for
9:   for all particles  $i$  do
10:     $\mathbf{a}_i \leftarrow \mathbf{a}_i + \frac{1}{\rho_i} \text{COMPUTE VISCOSITY}(i)$  ▷ Equation (2.27)
11:  end for
12:  for all particles  $i$  do
13:     $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t (\mathbf{a}_i + \mathbf{g})$ 
14:     $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$ 
15:  end for
16: end procedure

```

Here $\mathbf{a}$ denotes the acceleration, $\mathbf{v}$ the velocity and $\mathbf{x}$ the positions of each particle i . In addition, Δt denotes the discrete time step size and it should be noted that the newly “updated” velocity $\mathbf{v}_i^{n+1}$ is directly used to compute $\mathbf{x}_i^{n+1}$. The full sequence of steps is shown in Algorithm 1. When implemented correctly, this yields a fully functioning SPH simulator. However, due to explicit integration, this simulator is only stable for a restrictive set of time step sizes, pressure stiffness coefficients and viscosity coefficients.

2.2.2 Implicit

The previous section constructed a fully functional explicit SPH simulation method. Due to limitations inherent to explicit time integration, we will look at how these can be overcome by using implicit time integration alongside operator splitting. Indeed, current state-of-the-art SPH solvers commonly make use of implicit Euler time integration, which greatly improves simulation stability. Operator splitting is further employed, such that one does not have to solve the full PDE implicitly, but instead solves subproblems and then composes these to form the full solution. It is common for current SPH solvers to use fractional-step or Chorin splitting [CHORIN 1969, ISSA 1986, J. KIM and MOIN 1985] and to follow the scheme of

1. Computing non-pressure forces (viscosity, surface tension, turbulence, etc.),
2. Computing an intermediate velocity with constant accelerations and non-pressure forces,
3. Projecting pressure values to ensure a divergence-free final velocity using intermediate velocities.

The process of how to go from the full Navier-Stokes equations, Equation (2.3), to an algorithm that computes pressure and non-pressure forces implicitly will be detailed in the following. To do so, this thesis aims to be as general as possible and avoids some of the simplifications that are made in related works, such as IISPH [IHMSSEN et al. 2014a] and DFSPH [BENDER and KOSCHIER 2015].

Pressure Poisson Equation

For implicit integration, we run into the same problem as for explicit integration, namely that we need an equation to compute pressure forces. While we could use one of the closures defined in Equation (2.22), which use a stiffness parameter κ and allows for weak compressibility, we instead want

an equation that is accurate for incompressible fluids in the continuum. For this purpose, the typical closure is obtained by taking the divergence of the full Navier-Stokes equations

$$\nabla \cdot \frac{D\mathbf{v}}{Dt} = \nabla \cdot \left(-\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{v} + \mathbf{g} \right). \quad (2.29)$$

The divergence operator and the material derivative do not commute directly, but instead satisfy

$$\nabla \cdot \frac{D\mathbf{v}}{Dt} = \frac{D}{Dt} (\nabla \cdot \mathbf{v}) + \text{tr}((\nabla \mathbf{v})^2), \quad (2.30)$$

as follows from $\partial_i (\partial_t v_i + v_j \partial_j v_i)$ in index notation. Since we enforce that $\nabla \cdot \mathbf{v} = 0$ for mass conservation at all times, and therefore also $\frac{D}{Dt} (\nabla \cdot \mathbf{v}) = 0$, the left-hand side reduces to the purely kinematic term $\text{tr}((\nabla \mathbf{v})^2)$. The remaining terms simplify to

$$\begin{aligned} \nabla \cdot (\nu \nabla^2 \mathbf{v}) &\stackrel{\text{const. } \nu}{=} \nu \nabla^2 (\nabla \cdot \mathbf{v}) = 0, \\ \nabla \cdot \mathbf{g} &= 0, \end{aligned}$$

such that reordering for the pressure yields

$$\nabla \cdot \left(\frac{1}{\rho} \nabla p \right) = -\text{tr}((\nabla \mathbf{v})^2). \quad (2.31)$$

This is the general form for the pressure Poisson equation, with a velocity source term, assuming that the kinematic viscosity ν is constant. In the Eulerian frame, this source term is equivalently written as $-\nabla \cdot (\mathbf{v} \cdot \nabla \mathbf{v})$, since $\nabla \cdot (\mathbf{v} \cdot \nabla \mathbf{v}) = \text{tr}((\nabla \mathbf{v})^2)$ for divergence-free $\mathbf{v}$. Attempting to solve this equation alongside the Navier-Stokes equations results in a coupled system of non-linear equations between velocity and pressure and is generally not attempted for SPH simulation methods. Instead, an operator splitting scheme is applied to first compute velocity and then compute pressure using the “known” velocity. This splitting scheme splits up the full equation into two discrete steps, namely computing an intermediate velocity $\mathbf{v}^*$ from non-pressure forces

$$\frac{\mathbf{v}^* - \mathbf{v}^n}{\Delta t} = \nu \nabla^2 \mathbf{v}^n + \mathbf{g}, \quad (2.32)$$

and then using backward Euler integration to compute the pressure forces

$$\frac{\mathbf{v}^{n+1} - \mathbf{v}^*}{\Delta t} = -\frac{1}{\rho^*} \nabla p^{n+1}. \quad (2.33)$$

This kind of splitting into velocity and pressure is frequently used in the simulation of incompressible fluids, albeit in a variety of slightly modified forms [BENDER and KOSCHIER 2017, IHMSEN et al. 2014a, SOLENTHALER and PAJAROLA 2009].

We can take the divergence of Equation (2.33) and require that $\nabla \cdot \mathbf{v}^{n+1} = 0$, resulting in

$$\begin{aligned} \nabla \cdot \mathbf{v}^{n+1} = 0 &\Leftrightarrow \nabla \cdot \mathbf{v}^* - \Delta t \nabla \cdot \left(\frac{1}{\rho^*} \nabla p^{n+1} \right) = 0, \\ &\Leftrightarrow \nabla \cdot \left(\frac{1}{\rho^*} \nabla p^{n+1} \right) = \frac{1}{\Delta t} \nabla \cdot \mathbf{v}^*, \end{aligned} \quad (2.34)$$

where ρ^* is fluid density at the intermediate step. It should be noted that this equation is the discrete Lagrangian counterpart of Equation (2.31). The intermediate velocity $\mathbf{v}^*$ collects all non-pressure terms, such that taking its divergence and rearranging for pressure on the LHS recovers Equation (2.31) in the limit $\Delta t \rightarrow 0$. For a finite time step, however, the two are not identical. In the Lagrangian frame, the advective contribution is not part of $\mathbf{v}^*$, but is instead realized by the motion of the particles, such that the source term $\text{tr}((\nabla \mathbf{v})^2)$ of Equation (2.31) is not represented within a single step. Equivalently,

projecting onto $\nabla \cdot \mathbf{v}^{n+1} = 0$ on the current particle configuration does not yield a divergence-free velocity field on the advected configuration, since $\nabla \cdot$ and $\frac{D}{Dt}$ do not commute, cf. Equation (2.30). This form of the PPE is used in practice and results in stable simulations, but it is known to lead to a density drift/loss of volume over time [KUGELSTADT et al. 2021b], which is consistent with this splitting error and further numerical errors, depending on the actual method and implementation. Instead, Equation (2.34) can be reformulated using the discretized version of the continuity equation, Equation (2.21), to replace the velocity divergence by the density deviation

$$\frac{\rho^* - \rho^n}{\Delta t} = -\rho^n \nabla \cdot \mathbf{v}^*, \quad (2.35)$$

$$\Leftrightarrow \nabla \cdot \mathbf{v}^* = \frac{\rho^n - \rho^*}{\Delta t \rho^n}. \quad (2.36)$$

If we also assume that the density at time n is equal to the rest density ρ_0 , we have

$$\nabla \cdot \mathbf{v}^* = \frac{\rho_0 - \rho^*}{\Delta t \rho_0}. \quad (2.37)$$

This allows us to rewrite Equation (2.34) in terms of local density deviations

$$\nabla \cdot \left(\frac{1}{\rho^*} \nabla p^{n+1} \right) = \frac{\rho_0 - \rho^*}{\Delta t^2 \rho_0}. \quad (2.38)$$

This was a somewhat lengthy but mostly complete derivation of the form that the PPE usually takes in the context of SPH simulations, and it can also be found in the same form in the state-of-the-art report by KOSCHIER et al. [2022] with a small difference. Namely, the density on the LHS is assumed to be equal to the rest density ρ_0 instead of the density at the intermediate step ρ^* and can therefore be canceled out of the equation

$$\Delta t \nabla^2 p^{n+1} = \frac{\rho_0 - \rho^*}{\Delta t}. \quad (2.39)$$

However, doing so can result in inconsistencies as the density is variable across the incremental step, which needs to be taken into account. As a matter of fact, most implicit SPH pressure solvers already do this, as the Laplacian of the pressure force is calculated in two first-order derivatives, where

$$\mathbf{a}_{i,p} = -\frac{\nabla p_i}{\rho_i} \approx -\sum_{j \in \mathcal{N}_i} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla_i W_{ij}, \quad (2.40)$$

is the symmetric version of the first derivative and

$$\nabla \cdot \left(\frac{\nabla p_i}{\rho_i} \right) \approx -\nabla \cdot \mathbf{a}_{i,p} = -\sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} (\mathbf{a}_{j,p} - \mathbf{a}_{i,p}) \cdot \nabla_i W_{ij}, \quad (2.41)$$

is another first derivative of the pressure accelerations. This form assumes that the density is variable and does indeed compute the divergence of the full pressure acceleration numerically, even if it is often written differently. However, there is still a *slight* inconsistency, as the density values at the start of the timestep, i.e., ρ_i^n are used here, instead of the density values at the intermediate step ρ_i^* , as would be consistent with the derivation. The full discretized per-particle PPE in velocity form is given by

DIVERGENCE PPE

$$\sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} (\mathbf{a}_{i,p} - \mathbf{a}_{j,p}) \cdot \nabla_i W_{ij} = \frac{1}{\Delta t} \nabla \cdot \mathbf{v}_i^*, \quad (2.42)$$

and in density form by

DENSITY PPE

$$\sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} (\mathbf{a}_{i,p} - \mathbf{a}_{j,p}) \cdot \nabla_i W_{ij} = \frac{\rho_0 - \rho_i^*}{\Delta t^2 \rho_0}. \quad (2.43)$$

The density ρ_i is computed using Equation (2.25) while the divergence of the velocity is typically computed using the same anti-symmetric SPH approximation for first-order derivatives as is used for the divergence of the pressure gradient in Equation (2.41)

$$\nabla \cdot \mathbf{v}_i = \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} (\mathbf{v}_j - \mathbf{v}_i) \cdot \nabla_i W_{ij}. \quad (2.44)$$

In practice, the density ρ_i^* is computed by using Equation (2.35), which introduces the divergence of the velocity field into the density deviation, resulting in the RHS of Equation (2.43) to look like

$$\frac{\rho_0 - \rho_i^*}{\Delta t^2 \rho_0} = \frac{\rho_0 - \left(\rho_i - \Delta t \rho_0 \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} (\mathbf{v}_j^* - \mathbf{v}_i^*) \cdot \nabla_i W_{ij} \right)}{\Delta t^2 \rho_0}, \quad (2.45)$$

which contains only known quantities. It would also be possible to advect the particles into the intermediate state and to determine ρ^* using Equation (2.25). This is however not done in practice.

Note that these equations are still linear in the pressure values, which is why we can use typical linear-system solution methods to find the actual pressure values p_i . The most common method in SPH is to use either matrix-free conjugate gradient or Jacobi iteration.

Relaxed Jacobi Iteration

In this section, the relaxed Jacobi iteration will be presented, which is frequently used to solve the PPEs in Equation (2.42) and Equation (2.43). For this, we first define the linear system $\mathbf{A}\mathbf{p} = \mathbf{b}$ implied by these equations as

$$\begin{aligned} (\mathbf{A}\mathbf{p})_i &= \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} (\mathbf{a}_{i,p} - \mathbf{a}_{j,p}) \cdot \nabla_i W_{ij}, \\ (\mathbf{b}^{\text{density}})_i &= \frac{\rho_0 - \rho_i^*}{\Delta t^2 \rho_0}, \\ (\mathbf{b}^{\text{div}})_i &= \frac{1}{\Delta t} \nabla \cdot \mathbf{v}_i^*. \end{aligned}$$

The (relaxed) Jacobi iteration starts with an initial guess for the vector of pressures $\mathbf{p}$ and iteratively refines the individual values according to the scheme

$$p_i^{(l+1)} = (1 - \omega) p_i^{(l)} + \frac{\omega}{A_{ii}} (b_i - \sum_{j \neq i} A_{ij} p_j^{(l)}), \quad (2.46)$$

where A_{ii} and A_{ij} denote individual elements of matrix $\mathbf{A}$. This can be rewritten as

$$\sum_{j \neq i} A_{ij} p_j^{(l)} = (\mathbf{A}\mathbf{p}^{(l)})_i - A_{ii} p_i^{(l)}. \quad (2.47)$$

Doing so simplifies implementation, since we now only need to determine the full sparse matrix-vector product $\mathbf{A}\mathbf{p}$ and the diagonal matrix elements A_{ii} , instead of all individual matrix elements A_{ii} and A_{ij} . Finally, this yields the relaxed Jacobi update rule for the pressure values

$$\begin{aligned} p_i^{(l+1)} &= (1 - \omega) p_i^{(l)} + \frac{\omega}{A_{ii}} (b_i - (\mathbf{A}\mathbf{p}^{(l)})_i + A_{ii} p_i^{(l)}), \\ &= p_i^{(l)} + \frac{\omega}{A_{ii}} (b_i - (\mathbf{A}\mathbf{p}^{(l)})_i). \end{aligned} \quad (2.48)$$

This iteration further requires a stopping criterion, which tells the solver that the solution is sufficiently close to the “true” solution. Convergence is typically measured in the average absolute residual of the linear system.

Pressure Clamping

A significant issue for SPH simulation, and implicit pressure solvers in particular, is the problem typically known as “particle-deficiency” or “tensile-instability” at the fluid surface. The approximation properties of the SPH method are only well-defined for filled volumes. However, at the surface of the fluid, roughly half of the domain will be empty of particles and therefore lacking in integration points. This causes the fluid density to be severely underestimated by Equation (2.25) and a resulting large attractive pressure force that attempts to pack fluid very densely at the surface. The result is either unnatural particle clumping at the fluid surface for explicit solvers, or simulation instabilities for implicit solvers.

A common technique to mitigate this problem is to apply so-called “pressure clamping”, which simply clamps negative pressures to zero

$$\widehat{p}_i = \max(p_i, 0). \quad (2.49)$$

Pressure clamping can also be applied for the implicit solver using relaxed Jacobi iteration by clamping each individual iteration update

$$p_i^{(l+1)} = \max\left(p_i^{(l)} + \frac{\omega}{A_{ii}}(b_i - (\mathbf{A}\mathbf{p}^{(l)})_i), 0\right). \quad (2.50)$$

Applying pressure clamping unfortunately prohibits the use of potentially more efficient linear solvers, such as the conjugate gradient method. Clamping introduces a discontinuity and therefore causes the previously linear system to essentially become non-linear in nature.

Non-pressure Forces

The implicit pressure solver can be used as it was derived above. However, the non-pressure forces as computed by Equation (2.32) are still explicitly integrated. We can change this to

$$\frac{\mathbf{v}^* - \mathbf{v}^n}{\Delta t} = \nu \nabla^2 \mathbf{v}^* + \mathbf{g}, \quad (2.51)$$

for implicit integration. As done by WEILER et al. [2018] we can formulate this as a linear system in terms of velocity $\mathbf{v}^*$

$$\frac{\mathbf{v}_i^* - \mathbf{v}_i^n}{\Delta t} = 2(d+2)\nu \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} \frac{\mathbf{v}_{ij}^* \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \nabla_i W_{ij} + \mathbf{g}, \quad (2.52)$$

which results in a system of linear equations

$$\begin{aligned} (\mathbf{A}\mathbf{v}^*)_i &= \mathbf{v}_i^* - \Delta t 2(d+2)\nu \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} \frac{\mathbf{v}_{ij}^* \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \nabla_i W_{ij}, \\ (\mathbf{b})_i &= \mathbf{v}_i^n + \Delta t \mathbf{g}, \end{aligned}$$

which can be solved very efficiently using the matrix-free conjugate gradient method, because no clamping has to be applied here. Optionally, a block-diagonal preconditioner and initial guess can be used to accelerate convergence.

Algorithm 2 Implicit SPH

```

1: procedure FLUIDSOLVE
2:   for all particles  $i$  do
3:      $\rho_i \leftarrow \text{COMPUTEDENSITY}(i)$  ▷ Equation (2.25)
4:   end for
5:   for all particles  $i$  do
6:      $\mathbf{v}_i^* \leftarrow \mathbf{v}_i + \Delta t \mathbf{g} + \frac{\Delta t}{\rho_i} \text{COMPUTEVISCOSITY}(i)$  ▷ Equation (2.27)
7:   end for
8:    $p_i^0 \leftarrow 0, \forall i$ 
9:    $l \leftarrow 0$ 
10:  repeat
11:    for all particles  $i$  do
12:       $p_i^{(l+1)} \leftarrow \max \left( p_i^{(l)} + \frac{\omega}{A_{ii}} (b_i - (\mathbf{A} \mathbf{p}^{(l)})_i), 0 \right)$  ▷ Equation (2.50)
13:    end for
14:     $l \leftarrow l + 1$ 
15:  until  $\rho^{\text{err}} / \rho_0 < \text{tol}$  ▷ typically  $\text{tol} \approx 10^{-3}$ 
16:  for all particles  $i$  do
17:     $\mathbf{v}_i^{n+1} \leftarrow \mathbf{v}_i^* + \Delta t \mathbf{a}_{i,p}(\mathbf{p}^{(l)})$  ▷ Equation (2.40)
18:  end for
19:  for all particles  $i$  do
20:     $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i^{n+1}$ 
21:  end for
22: end procedure

```

Algorithm

In this section, we derived the pressure-Poisson equation (PPE) to determine pressure values in implicit solvers for incompressible fluids. Solving implicit integration for pressure and velocity simultaneously is a very challenging problem, which is why operator splitting/Chorin splitting is commonly used to separate the equation into two subsequent steps. The first part, Equation (2.32), computes an intermediate velocity field $\mathbf{v}^*$ that takes into account all non-pressure forces. The second part, Equation (2.33), determines the pressure forces by using (one of) two alternate but equivalent formulations of the previously derived PPE. One uses the velocity divergence as a source term, Equation (2.42), while the other uses deviation from rest density as a source term, Equation (2.43).

In the most obvious case, the non-pressure forces are computed by explicit integration and the pressure forces by implicit integration through the PPE. An example of such a scheme is shown in Algorithm 2. However, as shown in the previous subsection and by WEILER et al. [2018] it is possible to also use implicit integration for non-pressure forces. Nevertheless, this still uses two distinct and separate steps to compute pressure and non-pressure forces. An attempt at strong coupling can be made for instance by iterating between implicit pressure and non-pressure forces [S. LIU et al. 2022], which however, incurs increased computational cost.

2.3 Summary

This chapter briefly presented the fundamentals of the SPH method and applied them to fluid simulation. Specifically, we looked at how to construct an explicit and an implicit simulation method. The explicit method used an analytical closure to compute pressure forces from rest density deviations. The implicit method instead constructed an additional equation by taking the divergence of the entire Navier-Stokes equation, which results in the so-called pressure Poisson equation (PPE). Furthermore, the equation is split into two separate steps, the first of which only considers non-pressure forces and the second only pressure forces. Finally, implicit solvers typically use relaxed Jacobi iteration to solve the PPE and apply

pressure clamping to prevent artifacts at the free surface. Implicit pressure solvers represent the current state-of-the-art in SPH, with active progress being made on strong coupling between multiple materials [T. XIE et al. [2023](#)]. In the next chapters, we will first look at an effective way to model surface tension with the SPH method. Afterwards, we will examine applications to complex melt-driven processes, namely welding and thermal spray.

Implicit Surface Tension for SPH Fluid Simulation



Surface tension effects can be readily observed in everyday situations at home or in nature. A dripping water faucet, the formation and merging of soap bubbles or the movement of water striders on a pond are just a few examples which are impossible without surface tension. There are many use cases for simulating these effects in computer animation, whether for small-scale water features or artistic effects like droplets running down a soda can. Even outside these applications, modeling surface tension is crucial for specific engineering problems, for example in additive manufacturing. Overall, the numerical simulation of surface tension is an active area of research in various fields. In the context of this thesis, this chapter addresses the objective defined in Chapter 1 of improving stability and efficiency of point-based simulation methods via implicit formulations.

At its core, surface tension is the result of unbalanced molecular attraction forces at the fluid interface. This imbalance results in forces minimizing the surface area of the fluid and tensile stresses parallel to the surface, which can roughly be compared to a kind of “elastic” membrane. Of course this is a highly simplified description and actually incorporating surface tension in continuum mechanical fluid simulations poses several challenges. First of all, surface tension is not inherent to models based on the Navier-Stokes equations and has to be modelled explicitly in contrast to molecular simulations, where surface tension arises directly from the intermolecular potentials. Furthermore, depending on the choice of numerical discretization method, the fact that surface tension effects specifically apply to the fluid surface can cause nontrivial accuracy or stability issues, as well as complicate coupling with the rest of the fluid model.

Previous SPH surface tension methods in computer graphics have, for the most part, used explicit time integration. Building on the SPH fundamentals presented in Chapter 2, this becomes an issue when aiming to simulate cases with large surface tension coefficients, as they end up requiring small time steps in order to obtain a stable simulation. In addition, the simulation of certain effects containing

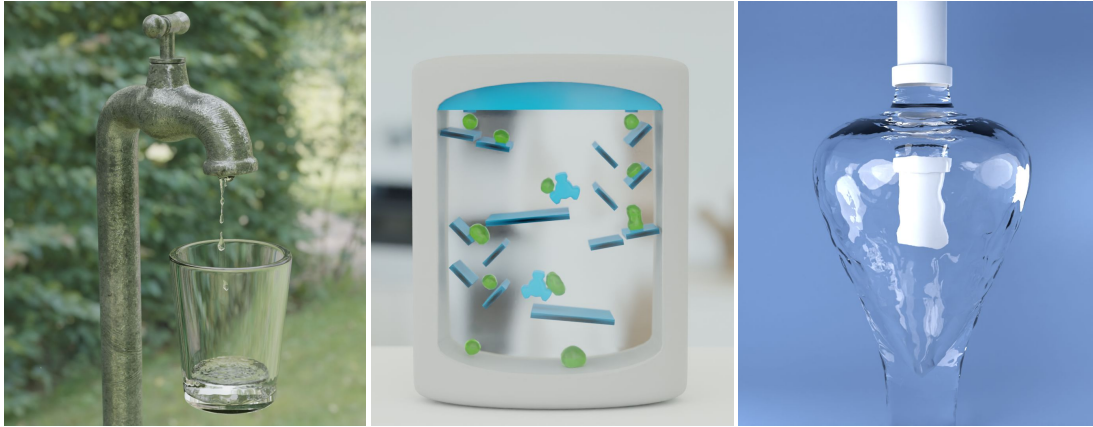


Figure 3.1. Left: A dripping water faucet showing realistic stream and droplet formation for small-scale water effects. Middle: A child’s toy in which droplets fall through a maze of obstacles due to gravity. A reservoir at the top of the toy causes droplets to be formed through funnels, falling through the maze and recombining into a single fluid mass at the bottom. Right: A water-bell, which is formed due to surface tension when two opposing vertical water streams meet. Our surface tension method is able to maintain the shape, while the surface at its thinnest point is just 2–3 particles thick.

surface tension, such as the formation of thin sheets during splashing, requires a delicate balance of forces and stability of surface tension that has not yet been shown using explicit SPH surface tension methods. As such, we aim in this chapter to develop an implicit scheme for computing surface tension forces in order to improve simulation stability and enable the use of large force coefficients without reducing time step sizes. To this end we present the following contributions.

First, we derive a cohesion force which is inspired by the balance of forces at the molecular scale. This force is defined such that it only acts on the surface of the fluid while cancelling out in the interior. This property is achieved by making use of the symmetric SPH kernel function, enabling our model to seamlessly integrate with existing SPH solvers. After discretizing with SPH we apply the backward Euler method for time integration, yielding a non-linear system of equations. For the system we subsequently propose a non-trivial linearization for improved solver performance.

Second, we extend our surface tension force by an adhesion term for interaction with rigid objects and further strongly couple it with a state-of-the-art implicit viscosity force [WEILER et al. 2018]. The strong coupling is achieved by solving a single monolithic linear system. This results in more stable simulations compared to solving surface tension and viscosity sequentially by avoiding coupling artifacts, which become more noticeable for cases with both larger surface tension and viscosity coefficients. The monolithic solver is able to directly find a solution satisfying both equations, instead of having to iterate between them or favoring one over the other by solving them in succession.

Finally, we demonstrate that using our method, we are able to simulate a large variety of surface tension effects. These range from cohesion dominant cases, such as droplet formation and fluid crowns, to adhesion dominant cases, such as covered spheres and surface wetting, as well as cases requiring a delicate balance between the two, such as a catenoid between two rigid tori. In Figure 3.1, a selection of these complex surface tension effects is shown.

3.1 Related Work

Over the years, a number of different approaches have been developed to simulate surface tension using a variety of discretizations. As the goal of our work is to present an improved and implicit surface tension model for SPH simulation, we first present a brief review of SPH surface tension approaches.

Afterwards, we outline the broader context of how surface tension can be simulated using different discretization and simulation methods. Finally, we summarize the findings most relevant to our work.

3.1.1 Particle-Based Surface Tension

At a microscopic scale, surface tension is known to originate from an imbalance of molecular forces at fluid interfaces [NELKON 1969, POPINET 2018]. As SPH is a particle-based Lagrangian discretization method, some works have attempted to emulate this molecular force by introducing attractive forces between particles [BECKER and TESCHNER 2007, CLAVET et al. 2005, TARTAKOVSKY and MEAKIN 2005, T. YANG et al. 2016, 2017]. These works have partially been documented to suffer from increased particle clustering at the surface [HUBER et al. 2015]. T. YANG et al. [2016, 2017] show that using a significantly larger support radius can alleviate the particle clustering problems in some of the previous works and propose an improved particle-particle interaction force for surface tension. They use a Jacobi iteration-based strong coupling between pressure and non-pressure forces (including surface tension). This however comes at the cost of significantly increased computational complexity.

At the macroscopic scale, surface tension is generally derived through the thermodynamic equilibrium across the fluid interface [LANDAU and LIFSHITZ 2013]. Derived from this is the Young-Laplace equation that describes forces acting on the surface of the fluid. BRACKBILL et al. [1992] reformulated this equation using a mollifier function for volumetric simulations. This is called the Continuum Surface Force (CSF) approach. This approach was initially adopted for SPH simulations by MORRIS [2000], and subsequently by several others [HE et al. 2015, X. HU and ADAMS 2006, M. MÜLLER et al. 2003, M. ZHANG 2010, ZORILLA et al. 2020]. It integrates well with the SPH framework and is able to produce comparable results to cohesion based approaches when tuned correctly. Nevertheless, the computation of the curvature with SPH is known to be prone to errors [MORRIS 2000], especially in the interior of the fluid, requiring the usage of non-trivial thresholds or explicit surface classification to alleviate this. Notably, ZORILLA et al. [2020] present an improved, but computationally expensive approach for computing accurate curvature values. Finally, AKINCI et al. [2013] present a hybrid approach utilizing both curvature (through non-normalized normal vectors) and cohesion forces aiming to combine the best of both methods. This method is further built upon by X.-K. WANG et al. [2017], by coupling it with the IISPH method [IHMSSEN et al. 2014a]. Comparisons of select models based on the microscale considerations (inter-particle force) and models based on the CSF approach are presented by HUBER et al. [2015] and L. YANG et al. [2019].

3.1.2 Further Approaches

Grid-Based Similar to Lagrangian particle methods, the fluid interfaces which are required for surface tension are often only implicitly defined in grid-based methods. The surfaces can be reconstructed by using an interface tracking technique such as the level-set method [AANJANEYA et al. 2013, KANG et al. 2008, NI et al. 2020, PATKAR et al. 2013, W. ZHENG et al. 2009] or the volume of fluid method [ALBADAWI et al. 2013, HONG and C.-H. KIM 2003, 2005]. However, while grid-based Eulerian methods allow for changing topology, the resolution is often constrained by the necessity to discretize a very large region outside of the actual region of interest. A recent work by Y.-L. CHEN et al. [2020] addresses this issue by proposing a cut-cell method able to resolve the liquid surface at sub-grid scale. The level-set method has also been successfully applied to the simulation of soap bubbles [KANG et al. 2008] as well as air bubbles and bubble clusters [W. ZHENG et al. 2009].

Mesh-Based Lagrangian mesh based approaches typically facilitate accurate computation of surface tension forces by explicitly modeling the surface of the fluid. There are a number of mesh-based ap-

proaches incorporating surface tension effects focusing on surface only (non-volumetric) discretizations of fluids. These include the simulation of droplets covering other objects [Y. ZHANG et al. 2012], the simulation of viscous thin sheets [BATTY et al. 2012] and threads [BERGOU et al. 2010], thin films on other surfaces [AZENCOT et al. 2015] and surface only discretizations of bubbles and thin films [DA et al. 2015, ISHIDA et al. 2020, 2017]. B. ZHU et al. [2015, 2014] propose a codimensional simulation method to allow for changing surface topology. Finally, MISZTAL et al. [2012, 2014] derive and implement a fully coupled solver containing pressure, viscosity and implicit surface tension on moving unstructured tetrahedral meshes.

Hybrid Hybrid methods typically aim to make use of Lagrangian particles and Eulerian grids together, combining benefits of both approaches to capture incredibly diverse effects. W. ZHENG et al. [2015] apply a hybrid particle marker-and-cell grid to better capture free surface flows. HONG et al. [2008] propose a novel approach of using SPH to simulate air bubbles within an Eulerian grid based simulator. A versatile surface tension approach based on the minimization of an energy which is proportional to the surface area of the liquid is proposed by HYDE et al. [2020]. This approach can be used within the material-point method or particle-in-cell approach and has been further extended to model phenomena with spatially varying surface tension coefficients [J. CHEN et al. 2021]. BOYD and BRIDSON [2012] propose a FLIP method which explicitly reconstructs the fluid surface for accurate surface tension computation. A hybrid method which uses a Lagrangian surface mesh instead of particles is introduced by SCHROEDER et al. [2012], in which the surface mesh is coupled to an Eulerian discretization. A similar approach has also been proposed by RUAN et al. [2021] and has been extended to strongly couple with solid objects interacting with the surface.

Recently, XING et al. [2022] introduced a hybrid position-based method for simulating small-scale surface tension phenomena, by detecting surface particles, reconstructing a local triangle mesh and formulating surface tension constraints.

3.1.3 Summary

While a number of particle-based surface tension models have already been proposed, there have always been drawbacks and limitations to what effects a single method was able to capture. Most previous SPH-based methods modeled surface tension with explicit forces, which can cause stability issues for large parameter values or large time steps. We propose an implicit surface tension method in SPH-based fluid simulation to address this problem. This includes an implicit approach for boundary adhesion, as well as strong coupling with an implicit velocity solver.

As opposed to the method of XING et al. [2022], we do not have to resort to detecting surface particles explicitly, or potentially costly reconstructions of local meshes. Surface particles are implicitly detected using our proposed cohesion kernel and surface forces constructed such that surface area is reduced.

Compared to the approach of T. YANG et al. [2016, 2017], we are able to implicitly integrate surface tension forces using comparatively small particle neighborhoods of only 30-40 particles. We also use conjugate gradient (CG) iteration to solve a linear system instead of nonlinear Jacobi iteration, which also has some more favorable convergence properties.

Ultimately, we show that the implicit nature of our method enables us to plausibly simulate a wide variety of surface tension phenomena within a purely particle-based surface tension model.

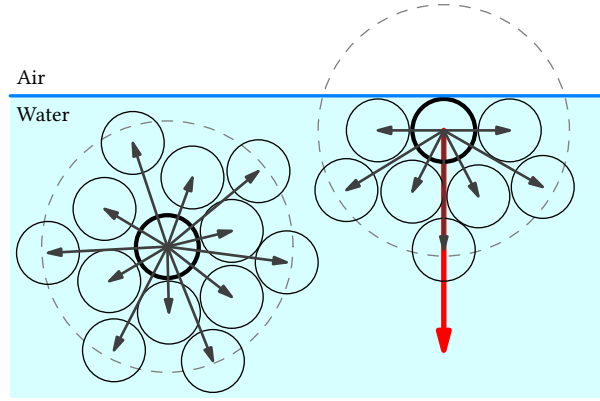


Figure 3.2. Surface tension force as a result of micro-scale force imbalances. Interior particles are completely surrounded in all directions, so that pairwise cohesive forces (grey arrows) add up to zero. Particles at the surface have no neighboring particles in a specific region, such that summation of pairwise cohesive forces (grey arrows) causes a net force (red arrow) in the direction of the fluid interior.

3.2 Method

As surface tension is fundamentally an effect caused by unbalanced molecular forces at the micro-scale (see Figure 3.2), this results in a net force at the fluid interface. It can be thought of as “tightening” the surface, forming a kind of elastic membrane with the tendency to reduce the surface area of the interface.

In the following, we will derive our surface tension force from continuum micro-scale considerations. We chose this approach because transferring surface tension originating from molecular cohesive forces to the domain of SPH particle simulation is the more natural transition, resulting in a native formulation. Meanwhile, curvature based approaches require specialized solutions to prevent non-zero curvature values in the interior of the fluid. Inspired by the molecular cohesive forces we define a continuous model for surface tension forces, which we discretize in space using SPH. For time discretization, we propose an adapted variant of the linearized backward Euler method. We strongly couple this linear system with implicit viscosity forces and further extend it to be able to include adhesion effects at solid boundaries.

We use the well-established operator splitting approach for SPH, previously discussed in Section 2.2.2, which means that we separately compute and integrate pressure and non-pressure forces. This weak coupling between pressure and non-pressure forces is the current standard for SPH simulation [IHSEN et al. 2014b, KOSCHIER et al. 2022], and we have not found it to be prohibitive for the effects we aimed to simulate with our method, especially when using the implicit pressure solver Divergence-Free Smoothed Particle Hydrodynamics (DFSPH) [BENDER and KOSCHIER 2017]. Nevertheless, we also present a short description of an iteration-based strong pressure-coupled approach, as well as how it relates to the weak pressure-coupled algorithm in Section 3.2.5. A comparison and brief discussion between the coupling schemes is then provided in Section 3.3.1.

Finally, we propose a particular weighting function for our cohesion force approach, which relates how strongly particles interact with each other depending on their distance vector. We construct it in a way to prevent previously mentioned clustering artifacts at the surface.

3.2.1 Derivation

For our method, we adopt the micro-scale assumption that the surface tension force results from a force imbalance at the fluid interface, as shown in Figure 3.2. We formalize this in the continuum as

$$\mathbf{f}_{\text{st}}(\mathbf{x}^*) = -\sigma \int_{\Omega} \frac{\mathbf{x}^* - \mathbf{x}}{\|\mathbf{x}^* - \mathbf{x}\|} W(\mathbf{x}^* - \mathbf{x}; h) d\mathbf{x}, \quad (3.1)$$

where W is a compactly supported weighting function on the integration domain Ω , and $\sigma \in \mathbb{R}^+$ denotes the surface tension coefficient. Note that σ is not directly proportional to the otherwise commonly used surface tension parameters derived for curvature-based surface tension, although it shares the same units $[\text{N m}^{-1}]$. W further depends on the distance vector between evaluation point $\mathbf{x}^*$ and integration point $\mathbf{x}$, and the smoothing length h (which is equivalent to the corresponding SPH quantity).

Intuitively speaking, this integral accumulates force contributions of all points $\mathbf{x}$ in the domain in the direction of evaluation point $\mathbf{x}^*$ and weights them through evaluation of the function W . As long as W is an even function, the total force will be zero when integrating over a symmetric domain, e.g., the interior of the fluid. Only when integrating over non-symmetric domains, e.g., at the fluid surface, does the integral evaluate to a non-zero value. In addition, W can be adjusted to exhibit different attractive or repulsive properties. For example, at the molecular level, W might be chosen such that the net force corresponds to the force resulting from the LENNARD-JONES [1931] potential. This potential is an even function yielding an attractive force when molecules are further than some molecule-specific distance apart, diminishing smoothly with increasing distance. For smaller distances it results in a rapidly increasing repulsive force, proportional to the inverse distance.

In our work we denote W^{cubic} as the standard cubic spline kernel [MONAGHAN 1992] often used in SPH simulations, with a small modification resulting in

$$W(\mathbf{x}^* - \mathbf{x}; h) = \|\mathbf{x}^* - \mathbf{x}\| W^{\text{st}}(\mathbf{x}^* - \mathbf{x}; h), \quad (3.2)$$

$$W^{\text{st}}(\mathbf{x}^* - \mathbf{x}; h) = \theta \begin{cases} W^{\text{cubic}}(\mathbf{x}^* - \mathbf{x}; h), & \text{for } \|\mathbf{x}^* - \mathbf{x}\| > d_{\text{part}} \\ W^{\text{cubic}}\left(\frac{\mathbf{x}^* - \mathbf{x}}{\|\mathbf{x}^* - \mathbf{x}\|} d_{\text{part}}; h\right), & \text{otherwise.} \end{cases} \quad (3.3)$$

Here d_{part} denotes the particle diameter, such that the value of W^{st} , and implicitly also the value of W , is clamped to a constant value as the distance between particles decreases below d_{part} . In all of our simulations we set $h = 2d_{\text{part}}$. The additional factor $\theta = \frac{10}{7}$ (for 3D simulations) is derived, such that the modified cubic-spline kernel function is again normalized, i.e. the integral of the kernel over the compact support is equal to one.

The multiplication by the distance $\|\mathbf{x}^* - \mathbf{x}\|$ and the clamping of the cubic-spline kernel ensure that the attraction is maximal at a one-particle-diameter distance d_{part} and linearly tends to zero as particles get closer. As such, the SPH discretization of Equation (3.1) is given by

$$\mathbf{f}_{\text{st}}(\mathbf{x}_i) = -\sigma \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} (\mathbf{x}_i - \mathbf{x}_j) W^{\text{st}}(\mathbf{x}_i - \mathbf{x}_j; h), \quad (3.4)$$

where m_j is the mass and ρ_j is the density of discrete particle j in the neighborhood $\mathcal{N}_i$ around position $\mathbf{x}_i$. Note that due to the dependency on the density ρ_j , the term also depends on all positions $\mathbf{x}_k$ in the neighborhood $k \in \mathcal{N}_j$, since

$$\rho_j = \sum_{k \in \mathcal{N}_j} m_k W^{\text{cubic}}(\mathbf{x}_j - \mathbf{x}_k; h). \quad (3.5)$$

This dependency has considerable significance for our implicit integration scheme which we introduce in the following. The formulation in Equation (3.4) is very similar to the surface tension force employed

by BECKER and TESCHNER [2007], yet differs in derivation and construction of the custom clamped kernel function.

In the following, we will use the shorthands $W_{ij} = W^{\text{st}}(\mathbf{x}_i - \mathbf{x}_j; h)$ and $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. In order to symmetrize Equation (3.4), we introduce $\bar{\rho}_{ij} = \frac{1}{2}(\rho_i + \rho_j)$ and $\bar{m}_{ij} = \frac{1}{2}(m_i + m_j)$ as the averaged densities and masses. This yields the symmetric force

$$\bar{\mathbf{f}}_{\text{st}}(\mathbf{x}_i) = - \sum_{j \in \mathcal{N}_i} \underbrace{\sigma \frac{\bar{m}_{ij}}{\bar{\rho}_{ij}} \mathbf{x}_{ij} W_{ij}}_{\bar{\mathbf{f}}_{ij,\text{st}}} . \quad (3.6)$$

Finally, the acceleration on particle i resulting from the discretized surface tension force can be written as

$$\mathbf{a}_{i,\text{st}} = -\frac{1}{m_i} \sum_{j \in \mathcal{N}_i} \bar{\mathbf{f}}_{ij,\text{st}} = -\frac{\sigma}{m_i} \sum_{j \in \mathcal{N}_i} \frac{\bar{m}_{ij}}{\bar{\rho}_{ij}} \mathbf{x}_{ij} W_{ij} . \quad (3.7)$$

Discretizing this equation in time using the forward Euler method yields

$$\mathbf{a}_{i,\text{st}} = \frac{\mathbf{v}_i^{t+1} - \mathbf{v}_i^t}{\Delta t} = -\frac{\sigma}{m_i} \sum_{j \in \mathcal{N}_i} \frac{\bar{m}_{ij}}{\bar{\rho}_{ij}^t} \mathbf{x}_{ij}^t W_{ij}^t , \quad (3.8)$$

resulting in a closed expression for $\mathbf{v}_i^{t+1}$. Discretizing with the backward Euler method instead, this equation becomes

$$\mathbf{a}_{i,\text{st}} = \frac{\mathbf{v}_i^{t+1} - \mathbf{v}_i^t}{\Delta t} = -\frac{\sigma}{m_i} \sum_{j \in \mathcal{N}_i} \frac{\bar{m}_{ij}}{\bar{\rho}_{ij}^{t+1}} \mathbf{x}_{ij}^{t+1} W_{ij}^{t+1} , \quad (3.9)$$

which results in a non-linear system for the velocity $\mathbf{v}_i^{t+1}$, since the (non-linear) kernel function and the density both depend on $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i^{t+1}$.

3.2.2 Implicit Surface Tension

In the following, we describe the implicit time integration in more detail. Using an implicit method improves the stability significantly when using large surface tension forces in combination with larger time steps. This enables the simulation of complex surface tension effects that explicit methods struggle to handle at typical time step sizes. The explicit version in Equation (3.8) directly yields a closed solution for velocity $\mathbf{v}_i^{t+1}$, while the implicit version in Equation (3.9) requires solving a non-linear system.

Unfortunately, a full non-linear solver such as Newton's method is not well suited for our specific problem. The combination of Newton iterations together with a sparse linear system solver quickly becomes prohibitively expensive, both in terms of computational complexity as well as memory requirements of storing the Jacobian and intermediate results for line-search. More crucially, a full Newton's method for surface tension would entail many more iterations over all particles, its neighbors, and neighbors of neighbors than just a linear system solve. It would further need to run the costly neighborhood search to evaluate new densities in each Newton iteration. As such, we instead describe our solution method for linearized implicit surface tension forces in the following.

Naive Linearization To be able to solve the non-linear system we choose to employ a linear approximation. The most obvious approach to this would be to simply linearize Equation (3.9) in velocity

$$\begin{aligned} \mathbf{a}_{i,\text{st}}(\mathbf{x}^{t+1}(\mathbf{v}^{t+1})) &= \mathbf{a}_{i,\text{st}}(\mathbf{x}^{t+1}(\mathbf{v}^t)) \\ &\quad + \nabla_{\mathbf{v}} \mathbf{a}_{i,\text{st}}(\mathbf{x}^{t+1}(\mathbf{v})) \Big|_{\mathbf{v}^t} (\mathbf{v}^{t+1} - \mathbf{v}^t) \\ &\quad + O(\Delta v^2) . \end{aligned} \quad (3.10)$$

Solving this linear system can be viewed as solving the non-linear system using a single Newton iteration. However, in our experiments we quickly disregarded this approach as untenable. The derivative has to be computed with respect to the velocities of all particles in the immediate neighborhood, as well as their immediate neighbors, due to the dependencies mentioned previously. Using an iterative matrix-free linear system solver, this led to significantly worse performance and scaling, quickly dominating all other terms in the SPH solver. In addition, we found that the quality and stability of the simulations using this approach were, if anything, slightly inferior to the following approach.

Improved Linearization Instead of directly linearizing Equation (3.9) in velocity, we instead split the term into two parts g and h

$$\frac{\mathbf{v}_i^{t+1} - \mathbf{v}_i^t}{\Delta t} = -\frac{\sigma}{m_i} \sum_{j \in \mathcal{N}_i} \bar{m}_{ij} \underbrace{\frac{W_{ij}^{t+1}}{\bar{\rho}_{ij}^{t+1}}}_{g_{ij}^{t+1}} \underbrace{\mathbf{x}_{ij}^{t+1}}_{h_{ij}^{t+1}}, \quad (3.11)$$

and linearize them independently with respect to time t . Realizing that the term h only depends linearly on particle positions $\mathbf{x}_i$ and $\mathbf{x}_j$, it is linearized using backward Euler time integration. This results in

$$h_{ij}^{t+1} = \left(\mathbf{x}_i^t + \Delta t \mathbf{v}_i^{t+1} - \mathbf{x}_j^t - \Delta t \mathbf{v}_j^{t+1} \right), \quad (3.12)$$

preventing any longer range dependencies on velocities outside of the immediate neighborhood of particle i . The other term g , which does have dependencies outside of the immediate neighborhood, is approximated using a first-order Taylor expansion at time t

$$g_{ij}^{t+1} \approx \tilde{g}_{ij}^{t+1} = g_{ij}^t + \Delta t \left. \frac{\partial g_{ij}}{\partial t} \right|_t = g_{ij}^t + \Delta t \left(\frac{\partial g_{ij}}{\partial \mathbf{x}} \cdot \mathbf{v} \right) \Big|_t. \quad (3.13)$$

This results in a formulation which only depends on quantities at time t . Therefore, it can be computed in advance of solving the linear system. The individual terms of g are

$$g_{ij}^t = \frac{W_{ij}^t}{\bar{\rho}_{ij}^t}, \quad (3.14)$$

and

$$\begin{aligned} \left(\frac{\partial g_{ij}}{\partial \mathbf{x}} \cdot \mathbf{v} \right) \Big|_t &= \frac{\nabla_i W_{ij}^t \cdot \mathbf{v}_i^t + \nabla_j W_{ij}^t \cdot \mathbf{v}_j^t}{\bar{\rho}_{ij}^t} \\ &\quad - \frac{W_{ij}^t}{2} \left(\frac{\sum_{k \in \mathcal{N}_i} m_k (\mathbf{v}_i^t - \mathbf{v}_k^t) \cdot \nabla_i W_{ik}^t}{\bar{\rho}_{ij}^{t^2}} \right. \\ &\quad \left. + \frac{\sum_{l \in \mathcal{N}_j} m_l (\mathbf{v}_j^t - \mathbf{v}_l^t) \cdot \nabla_j W_{jl}^t}{\bar{\rho}_{ij}^{t^2}} \right). \end{aligned} \quad (3.15)$$

The first line in this equation corresponds to the time derivative of the kernel function W_{ij} , which has been expanded using the product rule, see Equation (3.13), to the spatial derivative of the kernel function times the velocity. The spatial gradients ∇_i and ∇_j denote derivatives wrt. particle positions $\mathbf{x}_i$ and $\mathbf{x}_j$. The second and third line correspond to the time derivative of the reciprocal averaged density $\bar{\rho}_{ij}$, which has been expanded analogously. These terms can be recognized from the continuity equation as the negative divergence of the velocity field $\nabla \cdot \mathbf{v}$ times density ρ

$$-(\rho \nabla \cdot \mathbf{v})_i = \sum_{j \in \mathcal{N}_i} m_j (\mathbf{v}_i - \mathbf{v}_j) \cdot \nabla_i W_{ij}. \quad (3.16)$$

We precompute and store this scalar value per particle and thus eliminate the need for iterating over neighbors of neighboring particles in the linear solver. Now that all terms have been identified, we can assemble the final linearized system

$$\begin{aligned} \frac{\mathbf{v}_i^{t+1} - \mathbf{v}_i^t}{\Delta t} &= -\frac{\sigma}{m_i} \sum_{j \in \mathcal{N}_i} \bar{m}_{ij} \left(\mathbf{x}_{ij}^t + \Delta t \mathbf{v}_{ij}^{t+1} \right) \tilde{\mathbf{g}}_{ij}^{t+1}, \\ \Leftrightarrow \mathbf{v}_i^{t+1} + \Delta t^2 \frac{\sigma}{m_i} \sum_{j \in \mathcal{N}_i} \bar{m}_{ij} \mathbf{v}_{ij}^{t+1} \tilde{\mathbf{g}}_{ij}^{t+1} &= \mathbf{v}_i^t - \Delta t \frac{\sigma}{m_i} \sum_{j \in \mathcal{N}_i} \bar{m}_{ij} \mathbf{x}_{ij}^t \tilde{\mathbf{g}}_{ij}^{t+1}. \end{aligned} \quad (3.17)$$

This can also be written as

$$(\mathbf{I} + \Delta t \mathbf{A}) \mathbf{v}^{t+1} = \mathbf{v}^t + \mathbf{b}, \quad (3.18)$$

where

$$\begin{aligned} \mathbf{A}_{ii} &= \mathbf{I}_3 \frac{\sigma}{m_i} \Delta t \sum_{j \in \mathcal{N}_i} \bar{m}_{ij} \tilde{\mathbf{g}}_{ij}^{t+1}, \quad \mathbf{A}_{ij} = -\mathbf{I}_3 \frac{\sigma}{m_i} \Delta t \bar{m}_{ij} \tilde{\mathbf{g}}_{ij}^{t+1} \text{ for } i \neq j, \\ \mathbf{b}_i &= -\Delta t \frac{\sigma}{m_i} \sum_{j \in \mathcal{N}_i} \bar{m}_{ij} \mathbf{x}_{ij}^t \tilde{\mathbf{g}}_{ij}^{t+1}. \end{aligned} \quad (3.19)$$

Here, $\mathbf{A}_{ij}$ denotes the matrix block corresponding to the entry of particles i and j , while $\mathbf{A}_{ii}$ denotes the diagonal matrix element and $\mathbf{I}_3$ the 3x3 identity matrix. The constructed linear system is symmetric positive definite if all particles have equal mass $m = m_i$, which allows us to use the iterative matrix-free Conjugate Gradient (CG) method to find the solution. Indeed, we only perform simulations with uniform particle masses. It would, however, be possible to use a different iterative linear solver, such as BiCGStab, to solve the non-symmetric system for non-uniform particle masses.

The linear system could also be solved as three smaller systems, as the individual velocity components do not depend on each other. However, this can only be done when solving implicit surface tension in isolation and is no longer possible when strong coupling with viscosity is desired, see Section 3.2.4.

We use a matrix-free CG solver, since it is more efficient both in terms of memory and computation time, as the linear system never actually has to be fully assembled. We also use the previous change in velocity to compute a prediction to warmstart the linear solver

$$\mathbf{v}_{\text{pred}} = \mathbf{v}^t + (\mathbf{v}^t - \mathbf{v}^{t-1}). \quad (3.20)$$

We have found this to significantly reduce computation time by decreasing the number of required solver iterations. We have further implemented and tested solver acceleration using a block-diagonal preconditioner. However, we have not observed any significant improvements, although this may be a possible direction for further study.

3.2.3 Adhesion

We have formulated an implicit linear system that allows us to compute cohesive surface tension forces within the fluid itself. We extend this by defining an additional force which allows us to describe the surface tension properties between a fluid and a rigid object in contact. To achieve this we use an analogous formulation as for the previous cohesion model and introduce an adhesion coefficient σ^b , which defines how strongly the fluid is drawn to the rigid object. We further assume the usage of the *Volume Maps* boundary handling method [BENDER et al. 2019]. Within this method, boundaries are simply defined by the volume V_i^b occupied within the compact support of particle i , the closest point on the boundary $\mathbf{x}_i^b$, and the velocity of the boundary $\mathbf{v}_i^b$. All other properties, such as rest density, are

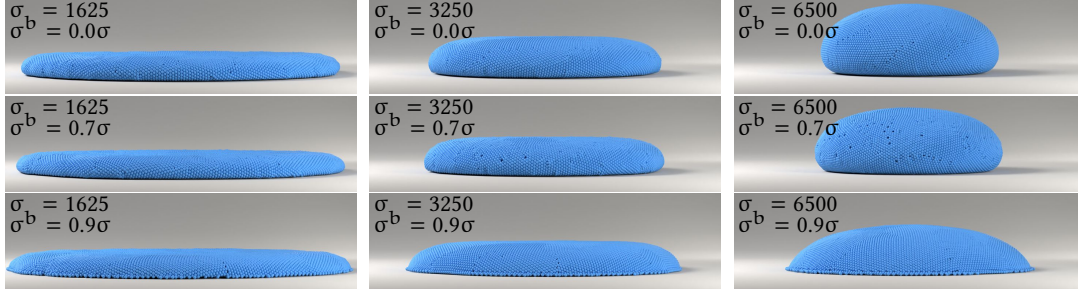


Figure 3.3. Effect of droplet adhesion to a flat plane when using different cohesion and adhesion parameters. The columns from left to right show increasing surface tension coefficients $\sigma = [1625, 3250, 6500] \text{N m}^{-1}$, while the rows from top to bottom show increasing adhesion coefficients $\sigma^b = [0\sigma, 0.7\sigma, 0.9\sigma] \text{N m}^{-1}$ as a fraction of the surface tension coefficient.

assumed to be identical to the fluid properties. The system in Equation (3.19) can easily be extended to incorporate adhesion

$$\begin{aligned} \mathbf{A}_{ii} &= \mathbf{I}_3 \frac{\Delta t}{m_i} \left(\sum_{b \in \mathcal{N}_i^b} \bar{m}_i^b \sigma^b \tilde{g}_i^{b,t+1} + \sigma \sum_{j \in \mathcal{N}_i} \bar{m}_{ij} \tilde{g}_{ij}^{t+1} \right), \\ \mathbf{b}_i &= -\frac{\Delta t}{m_i} \left(\sum_{b \in \mathcal{N}_i^b} \bar{m}_i^b \sigma^b \left(\mathbf{x}_i^t - \mathbf{x}_i^{b,t} - \Delta t \mathbf{v}_i^{b,t} \right) \tilde{g}_i^{b,t+1} \right. \\ &\quad \left. + \sigma \sum_{j \in \mathcal{N}_i} \bar{m}_{ij} \mathbf{x}_{ij}^t \tilde{g}_{ij}^{t+1} \right), \end{aligned} \quad (3.21)$$

where $b \in \mathcal{N}_i^b$ denotes all neighboring boundaries of particle i , and $\bar{m}_i^b = \frac{1}{2}(m_i + \rho_0 V_i^b)$ denotes the averaged mass. Here V_i^b corresponds to the volume occupied by the boundary in the compact support of particle i , obtained from the volume maps approach by BENDER et al. [2019]. Note that $\mathbf{A}_{ij}$ remains unchanged and that the linear system remains symmetric for uniform particle masses. The term $\tilde{g}_i^{b,t+1}$ looks slightly different,

$$\begin{aligned} \tilde{g}_i^{b,t+1} &= \frac{W_i^{b,t}}{\frac{1}{2}(\rho_i^t + \rho_0)} + \Delta t \left(\frac{\nabla_i W_i^{b,t} \cdot \mathbf{v}_i^t - \nabla_i W_i^{b,t} \cdot \mathbf{v}_i^{b,t}}{\frac{1}{2}(\rho_i^t + \rho_0)} \right. \\ &\quad \left. - \frac{W_i^{b,t}}{2} \left(\frac{\sum_{k \in \mathcal{N}_i} m_k (\mathbf{v}_i^t - \mathbf{v}_k^t) \cdot \nabla_i W_{ik}^t}{(\frac{1}{2}(\rho_i^t + \rho_0))^2} \right) \right). \end{aligned} \quad (3.22)$$

Here the term corresponding to the density gradient of the boundary is dropped, as the density of the boundary volume is assumed to be constant and equal to the rest density. While we show only the derivation of the adhesion terms for Volume Maps boundary handling, the formulation can be adapted easily to other implicit boundary representations (e.g., [KOSCHIER and BENDER 2017]) or explicit boundary representations (e.g., [AKINCI et al. 2012]).

The effect of combining different adhesion coefficients with different cohesion coefficients is shown in Figure 3.3. This figure shows that the droplet shape becomes more spherical as the surface tension coefficient increases. Increasing the adhesion coefficient makes the droplet flatter and spread out further across the plane, while also changing the contact angle between the droplet and plane.

3.2.4 Implicit Viscosity

Having obtained an implicit linear system containing both cohesive and adhesive surface tension effects, we propose to make one final modification. In the following, we introduce strong coupling of our

surface tension force with the implicit viscosity force proposed by WEILER et al. [2018] by solving both in a single linear system. In contrast to weak coupling where successive solves of the systems can be problematic for convergence, we expect strong coupling to yield better results. In practice we observed stability benefits as well as performance benefits since fewer overall CG iterations were required.

For low-viscosity fluids, the weakly coupled standard explicit viscosity model performs reasonably well. However, compared to the following strong coupling approach when using surface tension, using an explicit viscosity model is not actually more computationally efficient. This is due to the very similar matrix structure of our implicit surface tension model and the implicit viscosity model of WEILER et al. [2018].

Strongly coupling both effects can be accomplished by simply solving both linear systems simultaneously, i.e.

$$\frac{\mathbf{v}_i^{t+1} - \mathbf{v}_i^t}{\Delta t} = \mathbf{f}_{i,st}^{t+1} + \mathbf{f}_{i,visc}^{t+1}. \quad (3.23)$$

Because the implicit viscosity method by WEILER et al. [2018] is given in very similar notation, we extend the linear system in Equation (3.18) by the contribution of viscosity

$$(\mathbf{I} + \Delta t(\mathbf{A}_{st} + \mathbf{A}_{visc})) \mathbf{v}^{t+1} = \mathbf{v}^t + \mathbf{b}_{st}, \quad (3.24)$$

where the *st* subscript denotes the contributions from surface tension given by Equation (3.19) and Equation (3.21). The *visc* subscript denotes the contribution from viscosity forces

$$\mathbf{A}_{visc,ij} = 2(d+2) \frac{\mu \bar{m}_{ij}}{\rho_i^t \rho_j^t} \frac{\nabla_i W_{ij}^t \mathbf{x}_{ij}^{t \top}}{\|\mathbf{x}_{ij}^t\|^2 + 0.01h^2}, \quad \mathbf{A}_{visc,ii} = - \sum_{j \in \mathcal{N}_i} \mathbf{A}_{visc,ij}, \quad (3.25)$$

where μ is the dynamic viscosity coefficient, d the number of spatial dimensions and h the smoothing length of kernel function W . The kernel for viscosity is the standard SPH cubic kernel $W_{ij} = W^{\text{cubic}}(\mathbf{x}_i - \mathbf{x}_j; h)$. As both linear systems are symmetric positive definite individually — under the assumption of uniform particle masses — the combined system inherits this property. Therefore, we make use of the same iterative CG solver as described previously, with the same initial guess as warmstart. The full algorithm for our implicit surface tension method is shown in Algorithm 3. For all experiments, we use a relative residual error tolerance $\text{tol} = 0.001$ and a maximum number of iterations $\text{iter}_{\max} = 100$.

In the accompanying video material¹, we show a comparison using varying viscosity parameters with a constant surface tension coefficient. Larger viscosity values result in a smooth surface yet lead to damped fluid motion. Lower viscosity values are less dissipative, yet result in more chaotic motion at the fluid surface. This effect can generally be observed for inviscid fluids using SPH and is already well documented by MONAGHAN [1989, 1992], where a proposed solution is to apply velocity smoothing in order to add coupling between velocities of neighboring particles.

3.2.5 Pressure Coupling

Due to the nature of surface tension forces acting in normal direction to the fluid interface, they are directly opposed to pressure forces. Surface tension forces lead to a minimization of the surface area of the fluid, while pressure forces counteract compression and, to a certain extent, expansion of the fluid. While there have been previous works investigating strong coupling of pressure and non-pressure forces [HOPP-HIRSCHLER and NIEKEN 2019, S. LIU et al. 2022], we have not found weak coupling of our implicit surface tension and the implicit pressure solver to cause any issues. To verify this, we extended the “DFSPH-IV” approach of S. LIU et al. [2022] to strongly couple the implicit pressure solver by interleaving it with our proposed implicit surface tension solver. The outline of

¹<https://youtu.be/xWXTQJl4pZ0>

Algorithm 3 Implicit Surface Tension with Viscosity

```

1: procedure COMPUTESURFACETENSIONWITHVISCOSITY
2:   for all particles  $i$  do
3:      $(\rho \nabla \cdot \mathbf{v})_i^t \leftarrow \text{PRECOMPUTEDIVERGENCE}$  ▷ Equation (3.16)
4:      $\mathbf{b}_i^t \leftarrow \text{COMPUTECOHESSIONADHESIONRHS}$  ▷ Equation (3.21)
5:      $\mathbf{v}_{i,\text{pred}}^t \leftarrow \text{COMPUTEWARMSTART}$  ▷ Equation (3.20)
6:   end for
7:    $\mathbf{v}_i^{t+1} \leftarrow \text{SOLVECG}(\mathbf{b}_i^t, \mathbf{v}_{i,\text{pred}}^t)$  ▷ Equation (3.24)
8:   for all particles  $i$  do
9:      $\mathbf{a}_i \leftarrow \mathbf{a}_i + \frac{\mathbf{v}_i^{t+1} - \mathbf{v}_i^t}{\Delta t}$  ▷ Add acceleration
10:  end for
11: end procedure

```

Algorithm 4 Simulation Step

```

1: procedure SIMSTEP
2:   for all particles  $i$  do
3:      $\mathcal{N}_i \leftarrow \text{FINDNEIGHBORINGPARTICLES}(i)$ 
4:   end for
5:   for all particles  $i$  do
6:      $\rho_i \leftarrow \text{COMPUTEDENSITY}(i)$ 
7:      $\mathbf{v}_i^* \leftarrow \mathbf{v}_i + \Delta t \mathbf{g}$  ▷ Add gravity
8:      $\mathbf{v}_i^* \leftarrow \mathbf{v}_i^* + \Delta t \mathbf{a}_{i,\text{np}}$  ▷ Add explicit non-pressure forces
9:   end for
10:  while  $\text{iter} < \text{iter}_{\text{max}}$  do
11:     $\mathbf{v}^* \leftarrow \text{DFSPH\_DIVERGENCEFREE}(\mathbf{v}^*)$ 
12:     $\mathbf{a}^{\text{st}} \leftarrow \text{COMPUTESURFACETENSIONWITHVISCOSITY}(\mathbf{v}^*)$  ▷ Algorithm 3
13:     $\mathbf{v}^* \leftarrow \mathbf{v}^* + \Delta t \mathbf{a}^{\text{st}}$ 
14:     $\mathbf{v}^* \leftarrow \text{DFSPH\_CONSTANTDENSITY}(\mathbf{v}^*)$ 
15:  end while
16:  for all particles  $i$  do
17:     $\mathbf{v}_i \leftarrow \mathbf{v}_i^*$  ▷ Update velocity
18:     $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$  ▷ Update positions
19:  end for
20: end procedure

```

the resulting algorithm is shown in Algorithm 4. Weakly-coupled operator splitting can be recovered when setting $\text{iter}_{\text{max}} = 1$. Results comparing explicit surface tension, weakly pressure-coupled implicit surface tension and iteration-based strong pressure-coupled implicit surface tension are presented in Section 3.3.1.

Only for very large time steps, combined with very large surface tension coefficients, did the weakly coupled iterative solvers not converge. These configurations incurred a lot of compression from the surface tension forces. The subsequent application of the pressure solver to enforce constant density would then result in very large pressure forces and could lead to instabilities. This was, however, typically easy to resolve by slightly reducing the time step size, particle radius, or both. In addition, weakly coupled pressure and non-pressure solvers remain the current state of the art for SPH simulation methods [IHMPSEN et al. 2014b, KOSCHIER et al. 2022]. We leave further and more thorough investigations of strong pressure coupling as a possible direction of future work, as the selection of time step size was not a constraining factor in any of our simulations.

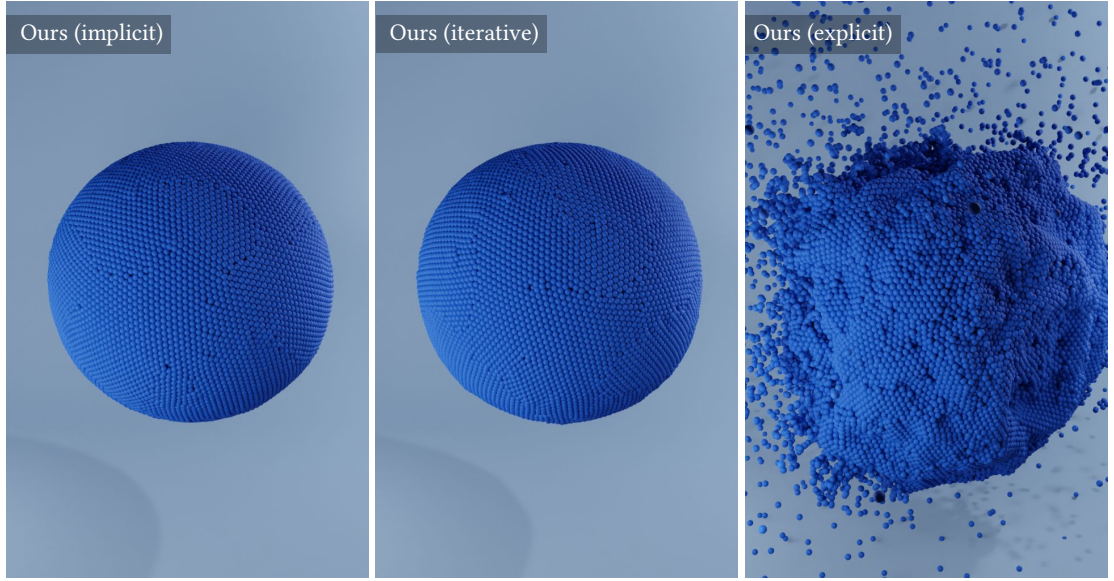


Figure 3.4. Comparison of a droplet forming in zero-gravity. Our implicit method (left) remains stable and forms a smooth sphere while the explicit version of our method (right) becomes unstable with increasing surface tension parameter. The iterative method for strong coupling with pressure forces (middle) also remains stable.

3.3 Results

In this section, we aim to show the efficacy of our implicit surface tension method using a wide variety of examples. To begin with, we investigate the different variations of our proposed method, namely explicit integration, implicit integration with weak pressure coupling and finally implicit integration with iteration-based strong pressure coupling. Subsequently, we directly compare our implicit method with weak pressure coupling to other SPH surface tension methods in order to show that we are able to match, if not even outperform existing approaches.

Afterwards, we show in more detail the capabilities of our method, especially with respect to the large surface tension coefficients enabled by our implicit method. We were not able to reproduce the majority of these examples with the other approaches due to instabilities from their explicit integration.

Unless otherwise specified, we use the *Divergence-Free SPH* pressure solver [BENDER and KOSCHIER 2017] and the implicit viscosity method of WEILER et al. [2018]. For all methods we use the implementations available in the open-source SPH framework SPlisHSPlasH [BENDER et al. 2026].

3.3.1 Sphere Stability

Visual Comparison To start with, and to further show the stability of the implicit formulation, we compare the implicit version of our surface tension method against the explicit version in a zero-gravity setting and without viscosity. For implicit surface tension, we present two versions: one utilizing weak pressure coupling, being commonly used in recent SPH publications, and another employing the iterative strong coupling scheme proposed by S. LIU et al. [2022] with $\text{iter}_{\max} = 5$ iterations (see Algorithm 4). The lack of viscosity ensures that surface tension and pressure are the only forces acting on the fluid. Furthermore, we use XSPH velocity smoothing as described by MONAGHAN [1989] with a factor of 0.5. The visual comparison is shown in Figure 3.4.

The fluid starts out as a block and the surface tension coefficient is initialized to a small value. After 2 s, this value is increased continuously — such that the block should transform into a sphere — until the explicit method becomes unstable (shown in the figure). It is clearly observable that the explicit

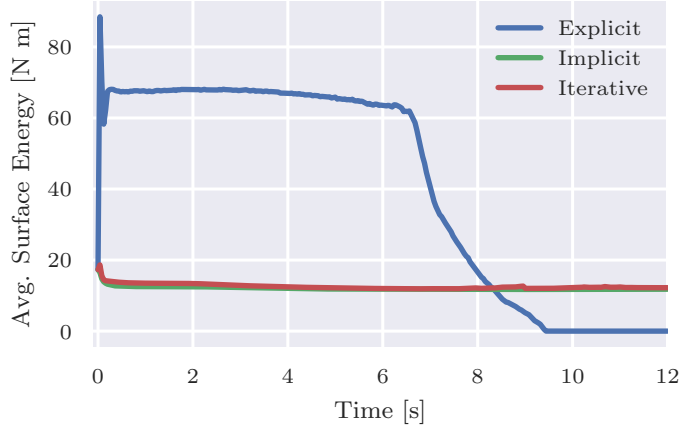


Figure 3.5. Surface energy over time for the simulations in Figure 3.4.

Table 3.1. Comparison of performance metrics for the sphere stability scene. ST abbreviates “surface tension”, while DFSPH iterations denotes the number of pressure solver iterations.

	Explicit	Implicit	Iterative
Avg. ST Time [ms]	3.95	9.34	31.92
Avg. ST & Pressure Time [ms]	12.61	44.74	154.51
$\text{iter}_{\max}$ [-]	1	1	5
Avg. DFSPH Iterations [-]	4.13	49.89	209.75
$\sigma_{t\text{start}}$ [N m^{-1}]	70000	54000	
$\sigma_{t\text{end}}$ [N m^{-1}]	140000	108000	

method forms irregular clumps at the fluid surface, while the implicit methods form a smooth and stable sphere. This disparity can be attributed to the explicit method getting “stuck” in locally optimal configurations, while solving a linear system in the implicit methods better accounts for the global shape of the fluid. Additionally, the explicit method seems to “gain” volume, which is another artifact where the outermost layer of particles detaches from the bulk of the fluid, to form a kind of “hull”.

Although both implicit methods form a stable sphere, the surface of the iteration-based strong pressure coupling appears somewhat “rougher”. We attribute this to the fact that we had to reduce the surface tension coefficient to 0.77x that of the weakly coupled simulation to obtain comparable results. This change in apparent magnitude of effects was also documented in the work of S. Liu et al. [2022].

Surface Energy The evolution over time of the average free surface energy (see HE et al. [2015] and CAHN and HILLIARD [1958]), of the three variations is shown in Figure 3.5. For the explicit solver, the surface energy oscillates in the beginning after settling into a kind of equilibrium. It slowly starts decreasing until dropping rapidly to zero as the simulation becomes unstable and particles scatter.

In contrast, both implicit methods drop quickly to a stable energy level as a sphere is formed. The energy continues to decrease slightly with increasing surface tension, as some particles in the sphere rearrange into a more optimal configuration.

Timing To conclude the comparison, we briefly outline some representative performance metrics in Table 3.1. It is clear that the explicit method is computationally the cheapest, followed by the implicit method with weak pressure coupling and finally the implicit method with strong pressure coupling. It is notable that 5 strong coupling iterations do not result in 5 times longer runtime but “only” incurs a

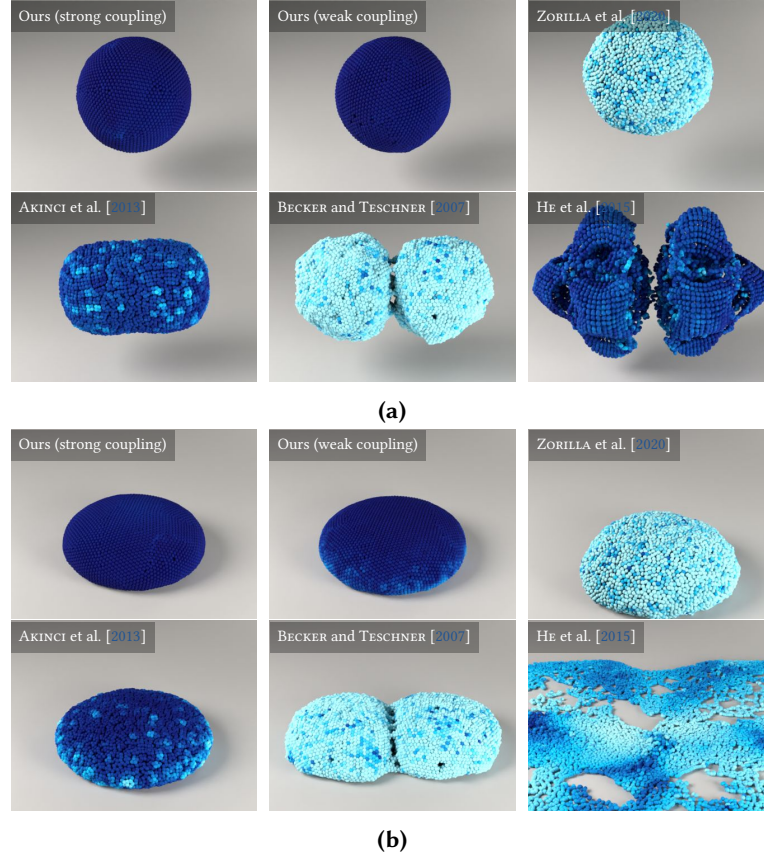


Figure 3.6. Comparison of a variety of state-of-the-art SPH surface tension methods. (a) Two fluid blocks are released in zero gravity and form a droplet. (b) After reaching a stable configuration, gravity is enabled and the droplets fall onto a flat plane without adhesion. The shading of the particles shows the magnitude of the velocity, with a lighter color indicating a larger velocity value.

factor of roughly 3.4. Also, due to the very large surface tension coefficient, a lot of pressure iterations and large pressure values are required to counteract the compression.

Summary This example shows that the proposed iteration-based strong coupling between surface tension and pressure forces is possible, but not needed to maintain stable interactions between our implicit surface tension and pressure solvers, even in the absence of viscosity. Overall, we did not observe any tangible benefits when using this particular coupling approach. Specifically, it did not allow larger time steps, did not result in more dynamic movement, and did not improve stability over weak pressure coupling. As such, we will use our implicit solver with weak pressure coupling for all following experiments, unless otherwise stated.

Further investigation of more advanced methods coupling pressure and non-pressure forces might yield different results, however, possibly making simulations of even stronger surface tension effects possible at even larger time steps.

3.3.2 Comparisons

For the following comparisons, the parameters for the pressure and viscosity solvers are kept consistent across the different surface tension models. The parameters of the other surface tension models were adjusted to the best of our abilities to yield the best possible result in each situation. Note that here, strong and weak coupling refers to coupling between surface tension and viscosity.

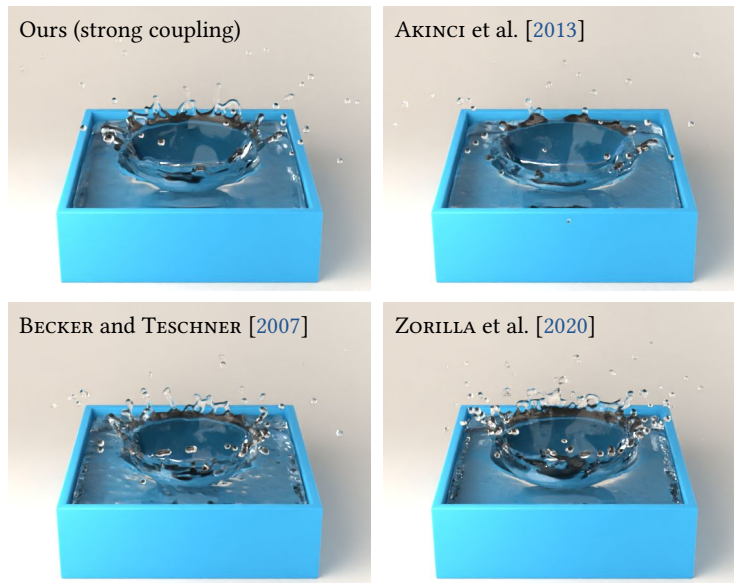


Figure 3.7. *The formation of a fluid crown when a ball of water is dropped into a container.*

Double Droplet and Impact First, two fluid blocks are released and kept in zero-gravity until they contract into a single stable fluid mass (see Figure 3.6a). In this comparison, only our method (using both strong and weak coupling of viscosity and surface tension) as well as the method of ZORILLA et al. [2020] are able to obtain an actual fluid sphere from the initial double block configuration. While the method of AKINCI et al. [2013] forms an ellipsoid, the method of BECKER and TESCHNER [2007] forms almost two distinct spheres and using the method of HE et al. [2015] we have unfortunately not been able to obtain a stable configuration as a single fluid mass. The reader is encouraged to refer to the accompanying video material² to confirm that our method also produces the most dynamic behavior during droplet formation.

Once a stable fluid mass has formed, ideally a droplet, gravity is reactivated and the droplet falls onto a flat plane (without adhesion) and should form a stable flattened droplet (see Figure 3.6b). We can observe all methods being able to form some kind of droplet, with the exception of HE et al. [2015]. Yet again, we can see that both the shape of the droplets as well as the velocity distributions on the surface are the most stable for our method. It is also possible to see that the (discouraged) weak coupling of viscosity with our method introduces some small velocity oscillations in the regions of largest curvature, while the result with our proposed strong coupling does not. The parameters of each method were adjusted such that the simulation is stable and produces droplets of similar shape on the flat plane at the end of the simulation. For a more dynamic view of this comparison, the reader is referred to the accompanying video material³.

Droplet Crown For the next comparison we show another common example for surface tension simulations, namely the droplet crown. A ball of water is dropped into a still pool, which causes a splash to form in the shape of a crown. The results of this experiment are shown in Figure 3.7. For the comparison, the method of HE et al. [2015] and our method with weak viscosity coupling are disregarded. It can be seen that all models are able to produce a fluid crown. The model of BECKER and TESCHNER [2007] and ZORILLA et al. [2020] produce more splashing individual droplets, while our method produces a slightly more cohesive crown with intricate details and fewer individual splashes. The crown for AKINCI et al. [2013] is more subdued and spread out but still visible. Note however, that

²<https://youtu.be/xWXTQJl4pZ0>

³<https://youtu.be/xWXTQJl4pZ0>

Table 3.2. Time step sizes and average timings per simulation step of the surface tension models in two of the comparison cases. All values are given in ms and the runtimes include the implicit viscosity solver. The abbreviations sc and wc denote strong coupling and weak coupling with implicit viscosity respectively, while BT denotes the method by BECKER and TESCHNER [2007].

	Double Droplet		Droplet Crown	
	Δt	Avg. Runtime	Δt	Avg. Runtime
Ours (sc)	2.0	1.90	2.0	22.00
Ours (wc)	2.0	3.06	2.0	36.57
Akinci et al.	1.0	2.91	2.0	32.60
BT	0.5	1.51	2.0	21.69
He et al.	0.5	1.92	2.0	32.25
Zorilla et al.	0.8	24.21	2.0	436.10

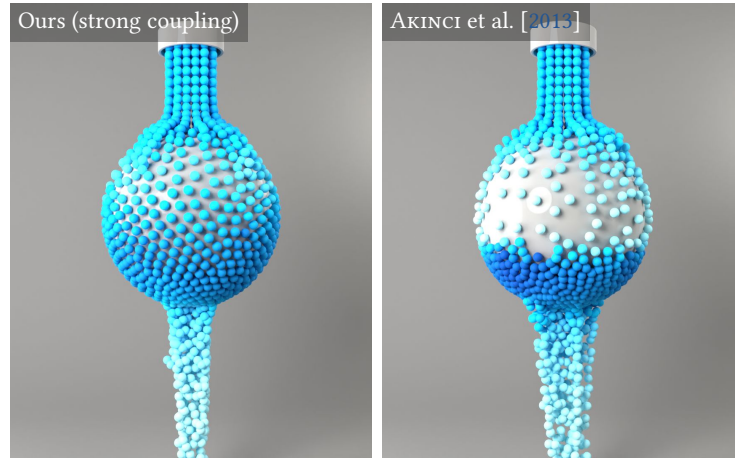


Figure 3.8. A stream of fluid is emitted over the top of a sphere, adheres to and covers the object and drips down in a uniform stream.

there is no absolute ground truth to what this droplet crown should look like. As such any one of the methods cannot be said to be better or more “realistic” than the others.

Table 3.2 shows a performance comparison of the various methods. It is evident that our method performs very competitively in terms of computational efficiency. Strong coupling especially shows significant performance improvements, which we attribute to the fact that the surface tension and the viscosity matrix have the same structure. It should further be noted that all methods have been parallelized using OpenMP. However, our method was further optimized using SIMD vectorization while the explicit methods were not. For the double droplet comparison, we were able to use larger time steps than all other explicit methods, while we used the same time step for all methods in the droplet crown example. For the droplet crown example the surface tension coefficient was chosen to be an order of magnitude smaller than for the double droplet comparison, which is why the other explicit surface tension methods also remained stable. This comparison shows that our method is already able to perform very competitively in cases with low to moderate surface tension forces, while it is able to outperform existing approaches for larger surface tension forces.

Sphere Adhesion For a final comparison we set up a scene in which fluid is emitted onto a sphere, coats the sphere due to adhesion and drips down at the bottom without separating from the object. We compare our method only with the method of AKINCI et al. [2013] because the methods of ZORILLA et al. [2020] and BECKER and TESCHNER [2007] do not implement adhesion. The comparison is shown in Figure 3.8. We can see that our method is able to cover the sphere almost perfectly uniformly, dripping down in a thin and contained stream. For the method of AKINCI et al. [2013], however, we struggled

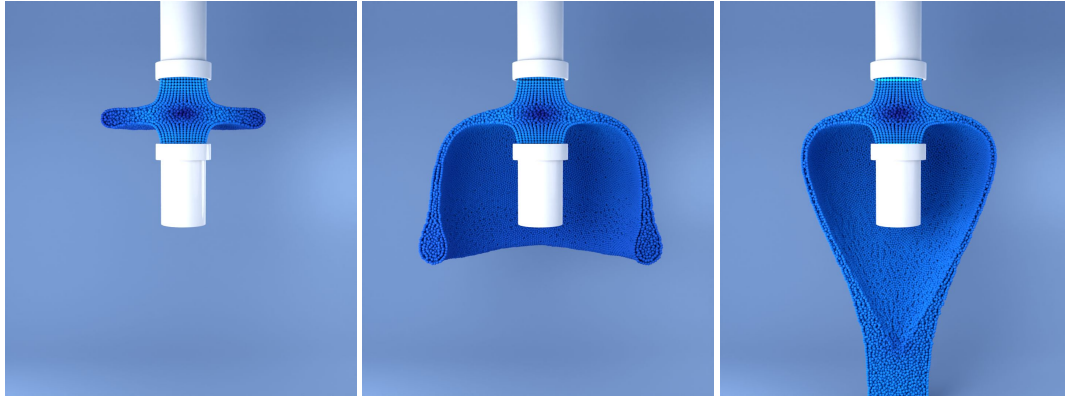


Figure 3.9. A water bell forms when the streams of two equal and opposite vertical emitters meet. The particle view is cut open by a clipping plane to visualize the interior. Parameters: $\Delta t = 0.001$ s, particle radius = 0.015 m, density = 1000 kg m^{-3} , $\nu = 0.2 \text{ m}^2 \text{ s}^{-1}$, $\sigma = 7560 \text{ N m}^{-1}$.

to find a good balance of adhesion and cohesion forces to achieve the same effect, resulting in the clustering of particles around the bottom of the sphere and a less uniform stream dripping down. In addition, the particle coloring, which indicates the velocity magnitude, shows that our method produces smooth velocity transitions, while the method of AKINCI et al. [2013] produces larger fluctuations.

3.3.3 Further Examples

In the previous subsection we compared our method to other SPH surface tension methods and were able to confirm the effectiveness and stability of our method on common surface tension benchmarks. In the following we will instead focus on testing the limits of our method by showing its full capabilities and versatility in a number of challenging scenarios.

Water-Bell First, we investigate two interesting effects which can be achieved using just two emitters. The first, called the *water bell*, is achieved by arranging two emitters vertically facing each other (see Figure 3.1 and Figure 3.9). In our example, the fluid streams meet and spread out horizontally until gravity causes them to drop down vertically. Due to the surface tension forces acting on the fluid, the vertically falling ring of fluid contracts before merging into a single stream. This results in the formation of a stable bell-like shape, which can also be observed in reality. In our experiments, we have not been able to reproduce this result with any of the other explicit SPH surface tension methods.

Fluid Chain The second effect, termed *fluid chain*, is achieved by arranging two fluid streams meeting at an angle from above (see Figure 3.10). As the streams meet and first expand into the perpendicular direction, similar to the *water bell* in Figure 3.9, a first link is formed. The second link is formed due to surface tension forces contracting the first chain link and expanding it into a plane perpendicular to the first chain link. Finally, the surface tension force balances out and contracts the fluid into a single stable stream. The size and number of chain links can be varied by changing the angle and velocity of the inflow. As with the water-bell, we have not been able to qualitatively reproduce this result with any of the other explicit SPH surface tension methods. While the method of AKINCI et al. [2013] was able to reproduce a cohesive fluid “string”, the chain links that formed were barely discernable at best.

Droplet Toy In Figure 3.1 we show the simulation of a child’s toy containing droplets. A reservoir at the top of the toy causes droplets to drip down through funnels, entering a maze of obstacles. The droplets fall through the maze and deform dynamically, bouncing against obstacles and merging with

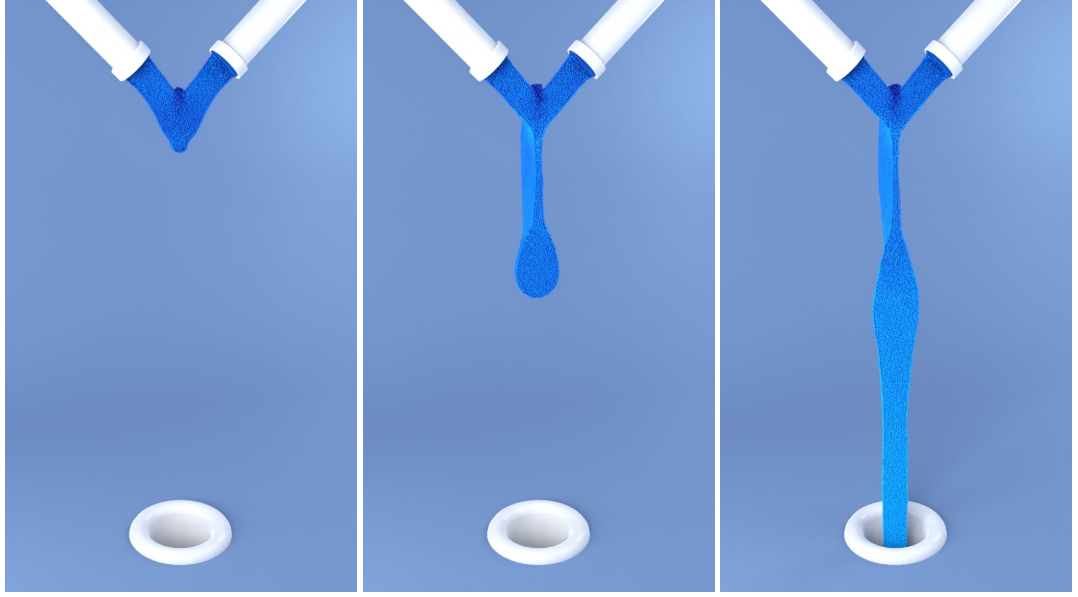


Figure 3.10. Two fluid streams causing the formation of a fluid chain. The particle view is cut open by a clipping plane to visualize the interior. Parameters: $\Delta t = 0.001$ s, particle radius = 0.01 m, density = 1000 kg m^{-3} , $\nu = 0.0125 \text{ m}^2 \text{ s}^{-1}$, $\sigma = 8400 \text{ N m}^{-1}$.

other droplets to form larger droplets. This simulation shows the ability of our method to readily simulate cases with large surface tension coefficients (the droplets maintain an almost spherical shape at any size) and interactions with boundary objects. The key parameters for this simulation are: $\Delta t = 0.001$ s, particle radius = 0.025 m, density = 1000 kg m^{-3} , $\nu = 0.002 \text{ m}^2 \text{ s}^{-1}$, $\sigma = 5 \times 10^5 \text{ N m}^{-1}$, $\sigma^b = 1.25 \times 10^5 \text{ N m}^{-1}$.

Dripping Faucet Another simulation presented in Figure 3.1 is the simulation of water dripping from a leaky faucet into a glass. The faucet has a diameter of 2 cm, while the glass is 14 cm tall. The simulation consists of 250k particles at the end of the simulation, with a diameter of 0.1 mm. Due to the slow emission of fluid, water gathers at the top of the faucet until the surface tension cannot counteract the gravitational force anymore. At this point, the water drips down in a small stream, which breaks up into smaller droplets due to surface tension. It should be noted that the faucet does not contain a so-called “aerator” at the tip, which is why this dripping behavior is more reminiscent of a garden hose water tap, rather than a kitchen sink. The key parameters for this simulation are: $\Delta t = 7 \times 10^{-5}$ s, particle radius = 5×10^{-5} m, density = 1000 kg m^{-3} , $\nu = 5 \times 10^{-6} \text{ m}^2 \text{ s}^{-1}$, $\sigma = 0.02 \text{ N m}^{-1}$, $\sigma^b = 0.01 \text{ N m}^{-1}$.

Thin-Films Using our implicit method and a careful balance of cohesion, adhesion and viscosity forces, we show the simulation of a catenoid soap film connecting two tori in Figure 3.11 on the left. The tori are first moved closer together and then apart, causing first an expansion and then a contraction of the catenoid.

Our model is also capable of simulating “soap” bubbles. The bubbles are made up of a thin soap phase, wrapped around a second much lighter “gas”-phase of particles, to emulate the internal air pressure of the bubble. For the coupling of the two phases we make use of the density-contrast multi-phase model from SOLENTHALER and PAJAROLA [2008]. A similar two-phase approach has been used by T. YANG et al. [2017] for the simulation of blowing bubbles, yet using a volume fraction-based multiphase model instead of two distinct phases. We show the simulation of two and 27 colliding bubbles in the middle and right of Figure 3.11, respectively.



Figure 3.11. *Simulation of a thin-film catenoid connecting two tori (left), two colliding bubbles (middle), and 27 colliding bubbles (right) using our method. The catenoid simulation in particular demonstrates the ability of our method to robustly handle complex interactions of surface tension, adhesion and viscosity. The inside of the bubbles are filled with air particles, while the film is made up of soap particles. Parameters: $\Delta t = 0.002$ s, particle radius = 0.004 m, density = 1000 kg m^{-3} , $\nu = 0.02 \text{ m}^2 \text{ s}^{-1}$, $\sigma = 107.52 \text{ N m}^{-1}$.*

While our model is able to keep the thin surfaces stable and intact, we have observed that the results generally lack surface detail (capillary-waves) and appear somewhat “smoothed”. We attribute this to the problem of using a 3D SPH kernel for simulating a 2D phenomenon. In order to more accurately handle these cases, we believe combining our method with a co-dimensional model [H. WANG et al. 2020] to be a promising direction of further research. Nevertheless, we believe our surface tension method being able to stably simulate thin-film behavior is an interesting and worthwhile result. This type of thin-film simulation is unthinkable with the aforementioned explicit SPH surface tension approaches.

3.4 Conclusion

Previous surface tension approaches for SPH have been mostly explicit [AKINCI et al. 2013, BECKER and TESCHNER 2007, HE et al. 2015, ZORILLA et al. 2020], requiring small time steps to simulate complex phenomena with large coefficients and intricate force interactions. Furthermore, explicit curvature-based methods often suffered from inaccurate curvature computation from the color field, while explicit cohesion-based methods have been known to suffer from increased particle clustering at the surface (see Section 3.1).

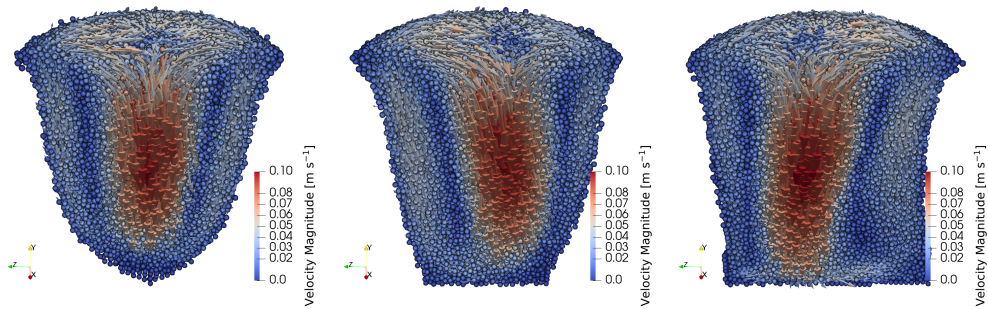
In this chapter we have presented our model for computing surface tension forces implicitly in SPH simulations. We have derived a non-linear surface tension force from inter-particle cohesion forces, have proposed a suitable linearization and solution method, and have described how to incorporate adhesion effects. In addition, we have shown how to strongly couple our approach with a state-of-the-art implicit viscosity solver [WEILER et al. 2018].

We have demonstrated the versatility of our approach on a number of well-known benchmarks, including droplet formation, droplet impact on a flat plane, fluid crown and sphere adhesion. Going beyond these simple benchmarks, we have presented a variety of more challenging scenarios which typically required a specific balance between cohesion, adhesion and viscosity forces to function as expected. This chapter therefore addresses the improved fidelity objective formulated in Chapter 1, building on SPH fundamentals from Chapter 2.

Nevertheless, we have observed our model to also have some limitations. While we are able to simulate strong surface tension effects, we sometimes needed to use the momentum-conserving and non-dissipative XSPH velocity smoothing to avoid overly chaotic motion at the surface. This, however, is well established for inviscid SPH simulations, which naturally decouple velocities of neighboring

particles [MONAGHAN 1989]. Furthermore, while we were even able to simulate thin films, these simulations were generally less predictable and harder to tune. We attribute this to the fact that we attempted to simulate a 2D phenomenon using a 3D discretization. Especially in SPH, thin films cause significant particle deficiency with large parts of the compact support being empty. We believe that combining our method with existing codimensional [H. WANG et al. 2020] or thin-film simulation [M. WANG et al. 2021] approaches could be interesting for future work. Finally, we observed some of our simulation results to appear overly smoothed, most likely due to the dissipative nature of implicit integration schemes. Possible approaches to reintroduce details would be the simulation of capillary waves [S. YANG et al. 2016], or the investigation of higher-order and less dissipative integration schemes [LÖSCHNER et al. 2020]. Together with the melt-driven application chapters, Chapters 4 and 5, further possibilities for future work are consolidated in Chapter 8.

Quantitative Evaluation of SPH in TIG Spot Welding



The Smoothed Particle Hydrodynamics method (SPH) was introduced in Chapter 2. In recent years it has been increasingly investigated for the simulation of manufacturing processes, such as arc welding [ITO et al. 2012, ITO et al. 2015, KOMEN et al. 2020, TRAUTMANN et al. 2018]. This chapter follows the overall goal defined in Chapter 1 to evaluate point-based SPH methods on melt-driven manufacturing processes against established Eulerian baselines. However, until now, SPH has not been quantitatively compared for the application to arc welding processes, in particular to assess its performance regarding calculation speed and computational accuracy. Therefore, in this chapter, the open-source implementation of the SPH method SPlisHSPlasH [BENDER et al. 2026] is extended to account for heat transfer, specific boundary conditions, and the influence of external forces (such as the Lorentz force). In order to study the applicability of our SPH method for the simulation of arc welding processes, a test case was set up modeling a simplified tungsten inert gas (TIG) spot-welding process. The test problem was solved separately by means of our SPH implementation and two different FEM models implemented in Wolfram Mathematica (WM) [O. P. SEMENOV et al. 2021] and COMSOL Multiphysics, allowing for the simulation of weld pool formation under the axisymmetric assumption.

For this comparison, the problem was idealized by using constant material properties in the simulation and assuming a flat free surface. Typically, this kind of problem is well within the strengths of Eulerian methods, whereas the SPH method is known to have difficulties with strictly bounded domains. While other works only consider the application of SPH to the entire process, we aim to first establish the accuracy of the SPH method in this idealized and bounded scenario. This is done by rigorously comparing against known Eulerian methods, whose properties have been well studied for many years. The electromagnetic Lorentz force, which is the main driving force for the convective

heat transfer in the melt pool, was calculated in advance by the WM-Code. The resulting vector field was exported and applied to the SPH simulation as an axisymmetric external force, while it was solved within COMSOL given the same boundary conditions.

We summarize the main contributions of this chapter as follows:

- The proposition of a novel SPH algorithm for the simulation of the TIG spot-welding process, which utilizes implicit time integration to significantly improve simulation stability, and enables the use of larger time steps (in line with the implicit SPH direction developed in Chapter 3).
- The construction and implementation of a fast projection scheme for the integration of surface effects for SPH fluids.
- A novel particle to grid interpolation method which, compared to existing methods such as Shepard interpolation, is able to generate smoother fields while maintaining very high interpolation accuracy. This is extended by a smoothing algorithm in order to fill in holes of the interpolated grid due to missing particles.
- The quantitative comparison and in-depth analysis of our SPH method against WM-FEM and COMSOL-FEM for TIG spot-welding simulation.

Overall, we show remarkable agreement between our SPH method and the two FEM methods, especially COMSOL, while still remaining within the margin of error exhibited by the comparison between WM-FEM and COMSOL-FEM. This result is made all the more remarkable considering that our SPH model solves the problem in full 3D while both WM and COMSOL solve the problem in 2D with rotational symmetry.

4.1 Related Work

The tungsten inert-gas (TIG) welding process is widely applied. While it has lower productivity than gas-metal-arc welding, it has advantages when high weld-seam quality and optical appearance are required. It is especially suitable for welding root passes, thin metal sheets, and materials prone to oxidation. It is therefore often applied in chemical and food processing plants.

For the present validation, the “spot-welding” variant of the process is studied for simplicity, where the welding torch position is fixed and no filler material is supplied. In addition, we consider high frequency TIG (HF-TIG), in which the electromagnetic Lorentz force, as a result of the concentration of the arc, causes the formation of significant flow patterns in the weld pool. For the usual process, in which the distribution of electric current on the surface of the liquid is typically less concentrated, the Lorentz force is less dominant. Therefore, only the weld pool formation caused by the heat flux from the arc and the influence of the Lorentz force is studied, which is sufficient for a first qualitative comparison with Eulerian methods.

To facilitate the comparison, the non-linearities of the material parameters, like latent heat and temperature-dependent thermal and electrical conductivity or viscosity were neglected, instead applying constant values for the relevant material parameters, which allowed the Lorentz force to be pre-computed. Also, since the modeling of the free surface of the weld pool requires a special treatment in the Eulerian framework, like volume of fluid (VOF) or level-set, the free surface deformation and Marangoni effect were neglected, although their modeling would pose a great advantage for the SPH method. The main focus of the present work is therefore not to present the most accurate model of the welding process. Instead, we aim to investigate the applicability of the SPH method for welding-process conditions and to quantitatively compare its performance against the common Eulerian methods under identical conditions.

4.1.1 SPH Methods in Welding Simulation

SPH has been adopted for many different applications, including the simulation of weld pool dynamics. The mesh-free nature of SPH enables the simulation of large deformations and free surface motion and coupling of many physical processes, which makes it an attractive method for many real world problems.

DAS and CLEARY [2016] use SPH in three-dimensional arc welding simulations in order to study temperature distributions, flow patterns and plastic strain in the filler material and residual thermal stresses in the workpiece.

ITO et al. [2015] perform full simulations of TIG welding using the SPH method and show results for different material properties due to different sulfur contents and evaluate the flow patterns and the shape of the weld pool.

TRAUTMANN et al. [2017] similarly perform weld pool simulations using SPH for a TIG welding process. They consider buoyancy, viscosity and surface tension as flow driving forces and the arc pressure, shear and thermal effect were parametrized using experimental studies. The penetration profiles of three different welding currents were compared to experimental results and show decent agreement.

A hybrid approach is investigated by KOMEN et al. [2020], who use an Eulerian grid and Lagrangian particles in conjunction to simulate gas metal arc welding (GMAW). Molten metal is simulated by means of SPH, while the arc plasma and gas are simulated on a grid. The methods are then weakly coupled and executed iteratively.

KOMEN et al. [2018] also simulate the GMAW process under consideration of droplet formation and compare the weld pool shapes against experimental results. To visualize results more easily, they use ensemble averaging to transfer particle data onto a regular grid.

These types of simulations are difficult to validate, as they describe very complex systems with many interacting components, and this is evident with TRAUTMANN et al. [2017] being one of the few works which attempts to validate their results using experimental data. In this chapter, we instead compare our proposed SPH method against Eulerian simulations as a proof of concept, showing that SPH can obtain very good agreement in the resulting weld-pool shapes, as well as temperature and velocity distributions. A similar study will be conducted in Chapter 5, which shows excellent performance of the SPH method when compared against the Eulerian VOF method for the simulation of droplet impacts in thermal spraying. This allows us to justify the usage of SPH to obtain physically meaningful results, also for cases that pose more difficulty for Eulerian methods, for example when considering a free surface. Our investigations confirm the quantitative accuracy of our proposed method, as well as the suitability of the SPH method as a whole for these kinds of applications.

4.1.2 Eulerian Methods in Welding Simulation

Eulerian methods are very common for the simulation of arc welding processes and have been used since the advent of computational welding simulation. A common difficulty in these methods is the calculation of free surface flows. There are several numerical approaches which are commonly used for description of these free surface flows. They can be divided into two main groups [SEMYONOV 2014]. The first group includes the so called Front Capturing Methods (FCM) in which a fixed Eulerian computational mesh is employed and a free surface is “expanded” along the volume of a certain layer. The thickness of this layer corresponds to several lengths of a computational cell. The most popular FCM are Volume of Fluid (VOF) [HAIDAR and LOWKE 1996, HERTEL et al. 2014] and Level Set (LS) [CADIOU et al. 2020] methods.

There exist also a number of other approaches where the free surface is considered as a sharp interface between two media, e.g., [MEDALE et al. 2004, 2008, NGUYEN et al. 2017]. For example, the Arbitrary-Lagrangian-Eulerian method (ALE) allows for deformation of the mesh, but it cannot solve problems with significant topological changes, like flow-splitting. However, most common arc welding processes involve melting, detachment and impingement of filler material into a weld pool. Additionally, the attachment of the electric arc to the liquid electrodes is determined by thin sheaths (anode and cathode sheaths), which are located exactly at the surface of the liquid electrodes and therefore a sharp definition of the free surface is necessary to consider these phenomena [MOKROV et al. 2021, O. SEMENOV et al. 2012]. Therefore, neither the VOF/LS approach, nor the ALE approach are completely satisfactory to accurately capture the process. Due to its strengths in modeling free surfaces as well as considering discontinuities and large topological changes and deformations, the SPH method became an interesting approach for modeling arc welding processes. However, little is yet known about the quantitative performance for the calculation of conductive/convective heat transfer compared to established Eulerian methods in the context of arc welding processes.

4.2 Method

Firstly, we want to refer to the table at the front of the document, providing a reference to unique symbols and shorthands used in this thesis. Please refer to this table when the meaning of symbols does not become clear from the immediate context.

In this section we briefly describe our SPH model and the discretization of the governing equations for incompressible weld pool dynamics. The general foundations of the SPH method are described in Chapter 2.

4.2.1 Fluid Flow

Recall from Equation (2.3), that the Navier-Stokes equations for incompressible fluid flow are given by

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}_{\text{ext}}. \quad (4.1)$$

For this work we make use of an implicit solver for both pressure [BENDER and KOSCHIER 2015] and viscosity [WEILER et al. 2018], as outlined in Section 2.2.2. The remaining forces in Equation (4.1), which are modeled as a part of the external volumetric force, are the Lorentz force and the Momentum Sink which accounts for the morphological material change during solidification and melting.

The Lorentz force is computed in advance of the simulation and interpolated bilinearly from a grid onto the particles. This is analogous to the widely used approach for calculating Lorentz forces in arc-welding process simulations, as used by CHO and NA [2021], where the stationary analytical solution developed by KOU and SUN [1985] is used to calculate the Lorentz forces.

The Momentum Sink in our arc welding simulation model is computed using the Darcy-term method, which is a common approach to model the solidification of pure metals [BRENT et al. 1988] and alloys [VOLLER et al. 1990]. The method adds an acceleration to the Navier-Stokes equation which has a strong flow inhibiting effect on the fluid, once the temperature of the fluid reaches below melting temperature T_l

$$\mathbf{a}_{\text{porosity}} = -\mathbf{v}C(1 - f_l(T)), \quad (4.2)$$

$$f_l(T) = \begin{cases} 1 & T > T_l, \\ 0 & T \leq T_l. \end{cases} \quad (4.3)$$

Algorithm 5 Fluid Solver

```

1: procedure FLUIDSOLVE
2:   DFSPH_DIVERGENCEFREE                                ▶ [BENDER and KOSCHIER 2015]
3:   SOLVEVISCOSITY                                       ▶ Equation (2.52)
4:   for all particles  $i$  do
5:     SOLVEMOMENTUMSINK( $i$ )                               ▶ Equation (4.6)
6:   end for
7:   DFSPH_CONSTANTDENSITY                                ▶ [BENDER and KOSCHIER 2015]
8:    $\Delta t \leftarrow \text{COMPUTE CFLTIMESTEP}()$ 
9:   for all particles  $i$  do
10:     $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \left( \mathbf{a}_i + \frac{\mathbf{f}_{\text{ext}}}{\rho} \right)$ 
11:     $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$ 
12:   end for
13: end procedure

```

In the above equations C denotes the morphological constant and $f_l(T)$ is the temperature-dependent liquid fraction, which is modeled using the heaviside function. The values for C are often very large, resulting in very large deceleration as soon as $T \leq T_l$, so large in fact that simulations using explicit time stepping can become unstable. We remedy this by constructing an algebraic equation which computes the acceleration using the projected velocity of the next time step as in the following

$$\mathbf{a}_{\text{porosity}} = \frac{\mathbf{v}^{t+1} - \mathbf{v}^t}{\Delta t} = -\mathbf{v}^{t+1} C(1 - f_l(T)), \quad (4.4)$$

$$\mathbf{v}^{t+1} = \mathbf{v}^t \frac{1}{1 + C\Delta t(1 - f_l(T))}. \quad (4.5)$$

The acceleration is then computed by inserting the expression for $\mathbf{v}^{t+1}$

$$\mathbf{a}_{\text{porosity}} = \frac{\mathbf{v}^t}{\Delta t} \left(\frac{1}{1 + C\Delta t(1 - f_l(T))} - 1 \right). \quad (4.6)$$

This semi-implicit formulation allows the usage of larger time steps in the simulation without causing instabilities. It should be noted that using large time steps with implicit Euler integration is known to add a significant damping effect, which diminishes the effect of the Darcy-term. This can be controlled by carefully determining a time step in the order of magnitude of C^{-1} which is typically still very small $C^{-1} \in [10^{-8}, 10^{-4}]$ s, but not necessarily as small as would be required for explicit integration.

The main fluid driving forces in our simplified model of TIG spot welding is the Lorentz force while the fluid behavior is otherwise significantly influenced by viscosity and the melting of solid material. The extensive use of implicit solvers enables us to have stable simulations when using large simulation time steps as well as large morphological constants C .

Finally, an overview of our SPH fluid solver is presented in Algorithm 5.

4.2.2 Heat Transfer

Heat transfer is computed using the Fourier equation

$$\frac{D(\rho c_p T)}{Dt} = \nabla \cdot (\lambda \nabla T) + \dot{q}''', \quad (4.7)$$

where ρ denotes the material density [kg m^{-3}], c_p the specific heat capacity [$\text{J kg}^{-1} \text{K}^{-1}$], T the temperature [K], λ the thermal conductivity [$\text{W m}^{-1} \text{K}^{-1}$] and $\dot{q}'''$ the contribution from volumetric heat sources [W m^{-3}]. For the purpose of our SPH simulation we make use of the relationship between specific enthalpy $\hat{h}$ [J kg^{-1}] and the temperature when the observed medium is incompressible

$$\hat{h}(T) = \int_0^T c_p(T) dT. \quad (4.8)$$

In this formulation c_p may also be a function of temperature, taking into account, e.g., the latent heat of melting. Resulting in the following formulation in terms of specific enthalpy

$$\widehat{h} \frac{D\rho}{Dt} + \rho \frac{D\widehat{h}}{Dt} = \nabla \cdot (\lambda \nabla T) + \dot{q}''' \quad (4.9)$$

$\frac{D\rho}{Dt} = 0$ is already enforced by the constant density component of the implicit pressure solver so that this term immediately drops out. The remaining terms are discretized using SPH and explicit Euler time integration, resulting in the following discrete equation for the fluid particle with index i

$$\rho_i \frac{\widehat{h}_i^{t+1} - \widehat{h}_i^t}{\Delta t} = \nabla \cdot (\lambda \nabla T)_i^t + \dot{q}_i^{t'''}. \quad (4.10)$$

The heat conduction term is discretized by

$$\nabla \cdot (\lambda \nabla T)_i = \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} \frac{4\lambda_i \lambda_j}{\lambda_i + \lambda_j} (T_i - T_j) \frac{\nabla_i W_{ij} \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2}, \quad (4.11)$$

as proposed by BROOKSHAW [1985] and is well documented in the literature, e.g., by TRAUTMANN et al. [2017] and DAS and CLEARY [2016]. It should be noted, that $0.01h^2$ is a small value dependent on the smoothing length h to avoid division by zero, and that the thermal conductivity $\lambda_i = \lambda(T_i)$ is generally a function of temperature.

4.2.3 Heat Sources

In this section, a surface heat source $\dot{q}_{i,\text{surface}}^t$ with constant power and Gaussian distribution is used. Many surface classification techniques are available, however the more accurate they are, the more computationally expensive they typically become. In this work it was found, that the coverage vector technique, as presented by BARECASCO et al. [2013], works quite well. It results in almost flawless classification for the initial grid configuration and remains stable throughout. The normalized coverage vector $\widehat{\mathbf{n}}_i$ for particle i is computed as

$$\mathbf{n}_i = \sum_{j \in \mathcal{N}_i} (\mathbf{x}_i - \mathbf{x}_j), \quad (4.12)$$

$$\widehat{\mathbf{n}}_i = \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|}. \quad (4.13)$$

This vector points in the direction of the lowest particle density in the particle neighborhood. The coverage angle α_i for particle i is calculated using

$$\alpha_i = \min_{j \in \mathcal{N}_i} \cos^{-1} \left(\widehat{\mathbf{n}}_i \cdot \frac{(\mathbf{x}_j - \mathbf{x}_i)}{\|\mathbf{x}_j - \mathbf{x}_i\|} \right), \quad (4.14)$$

which indicates the smallest angle between the coverage vector and any neighboring particle j . Here $\mathcal{N}_i$ contains the indices of particles in the compact support radius of particle i . Particle i is classified as a surface particle only if $\alpha_i > \alpha_0$. This is the case when there is no other particle j within a cone of α_0 around the coverage vector $\widehat{\mathbf{n}}_i$. In other words, particle i is uncovered by other particles. We have observed excellent classification results using an angle of $\alpha_0 = 35^\circ$. Surface classification using the coverage vector is displayed in Figure 4.1. The particle shaded in green is classified as a surface particle, because no other particles are present in a cone of α_0 around the coverage vector $\widehat{\mathbf{n}}_i$. The red particle is classified as an interior particle since there are two other particles in the α_0 cone around the coverage vector.

For the particles classified as surface particles, we propose the following fast projection method to compute the influence of a directed heat source. In general however, this method could also be

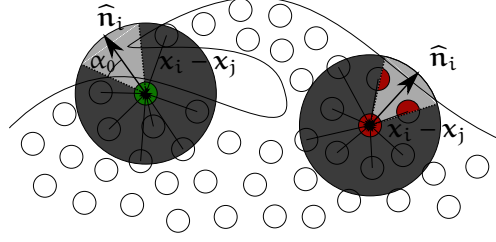


Figure 4.1. Surface particle classification using Equation (4.14). The green particle is classified as a surface particle while the red particle is classified as an interior particle.

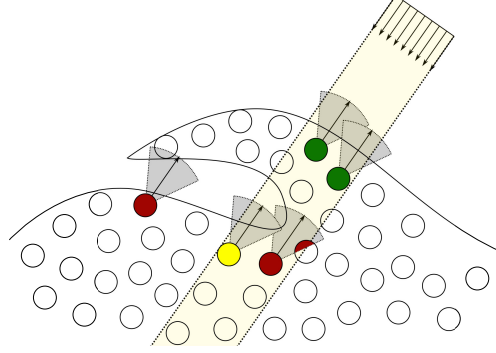


Figure 4.2. Projection of an area heat source onto previously detected surface particles. The green particles are visible, the yellow particle is discarded by a depth test, and the red particles are invisible to the heat source.

used to apply arbitrary boundary conditions at the fluid surface. First, an area heat source checks for particle visibility in its own projection region. This is shown in Figure 4.2. All surface particles which lie outside of the region in normal direction of the area source (white area) are discarded. Then all remaining surface particles are checked for other neighboring particles in a cone around the vector in normal direction to the source. The green particles are eventually classified as being visible from the surface while the red one is covered by a neighboring particle in the direction of the source. The yellow particle would be classified as visible, but can be discarded by an additional depth test.

In this work, the heat source is parametrized as a Gaussian and in order to ensure constant power of the heat source, weighted normalization is applied using precomputed Gaussian weights. This results in the expression

$$\dot{q}_{i,\text{surface}}'' = P_{\text{source}} w_i \frac{3}{\pi r_{\text{HS}}^2} \exp\left(-3 \frac{\|\mathbf{x}_i - \mathbf{x}_0\|^2}{r_{\text{HS}}^2}\right), \quad (4.15)$$

where r_{HS} is the standard deviation, $\mathbf{x}_0$ the center of the Gaussian, and w_i the weight needed for normalization of the total power to the target power P_{source} .

The subscript $i,\text{surface}$ denotes only the particles which have in the previous step been identified as surface particles and have been determined to be “visible” from the heat source, while $\dot{q}''$ denotes the heat source as Watts per unit area [W m^{-2}].

In our TIG simulation setup, all boundaries are adiabatic and do not require any special handling for heat transfer. Since heat can only be conducted within the material itself, the SPH formulation is adiabatic by construction. Finally, the enthalpy in terms of the temperature (and vice versa) is precomputed by integration of Equation (4.8). Our heat solver is outlined in Algorithm 6, while the full algorithm containing both the heat solver and fluid solver is shown in Algorithm 7.

Algorithm 6 Heat Transfer Computation

```

1: procedure COMPUTEHEATTRANSFER
2:   for all particles  $i$  do
3:     COMPUTETEMPERATURE( $i$ ) ▷ Equation (4.8)
4:   end for
5:   for all particles  $i$  do
6:     CLASSIFYSURFACE( $i$ ) ▷ Equation (4.14)
7:   end for
8:   for all particles  $i$  do
9:     APPLYHEATSOURCE( $i$ ) ▷ Equation (4.15)
10:  end for
11:  for all particles  $i$  do
12:    COMPUTEHEATCONDUCTION( $i$ ) ▷ Equation (4.11)
13:  end for
14:  for all particles  $i$  do
15:    EXPLICITEULERINTEGRATION( $i$ ) ▷ Equation (4.10)
16:  end for
17: end procedure

```

Algorithm 7 SPH Solver

```

1: procedure SOLVEFLUIDANDHEATCONDUCTION
2:    $t \leftarrow t_{\text{start}}$ 
3:   while  $t < t_{\text{end}}$  do
4:     COMPUTEHEATTRANSFER() ▷ Algorithm 6
5:     FLUIDSOLVE() ▷ Algorithm 5
6:      $t \leftarrow t + \Delta t$ 
7:   end while
8: end procedure

```

4.2.4 Particle to Grid Transfer

For scientific evaluations it can be quite cumbersome to use particle data, since many visualization techniques, i.e., line plots, contour plots, stream lines, etc. rely on the ability to interpolate the underlying data at any point in space. While this can be achieved by using the common SPH summation from Equation (2.14), this method suffers from the issue of particle deficiency at fluid boundaries, where the interpolation quality will decrease rapidly. The ability to accurately reconstruct constant functions can be restored by using normalization, as is also done by KOMEN et al. [2018], which often yields satisfactory results in practice. This normalized SPH interpolation is also often called Shepard interpolation in literature and simply normalizes Equation (2.14)

$$A(\mathbf{x}) = \frac{\sum_{j \in \mathcal{N}_x} V_j A_j W(\mathbf{x} - \mathbf{x}_j; h)}{\sum_{j \in \mathcal{N}_x} V_j W(\mathbf{x} - \mathbf{x}_j; h)}, \quad (4.16)$$

which locally restores the property of accurately representing constant functions.

Nevertheless, we propose a novel method which does not rely on the SPH kernel for interpolating particle data onto a grid. Instead, it provides an efficient and accurate way of transferring the underlying continuous field from the particle nodes to specific grid nodes. Our proposed method overlays a regular grid (in theory this could be any kind of unstructured grid) and solves a linear least squares problem for the values at the grid nodes

$$\min \sum_{i=1}^N (f(\mathbf{x}_i; \boldsymbol{\theta}) - y_i)^2, \quad (4.17)$$

where $f(\mathbf{x}_i; \boldsymbol{\theta})$ is the function which linearly interpolates the values at the nodes of the grid $\boldsymbol{\theta} = (\theta^0, \dots, \theta^M) \in \mathcal{R}^M$ to the position $\mathbf{x}_i$ of particle i and y_i is the field value at particle i . The normal

equations for this problem are given by

$$\mathbf{M}^T \mathbf{M} \boldsymbol{\theta} = \mathbf{M}^T \mathbf{y} \quad \mathbf{M}_{ij} = w_{ij}. \quad (4.18)$$

The matrix $\mathbf{M} \in \mathcal{R}^{N \times M}$ has as many rows as there are particles and as many columns as there are grid nodes. w_{ij} denotes the linear interpolation weight of particle i from grid node j . Matrix $\mathbf{M}$ is very sparse and the linear system is solved using the matrix-free conjugate gradient solver of Eigen. $\mathbf{M}$ is further extended by a set of identity equations for empty grid cells, using a field-dependent default value for the right-hand side.

Depending on the desired grid resolution, it may be possible for gaps to appear in the interpolation grid, where no particles contribute to the interpolated value. In order to avoid this we perform a smoothing step where the values in the missing cells are filled in by using Laplacian smoothing

$$\Delta \boldsymbol{\theta} = 0. \quad (4.19)$$

Other smoothing kernels, e.g. Bi-Laplacian or Gaussian kernels, may also be used, but we have found the simple 7 point 3D Laplacian kernel to work well for our purposes. To do the smoothing step, we implemented a fast marching algorithm to select the grid nodes which are directly adjacent to grid nodes with neighboring particles. For the smoothing operation, the already computed grid values serve as boundary conditions such that gradients from the already computed fields are preserved and the gaps can be filled in with meaningful values. The actual boundary of the enclosing grid is treated using zero-gradient boundary conditions. In practice, we frequently used a grid size equal to the particle diameter and smoothed for up to a distance of 1 to 3 nodes. This yielded very good results in the regions with gaps as will be discussed later in the evaluation of Figure 4.7.

In comparison to the usual interpolation technique which uses SPH to average local information at a specific position, our method attempts to approximate the underlying field globally and continuously over all particles. In practice, we have found this to result in smoother interpolated fields while at the same time having equal or better approximation quality. A brief evaluation of our interpolation method is shown in Section 4.3.1.

4.2.5 Boundary Conditions in SPH

Boundary volumes are taken into account using the analytical approach for Volume Maps by BENDER et al. [2019]. As the name suggests, the method computes analytical volume contributions of solid objects to the SPH integral as a precomputation step, so that it can be evaluated using a simple lookup and polynomial interpolation. As discussed in the original paper, this method has the advantage of accurately representing solid objects and boundary conditions, and the resulting surface varies smoothly with position. If the boundary is sampled by particles, e.g., using the method of AKINCI et al. [2012], particle movement becomes bumpy due to the unevenness of the discretization.

Using Volume Maps, boundaries can simply be integrated into the typical SPH summation of Equation (2.14) resulting in the following equation

$$A_i = \sum_{j \in \mathcal{N}_i^f} V_j^f A_j^f W_{ij} + V_i^b A_i^b W_{ib}, \quad (4.20)$$

where the superscript f denotes contributions from fluid particles and b from boundary objects. The volume V_i^b denotes the precomputed volume occupied by the boundary in the compact support radius of particle i , while the weight W_{ib} is computed for a representative particle on the surface of the boundary. Finally, A_i^b is chosen to comply with prescribed boundary conditions, e.g., being set to a fixed value for Dirichlet boundary conditions. For our model, the Volume Maps boundaries are mainly

used to extend the fluid domain and to avoid particle deficiencies for the computation of density and pressure.

We use free slip boundaries at all surfaces, meaning that

$$\left. \frac{\partial \mathbf{v}_t}{\partial \mathbf{n}} \right|_{\partial \Omega} = 0, \quad (4.21)$$

where $\partial \Omega$ denotes the entirety of the bounding cylinder (see Figure 4.3), while $\partial/\partial \mathbf{n}$ denotes the derivative in normal direction. The velocity in tangential direction to the surface is denoted by $\mathbf{v}_t$. This does not require any special treatment in the SPH formulation. Additionally, we impose adiabatic boundary condition for the heat equation

$$\left. \frac{\partial T}{\partial \mathbf{n}} \right|_{\partial \Omega} = 0, \quad (4.22)$$

for the entire boundary, while additionally adding a source term at the top surface (line DC in Figure 4.3) of the boundary

$$\left. \dot{q}'' \right|_{DC} = \dot{q}''_{\text{surface}}, \quad (4.23)$$

which is computed using the heat source from Equation (4.15). The adiabatic condition is also satisfied by construction and the method described in Section 4.2.3 is used to apply the heat sources.

4.2.6 2D-FEM Model

Model of electromagnetic processes in COMSOL

For the magnetostatic simulation by means of COMSOL Multiphysics we use the so called $\mathbf{A}_m - \varphi$ formulation which gives equations for the magnetic vector potential $\mathbf{A}_m$ and the scalar electric potential φ

$$\Delta \mathbf{A}_m = -\mu_0 \mathbf{J}, \quad (4.24)$$

$$\mathbf{B} = \nabla \times \mathbf{A}_m, \quad (4.25)$$

$$\nabla \cdot (\sigma_e \nabla \varphi) = 0. \quad (4.26)$$

Here $\mathbf{J}$ and $\mathbf{B}$ are the current density and magnetic induction respectively, μ_0 is the permeability of vacuum and σ_e is the electrical conductivity. The volumetric Lorentz force is governed by

$$\mathbf{F}_L = \mathbf{J} \times \mathbf{B}. \quad (4.27)$$

The governing equations, Equation (4.24) and Equation (4.26), are subject to the boundary conditions

$$\mathbf{n} \times \mathbf{A}_m \Big|_{AB \cup BC \cup CD} = 0, \quad (4.28)$$

$$\left. \frac{\partial \varphi}{\partial \mathbf{r}} \right|_{BC} = 0, \quad (4.29)$$

$$\varphi \Big|_{AB} = 0, \quad (4.30)$$

$$\sigma_e \left. \frac{\partial \varphi}{\partial z} \right|_{CD} = J_n, \quad (4.31)$$

where J_n denotes the current density distribution along surface CD.

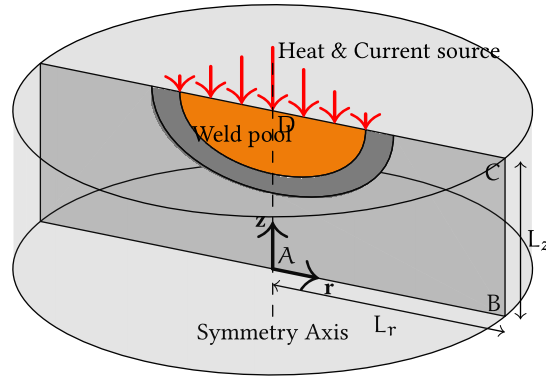


Figure 4.3. Computational domain for both SPH and FEM simulations. The FEM methods solve the problem in rotational symmetry in the ABCD plane, while SPH considers the full cylinder by explicit rotation of ABCD around the denoted symmetry axis.

Model of electromagnetic processes in WM

The model of electromagnetic processes is based on the stationary magnetic field diffusion equation [DAVIDSON 2001] which in the case of axial symmetry reads

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial B_\theta}{\partial r} \right) + \frac{\partial^2 B_\theta}{\partial z^2} - \frac{B_\theta}{r^2} = 0, \quad (4.32)$$

where B_θ is an azimuthal component of magnetic induction. From Ampere's law

$$\mu_0 \mathbf{J} = \nabla \times \mathbf{B}, \quad (4.33)$$

the next boundary conditions for Equation (4.32) can be readily obtained

$$B_\theta \Big|_{CD} = \frac{\mu_0}{r} \int_0^r J_n(s) s ds, \quad (4.34)$$

$$B_\theta \Big|_{BC} = \frac{\mu_0 I_w}{2\pi L_r}, \quad (4.35)$$

$$\frac{\partial B_\theta}{\partial z} \Big|_{AB} = 0. \quad (4.36)$$

Here L_r is the radius of computational domain (see Figure 4.3), while I_w denotes the welding current. Under the given magnetic field, the current density and Lorentz force are obtained from Equation (4.33) and Equation (4.27), respectively.

Numerical procedure in COMSOL Multiphysics

The governing equations in COMSOL are solved using the Finite Element Method (FEM). For temporal discretization the BDF (backward differentiation formula) was applied, with default settings of COMSOL 5.6. The spatial discretization can be seen in Figure 4.4. Here the nodes used quadratic shape functions for temperature, velocity, magnetic strength and for scalar and vector potential. The shape functions utilized for the solution of the pressure were linear. The sparse direct solver PARDISO was selected to solve the resulting system of linear equations.

Numerical procedure in WM

The governing equations are solved using the Galerkin Finite Element Method (FEM) [O. ZIENKIEWICZ et al. 2014]. A characteristic-based scheme [O. C. ZIENKIEWICZ and CODINA 1995] was used in temporal discretization of Equation (4.1) and Equation (4.7) along with FEM approximation in space, see Figure 4.5.

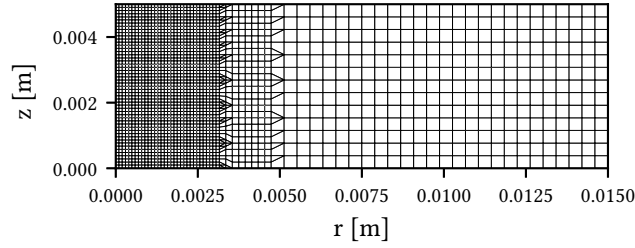


Figure 4.4. COMSOL mesh as mixture of quadrilaterals and triangles.

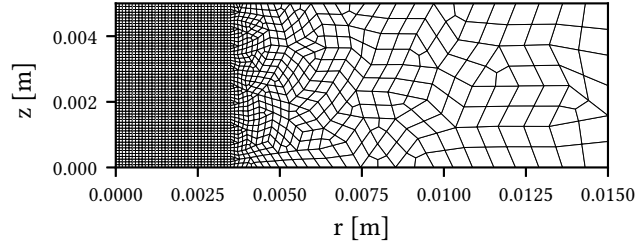


Figure 4.5. FEM mesh as Quad9 quadrilaterals. The Navier-Stokes equations are solved in the region of interest $\Omega = \{0 \text{ mm} < r < 3.5 \text{ mm}, 0 \text{ mm} < z < 5 \text{ mm}\}$ while heat transport is solved across the entire domain.

Nine-node biquadratic shape functions are employed for velocity, temperature and magnetic fields whereas pressure is approximated by four node bilinear shape functions. Such a choice of mixed interpolation for velocity and pressure fields provides stability of the numerical procedure [O. ZIENKIEWICZ et al. 2014]. The sparse direct solver PARDISO [SCHENK and GÄRTNER 2004] was employed to solve the resulting systems of linear equations. All numerical algorithms were implemented in the Wolfram Mathematica Language (WM).

4.3 Results

4.3.1 Grid Interpolation

As noted before, while all subsequent evaluations could technically be done on particle data alone, many existing evaluation functions and programs, e.g., ParaView and Tecplot, are tailored to grid structures and unstructured meshes. This makes regular and unstructured meshes very appealing for quantitatively evaluating and comparing scientific data. To this end, we compare the interpolation accuracy using naïve SPH interpolation, Equation (2.14), Shepard-corrected SPH interpolation, Equation (4.16), and our proposed least-squares linear interpolation, Equation (4.17).

Figure 4.6 shows a comparison of the different interpolation approaches. Each method has been used to compute the particle to grid transfer of the same particle data. The error is computed by interpolating these grids back to the original particle positions and computing the mean absolute error. The underlying particle data is the velocity field of a randomly selected frame of a fluid dynamics simulation in motion, but the same trend is visible regardless of which field is approximated. The regular grid spacing d_{spacing} is determined by a multiple n_{grid} of the SPH particle radius r_{part}

$$d_{\text{spacing}} = n_{\text{grid}} \cdot r_{\text{part}}. \quad (4.37)$$

The resolution of the grid shown in Figure 4.6 is then computed as the inverse of this factor, n_{grid}^{-1} , i.e., a higher resolution corresponds to a smaller grid spacing d_{spacing} and thus a larger number of total grid nodes.

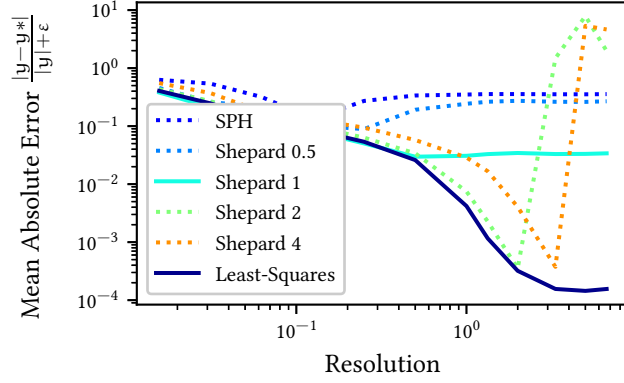


Figure 4.6. Comparison of particle-to-grid interpolation techniques.

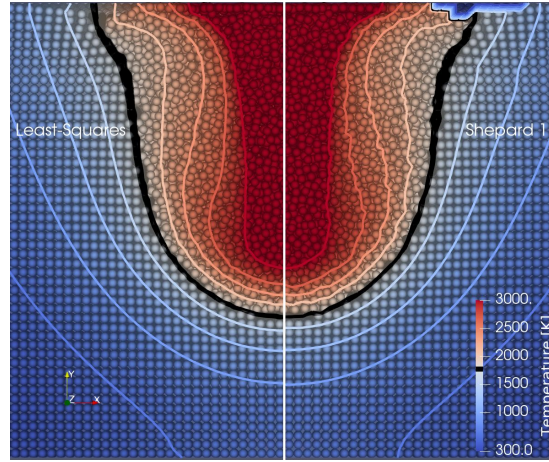


Figure 4.7. A comparison of our proposed least-squares interpolation method using smoothing of one surrounding grid node (left) and Shepard interpolation (right) without smoothing.

Shepard interpolation was evaluated using a varying factor for the SPH compact support radius depending on the factor for grid spacing $h_{\text{shepard}} = d_{\text{spacing}} \cdot n_{\text{shepard}}$. It can be seen that both our proposed least-squares interpolation as well as Shepard interpolation using the grid spacing with a factor of 1 and 2 as compact support radius perform the best in this comparison. Naïve SPH interpolation performs best when the compact support radius is equal to the compact support used in the actual simulation. It also suffers from worse interpolation quality due to particle deficiency at the surface, leading to large errors in the interpolation.

For finer resolutions it can be seen that the least-squares interpolation performs best in all cases, while the Shepard interpolation with a factor of 2 and subsequently a factor of 4 have a comparable error when a grid spacing of $0.5r_{\text{part}}$ and $0.25r_{\text{part}}$ is used respectively. From this comparison we conclude that both Shepard interpolation, as well as our proposed least-squares interpolation can work very well for particle data. The interpolation quality of our least-squares approach becomes more apparent when the grid-resolution increases further still.

Another advantage of our interpolation can be seen in Figure 4.7, where the particle to grid transfer of the temperature field using two different interpolation methods is shown. The left half shows the interpolation of our proposed least-squares method and smoothing for up to one neighboring grid node, and the right side shows Shepard interpolation.

From this qualitative comparison it is possible to see that while the values are almost identical in the entire domain, the least squares interpolation results in a much smoother field. This is especially

Table 4.1. *Material parameters of high-alloy steel*

Property	Unit	Value
Density ρ	kg m^{-3}	8100
Surface tension σ	N m^{-1}	disabled
Viscosity μ	Pa s	0.004
Thermal conductivity λ	$\text{W m}^{-1} \text{K}^{-1}$	22.9
Electrical conductivity σ_e	S m^{-1}	10^6
Heat capacity c_p	$\text{J kg}^{-1} \text{K}^{-1}$	800
Melting temperature T_l	K	1773
C from Equation (4.6)	s^{-1}	$3.7 \cdot 10^4$

Table 4.2. *Domain parameters*

Property	Unit	Value
Radius of the cylinder L_r	m	$15 \cdot 10^{-3}$
Height of the cylinder L_z	m	$5 \cdot 10^{-3}$
Initial temperature	K	300
Heat source center $\mathbf{x}_0$	m	$(0, 6 \cdot 10^{-3}, 0)$
Heat source dist. radius r_{HS}	m	$2.7 \cdot 10^{-3}$
Heat source power P_{source}	W	1213.8
Total current I_w	A	140
External forces $\mathbf{f}_{\text{ext}}$	N m^{-3}	See Figure 4.8

visible at the temperature contour lines within the black border. In this figure the advantages of our smoothing approach can also be observed. While the top right of the Shepard interpolation shows gaps in the interpolation due to missing particles, the left side shows that this part (top left) can be filled in with very plausible values using the aforementioned Laplacian smoothing. While the filled in values should be treated with care, this kind of filling in allows consistent reconstruction of fixed domains from particle data.

For all following particle to grid interpolations we use our least-squares interpolation with a grid resolution of twice the particle radius, because this is the same resolution as for the WM and COMSOL simulations. In addition, we apply a smoothing of up to 3 grid nodes, although only for times close to the end of the simulation is the full smoothing length actually needed. At this resolution it can be expected to incur a mean absolute error in the interpolated data of roughly 4 % for scalar values.

4.3.2 Simulation parameters

To allow for a first validation, the relevant material parameters have been simplified to constants, see Table 4.1. It should be mentioned that the morphological constant C is usually given without division by density, where it evaluates to $C = 3 \times 10^8 \text{ kg m}^{-3} \text{ s}^{-1}$.

The properties are applied homogeneously over the calculation domain, which consists of a rotationally symmetrical cylinder, as seen in Figure 4.3. The details of the computational domain can be seen in Table 4.2. It is important to note that the SPH domain is realized in full 3D, while the FEM domain considers only a 2D plane with rotational symmetry around the symmetry axis AD.

The external Lorentz force field due to electromagnetic forces is computed in advance of the fluid-flow simulation, according to Equation (4.27) in order to be able to use the same external forces for both the SPH and WM simulations. The magnetic strength field is obtained from the solution of Equation (4.32) by the WM solver. From these the Lorentz forces $\mathbf{F}_L$ are computed using Equation (4.27), and are then included as external volumetric forces $\mathbf{f}_{\text{ext}}$ in the Navier-Stokes equations, Equation (4.1),

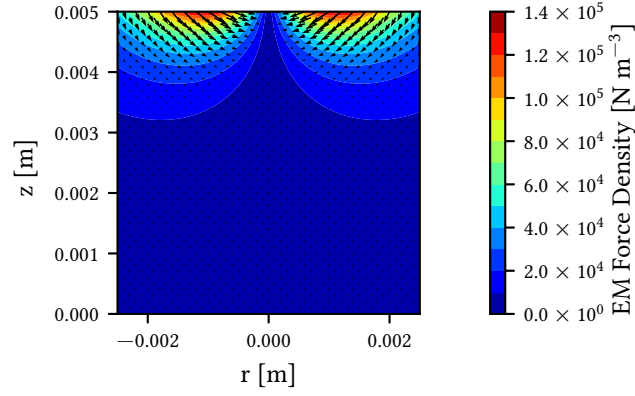


Figure 4.8. Lorentz force in the region of interest with the direction indicated as an arrow and the magnitude indicated as the underlying color.

by a rotation of the 2D plane around the symmetry axis

$$\mathbf{f}_{\text{ext}} = \mathbf{F}_L. \quad (4.38)$$

The vector field of the forces is calculated from the applied electric current density $J_n(s)$ in Equation (4.34), distributed as a Gaussian along the DC boundary (see Figure 4.3) in the same way that the heat source distribution is defined in Equation (4.15), but with the welding current I_w

$$J_n(s) = I_w \frac{3}{\pi r_{\text{HS}}^2} \exp\left(-3 \frac{\|\mathbf{x}_i - \mathbf{x}_0\|^2}{r_{\text{HS}}^2}\right). \quad (4.39)$$

Figure 4.8 shows the Lorentz force in the region of interest. The Lorentz force for COMSOL was calculated within the software according to Section 4.2.6 and was found to be nearly identical in the region of interest, showing only small deviations farther away due to differences in the system of equations (see Section 4.2.6 and Section 4.2.6).

The convergence of the simulation was achieved using the numerical parameters listed in Table 4.3 and Table 4.4. The SPH particle diameter is chosen to be equivalent to the element size in the region of interest for the WM and COMSOL simulation. The WM method and the SPH method use adaptive time integration using the CFL criterion, with identical maximum time step sizes but different factors. The WM method makes use of both second-order consistent implicit time integration as well as higher order finite-element basis functions. As such, the CFL factor in SPH was chosen heuristically to be smaller, since only a first-order implicit time integration scheme with operator splitting is used. For COMSOL, the backwards differentiation formula (BDF) with default parameters was used for temporal discretization.

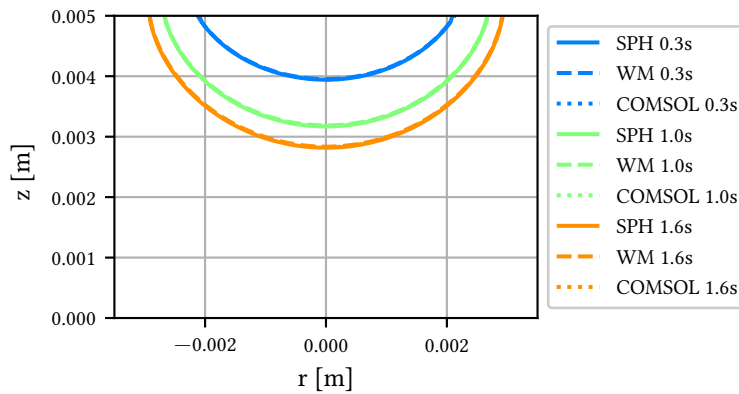
The SPH simulation was run on a compute node with 36 cores and took 86 h for 2.5 s of simulated time. The WM simulation was run on a 4 core machine and completed in roughly 9 h. Finally, the COMSOL simulation was run on a compute node with 48 cores and completed in approximately 20 min. It should be strongly noted that these computations are not directly comparable. For one, the simulations were run on very different hardware. Additionally, the SPH method was run in full 3D which has significantly more degrees of freedom than the rotationally symmetric WM and COMSOL simulations. In the SPH simulation we use roughly 3.5 million particles with 3 degrees of freedom for all vector quantities, while the meshes of the COMSOL and WM simulation have roughly 10k grid nodes with 2 degrees of freedom for all vector quantities. As such the number of degrees of freedom of the SPH simulation is ≈ 500 times larger than for the other simulation methods. In our opinion, this shows good performance and scaling of our SPH method.

Table 4.3. Numerical parameters of SPH

Property	Unit	Value
SPH particle radius	m	$5 \cdot 10^{-5}$
Maximum time step	s	$1 \cdot 10^{-4}$
CFL criterion	-	0.1

Table 4.4. Numerical parameters of WM

Property	Unit	Value
Element size in region of interest	m	$10 \cdot 10^{-5}$
Maximum time step	s	$1 \cdot 10^{-4}$
CFL criterion	-	0.5

**Figure 4.9.** Melting temperature (1773 K) iso-contours at different times for SPH, WM and COMSOL without Lorentz force $\mathbf{f}_{\text{ext}} = 0 \text{ N m}^{-3}$, i.e., pure heat conduction.

4.3.3 Simulation results

Conductive Heat Transfer

Before the quantitative comparison of forced convective and conductive heat transfer is performed, a comparison without the external forces $\mathbf{f}_{\text{ext}}$ was made, i.e., without any fluid movement and therefore no convection. As can be seen from the shape of the melting temperature iso-contours in Figure 4.9, the calculation for the conductive heat transfer was in perfect agreement between SPH, WM and COMSOL for all time steps. Therefore, we consider the purely conductive part of the heat transfer module in SPH as validated.

Forced Convective and Conductive Heat Transfer

Quantitative Comparison The dimensions of the melt-pool over the course of the simulations are shown in Figure 4.10, for the case when the external forces were set to the Lorentz force as shown in Equation (4.38). The weld pool dimensions are computed by solving a non-linear algebraic equation for the position of the melting front. Overall both characteristic melt-pool dimensions, the depth and the width, show very good agreement between all three methods and exhibit a similar development over time. The depth of the weld pool for the SPH and COMSOL simulation match almost perfectly across all time steps. During the first $t < 0.5 \text{ s}$ the WM weld pool is deeper, but afterwards agreement between all three methods is again observed for $t > 0.5 \text{ s}$. Starting at roughly $t > 1.25 \text{ s}$ the development of the depth increases for the WM solver, reaching full penetration about $\Delta t \sim 0.1 \text{ s}$ earlier than the SPH method, while the COMSOL solution is a bit slower still, reaching full penetration $\Delta t \sim 0.03 \text{ s}$ later.

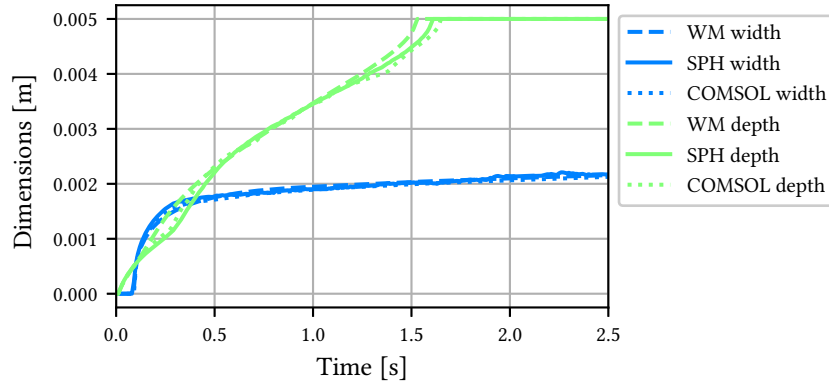


Figure 4.10. Evolution over time of the melt-pool dimensions, depth and width, at 4.5 mm.

Also the depth of the COMSOL solution does not seem to develop entirely smoothly, displaying minor kinks at $t = 1.35$ s and $t = 1.5$ s.

As the interpolation of the SPH solution onto a grid has limitations close to the boundaries, because fewer particles are available for interpolation, the width was not computed exactly on the boundary, but 0.5 mm below the surface. While we use our proposed interpolation method to obtain a plausible filling-in of the gap close to the boundary, a better result is obtained when looking at the weld pool width within the domain. Here the width matches up nearly perfectly between all three solver methods, with a minor deviation for $t < 0.5$ s, where the COMSOL solution appears right between the SPH and WM solution.

In Figure 4.11 the melting temperature iso-contour can be seen at characteristic time steps of $t = 0.3$ s, $t = 1.0$ s and $t = 1.6$ s. These characteristic times were also chosen for all future comparisons. At 0.3 s the melt-pool is more shallow in SPH, compared to WM and COMSOL, with its melting contour having a very similar shape to that shown in Figure 4.9 for the same time. This suggests that the convective heat transfer has not yet started in the SPH case, which could be due to insufficient amount of particles to allow for any fluid movement. At 1.0 s we can see excellent agreement between the SPH, WM and COMSOL method, not just in terms of maximal width or depth but also in the general shape of the melt-pool. At 1.6 s the agreement between SPH and COMSOL can still be regarded as very good, while WM deviates for these solutions in the lower part of the melt-pool $z \leq 0.001$ m, as full penetration of the workpiece is already achieved for WM at this time step, whereas both SPH and COMSOL are just about to reach the bottom boundary. For the intermediate part $0.001 \text{ m} \leq z \leq 0.0045$ m, the COMSOL solution is right in between the WM and SPH solution. It should be mentioned that there are slight deviations in the top part $z > 0.0045$ m for the SPH solution, which occur due to missing particles.

To obtain a better understanding of the significant velocities, the velocity distributions at $t = 1.0$ s along the symmetry axis (z) are shown in Figure 4.12. The shape of the velocity distribution is very similar for all three solutions, while the difference between WM and COMSOL is quite comparable to the difference between COMSOL and SPH. This is especially remarkable since both WM and COMSOL are FEM solvers of the same problem. In the light of the observed differences between both FEM approaches, the disagreement of the SPH method with the FEM solvers appear to actually be within the margin of error that can be expected from the application of FEM in general. On the contrary, the overall level of agreement is especially remarkable when considering that in SPH an entirely different discretization is used and that a whole three dimensional domain is simulated. The agreement in the radial direction is very good for all three methods. For SPH the velocities fluctuate around 0 m s^{-1} , due to the three dimensional nature of the simulation. The fluctuations of the velocity for SPH may also be partly attributed to the fact that only a slice of full 3-D data is examined, where no constraints

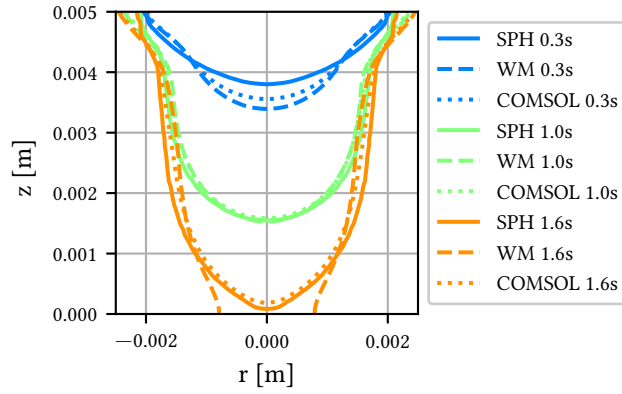


Figure 4.11. Melting temperature (1773 K) iso-contours at different times for SPH and WM, including conductive heat transfer driven by the Lorentz force.

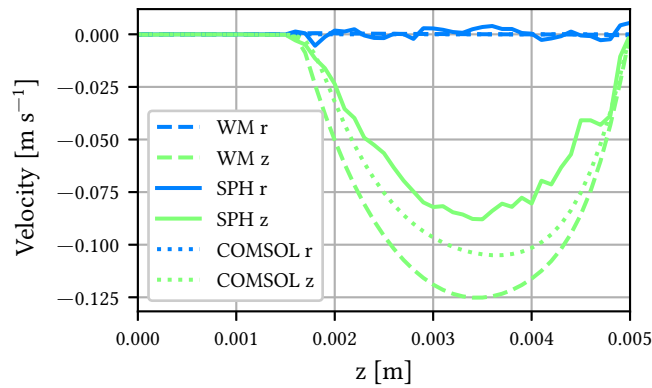


Figure 4.12. Velocity by component along the symmetry axis at $t = 1.0$ s.

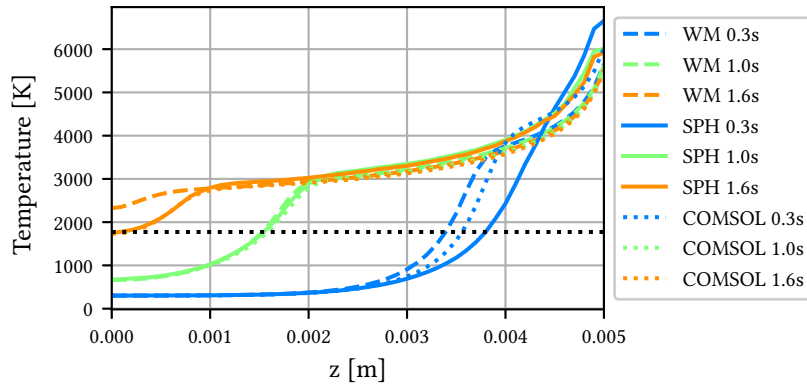


Figure 4.13. Temperature along symmetry axis at different times. The dotted black line marks the melting temperature (1773 K).

explicitly enforce strict movement within the radial plane. As such, it is only to be expected for a certain amount of statistical noise to be present in the SPH velocity components, compared to the FEM methods where the considered axis was realized as a rotational symmetry boundary, and as such does not permit movement in radial direction.

Lastly, the temperature for the chosen characteristic time steps along the symmetry axis (z) is shown in Figure 4.13. Again we can make similar observations as before. We see excellent agreement in the temperature distribution for $t = 1.0$ s between all three methods, while some deviation can be

observed for times close to the beginning $t = 0.3$ s and for times close to full penetration $t = 1.6$ s. At time $t = 0.3$ s it should be pointed out that SPH reaches a higher maximal temperature at $z = 0.005$ m and does not share the characteristic shape we observe for the other distributions, but exhibits a more uniform temperature decay as one would expect from a purely conductive heat transfer. As mentioned earlier in the description of Figure 4.11, this is most likely due to inhibited fluid movement at early times, as can also be observed later in Figure 4.14b for $t = 0.3$ s. Again, for time step $t = 0.3$ s COMSOL aligns more closely with WM, although not reaching perfect agreement. The shape of the COMSOL solution at this time seems to indicate weaker conductive heat transfer than WM, therefore most likely representing an intermediate state between the SPH and WM solution, which is an observation already made for some of the previous data, e.g., Figure 4.12.

For $t = 1.0$ s we see excellent agreement between all three solutions for the lower half of the temperature distributions ($z > 0.0025$ m), however in the upper half we observe slightly higher temperatures in SPH compared to WM and COMSOL.

At $t = 1.6$ s the largest discrepancy between the three solutions can be found at the bottom of the melt-pool. While SPH and COMSOL show very good agreement at the lower part of the axis ($z \leq 0.001$ m), the WM solution differs strongly from both. This could be, as previously mentioned, due to the effect of the lower boundary, as WM has already achieved full penetration at this time step, while SPH and COMSOL have not yet achieved it. However, for the upper part of the axis ($z > 0.001$ m), the agreement between COMSOL and WM is very good, while the temperatures calculated by SPH are slightly higher, with the deviation increasing slightly with increasing z . All distributions have the previously mentioned characteristic shape, however the WM solution consistently predicts a deeper weld pool, indicating a stronger convective heat transfer in the entire melt-pool, consistent with the observations on the velocities in Figure 4.12. Regarding the temperature profile along the height axis it can also be said that our SPH method lies well within the margin of error that can be expected between two different FEM solvers.

Qualitative Comparison Following the quantitative analysis, we now show a qualitative comparison of the SPH and COMSOL solutions for half of the cross section at times $t = 0.3$ s, $t = 1.0$ s and $t = 1.6$ s. These times were selected, as both $t = 0.3$ s as well as $t = 1.6$ s showed the greatest differences between the methods with regard to the melt-pool dimensions shown in Figure 4.10. While the deviations were strongest in comparison with the WM solution, this qualitative comparison focuses on SPH and COMSOL, where agreement of the melt-pool dimensions was best.

Over the whole duration, the time $t = 0.3$ s showed the largest disagreement between both methods, as can also be seen from Figure 4.10 and Figure 4.11. The disagreement is within the expected margin of error indicated by the comparison of the two FEM methods. Looking in more detail at Figure 4.14a and Figure 4.14b, one can see that the melt-pool in the COMSOL simulation is slightly deeper than the melt-pool in the SPH simulation, while the melt-pool radius appears comparable. It is also possible to observe the distinct deviation in the velocity field between the two simulations, which is in line with the observation of the velocity in Figure 4.12. A stable vortex seems to have already developed in the COMSOL simulation, while the SPH particles are barely in motion. We explain this by the very restrictive boundary condition in the SPH simulation. The fluid will remain motionless until there is enough space for particles to overwhelm, or circumvent, the pressure forces keeping the fluid incompressible. Since we are using an implicit pressure solver, the pressure forces will be quite large until a suitably large region of the fluid has melted, allowing for more movement while still maintaining incompressibility. The difference in velocity also explains the deviation in melt-pool shape, since at $t = 0.3$ s the SPH simulation is still transferring heat only by conduction, while the melt-pool in the FEM solutions deepens due to increasing convection.

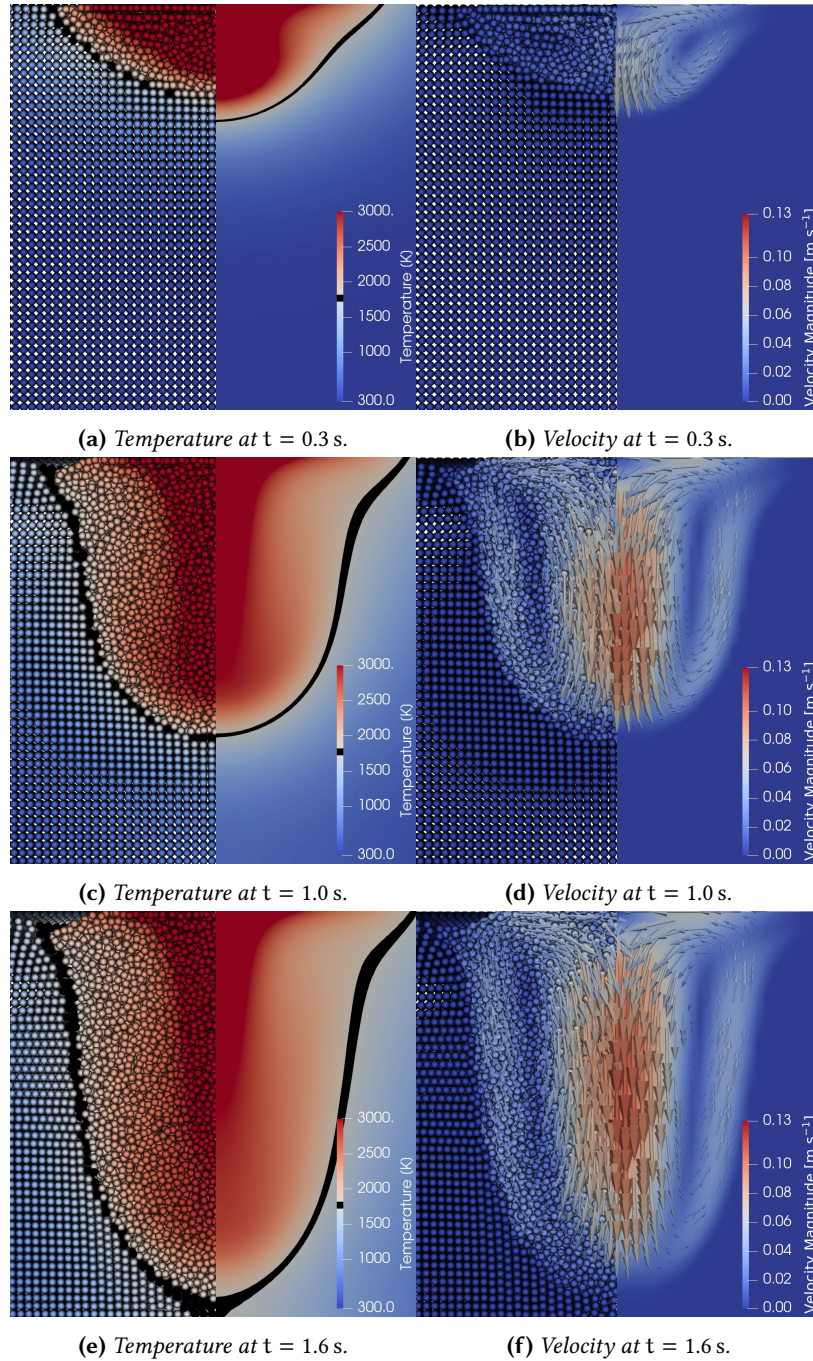


Figure 4.14. Side-by-side comparison of SPH and FEM results of temperature (left column) and velocity (right column). The SPH solution is always shown on the left within each figure while the WM solution is shown on the right.

Next, in Figure 4.14c and Figure 4.14d, we show the temperature and velocity distributions at $t = 1.0$ s for both SPH and COMSOL. Overall these results show excellent agreement, both in terms of the melt-pool dimensions (width and height) as well as the actual temperature and velocity field. The temperature fields appear almost identical, however, the melt-pool contour exhibits small deviations, especially close to the top boundary. In addition, the velocity magnitude of the SPH simulation at the symmetry axis is slightly lower than that of the COMSOL simulation. We are unsure about this deviation and can only speculate that the velocity slightly decelerated due to the increased pressure forces (slight over-pressure). Otherwise, there are no movement inhibiting forces acting on the fluid

which are not shared across both simulations, i.e., viscosity and momentum-sink. Regardless of the difference in velocity magnitude, the flow pattern is, similar to the temperature field, almost identical.

Finally, considering the simulations at $t = 1.6$ s in Figure 4.14e and Figure 4.14f we show the instance where full penetration is reached for both solvers, almost simultaneously, see also Figure 4.10. One can see that the comparison of the temperature fields shows very good agreement as well, where the general shape of the melting-temperature iso-contour is captured accurately across almost the entire region, except for a small area close to the top surface. The velocity field also exhibits the same flow pattern as at $t = 1$ s, where the velocity magnitude is again slightly lower in the SPH simulation than in the COMSOL simulation.

Overall, we can observe very good agreement between the flow patterns and temperature distributions of both simulation methods. We speculate that the observed differences at $t = 0.3$ s result from restricted motion in SPH due to high pressure forces, which would also explain the diminished velocity magnitude at subsequent time steps, but we could neither confirm nor refute this speculation in a brief investigation. While this discrepancy warrants further study, we have shown successfully that the observed flow pattern and temperature field agree very well over a long duration and retain distinct similarities when the melt-pool depth differs.

4.4 Discussion

The comparison of the results from the SPH, WM and COMSOL solvers yields intriguing results. The simplifications strongly favor the application of an Eulerian method, and one of the main goals of this comparison was to investigate whether the SPH method is able to achieve good results in such a setting. Based on the preceding evaluation, we believe this is indeed the case, as we observed very good quantitative agreement between the SPH and COMSOL methods, while also observing the same overall trends in the WM method.

Firstly, the conductive heat transfer was confirmed to work as expected, up to differences which can be accounted for by differences in spatial and temporal discretization. Secondly, the convective heat transfer seems to be in good agreement between SPH and COMSOL, as can be seen, for example, from the melting-temperature iso-contours in Figure 4.9. Convective heat transfer in WM appears to occur slightly faster. This is in line with the observations on the velocities, see Figure 4.12, that seem to exhibit some deviation for all three methods of $\approx \pm 20\%$. In the light of these deviations the excellent agreement for the melt pool depth at $0.6 \text{ s} < t < 1.3 \text{ s}$ seems somewhat surprising. Figure 4.13 seems to confirm that the convective heat transfer in downward direction is slightly slower at all times of the SPH simulations as the temperature at the top, close to the heat source, is slightly higher. Nevertheless, the temperature profiles align almost perfectly for SPH and COMSOL towards the bottom of the melt pool, where the WM solution deviates significantly from both. The evolution of melt-pool depth in SPH and COMSOL between $0.2 \text{ s} < t < 0.5 \text{ s}$, as seen in Figure 4.10, seems especially intriguing, as both methods agree while the WM solution shows a deviation. We can only assume that this deviation between the two FEM solvers is due to implementation specific details.

Another observation relates to the fact that the SPH solution deviates from the other two mostly at earlier times, i.e., $t < 0.3$ s, where the number of particles in the melt pool seems to be insufficient to allow development of a flow pattern. At the same time the implementation of a fixed boundary wall is not native to SPH, as SPH has its strengths especially in the treatment of unbounded domains and, to a certain degree, free fluid surfaces. The pressure forces in SPH might actually block the flow, while for WM the consideration of fixed wall boundary conditions is natural to the method and no such restrictions exist when a free-slip condition is used.

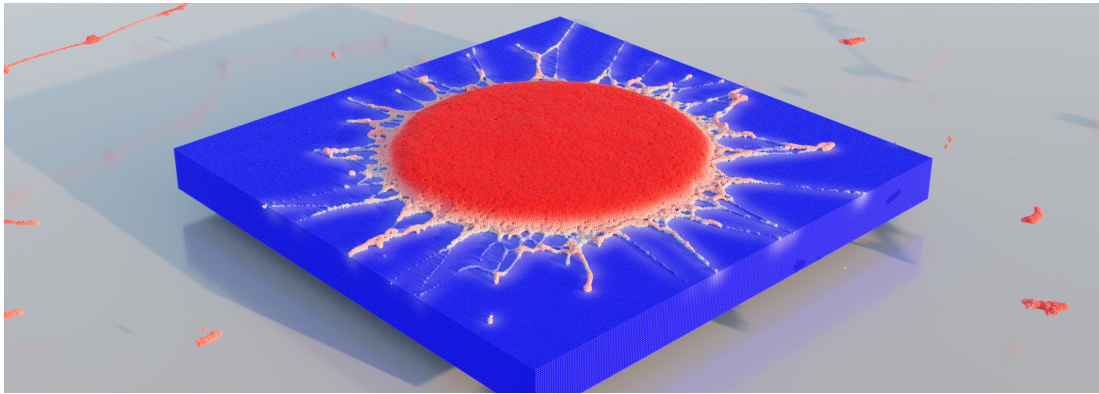
It should be kept in mind however, that for a real process, the free surface is not actually confined in such a way. Of course, for a comparison of melt pool development with the real process, also the non-linearity of the material parameters must be considered. Here, the inclusion of the latent heat in terms of the Stefan-Problem is known to present a significant challenge, among others. Nevertheless, even established Eulerian approaches have difficulty reaching satisfactory agreement without the use of strong compensation models and all the peculiarities that derive from the challenges related to the free surface deformation, discontinuities and strong topological changes, that are typical for arc welding processes. As such, while the inclusion of latent heat in terms of the Stefan-Problem is also no trivial task for SPH methods, we consider our SPH model to be a suitable candidate for performing simulations of the more complex interactions and physics of the real process, due to the improved ability to model surface deformation, discontinuities and significant topological changes.

4.5 Conclusion

In this chapter, a full 3D SPH model for the simulation of a TIG spot-welding process is proposed and compared quantitatively with 2+1D FEM simulations implemented in Wolfram Mathematica and COMSOL. We show remarkable agreement between our SPH method and the WM and COMSOL methods, especially given the differences observed between WM-FEM and COMSOL-FEM when solving identical problems. In the context of this thesis, these results provide the first quantitative benchmark for melt-driven SPH processes introduced in Chapter 1.

Overall, our work confirms the expectation that the SPH method has at least equivalent capability for quantitatively modeling welding processes. This holds even under conditions simplified to compensate for limitations of Eulerian methods (free surface), while at the same time strongly favoring the application of an Eulerian method (bounded domain). Therefore, we expect that SPH can outperform Eulerian approaches under more realistic conditions, i.e., allowing for a free surface and requiring full 3D also for the Eulerian method, especially once numerical treatment of non-linear material parameters is included. This chapter establishes a baseline for the corresponding thermal-spray study in the upcoming chapter.

Application and Benchmark of SPH for Modeling the Impact in Thermal Spraying



Thermal spraying is a coating technology where particles of a feedstock material are heated, fully or partially melted and accelerated to high speeds onto a substrate. Through the impact of many particles a coating is built up. As a coating technology, thermal spraying is divided into three major process variants: flame spraying, electric arc spraying and plasma spraying [DAVIES 2004]. This chapter follows the overall goal defined in Chapter 1 to evaluate point-based SPH methods for melt-driven processes against established Eulerian baselines, and complements the TIG welding study conducted in Chapter 4.

In this chapter, we take a closer look at plasma spraying, which is characterized by particle velocities of up to $v = 800 \text{ m s}^{-1}$ [PAWŁOWSKI 2008] and high plasma temperatures in the range of $T = 6000 \text{ °C}$ to $T = 15\,000 \text{ °C}$, which are significantly above the melting temperature of any known material [DAVIES 2004]. The injected particles are mostly in the size range of $d = 20 \text{ μm}$ to 90 μm [PAWŁOWSKI 2008]. As the particles spread upon impact, rapid solidification occurs with cooling rates in the range of $\dot{q}_{\text{cooling}} = 10^7 \text{ K s}^{-1}$ to 10^8 K s^{-1} as a result of heat transfer from the liquid material to the underlying substrate and to the ambient atmosphere [A. VARDELLE et al. 2014]. Thus, the particle deformation on the substrate, cooling, and solidification occur in rapid succession.

The properties of the coating, such as its durability or thermal conductivity are directly related to its microstructure, which is in turn directly related to the particle impact process. Therefore, a detailed understanding of the dynamics of particle impact on the substrate is essential for better control of

the coating build-up. The deposition of particles during plasma spraying can only be poorly observed experimentally, due to the fact that splat formation and solidification occur within a few microseconds [GOUTIER and FAUCHAIS 2011]. Consequently, many studies have been devoted to numerical and analytical investigation of particle impact, splat formation, and solidification [CHANDRA and FAUCHAIS 2009, GHAFOURI-AZAR et al. 2003, PASANDIDEH-FARD et al. 2002b, M. VARDELLE et al. 1995, K. YANG et al. 2012].

Apart from the simulation of the plasma jet and the heat transfer from the plasma to the particles, the impact of the melted particle onto the substrate and its subsequent deformation are of great interest. Given this context, the core contribution of this chapter is a novel SPH model that makes extensive use of implicit solvers and improves both simulation performance and stability (in line with the implicit SPH direction developed in Chapter 3). The novelty of our model comes from the unique combination of SPH components for thermal spraying, as well as the reformulation of explicit models into a single unified implicit linear system. Within this scope, we also introduce implicit formulations for established SPH correction methods that aim to improve physical accuracy. To better understand the capabilities of SPH for modeling droplet impacts, we quantitatively compare our SPH implementation to the volume of fluid (VOF) method on Eulerian grids using Ansys Fluent for a highly simplified case. This comparison shows strong agreement in the overall shape of the impacted droplet. Using our SPH method, we also simulate the droplet at significantly higher resolution while requiring only a fraction of the computational cost.

Furthermore, we present and discuss SPH simulation results in terms of droplet spread factor with varying initial droplet diameters. For these simulations we explicitly discretize the substrate and take into account the transient heat transfer between droplet and substrate as well as temperature-dependent material properties. Overall, we are able to show that SPH is a very suitable method for the simulation of droplet impacts in thermal spraying.

5.1 Related Work

In the past, droplet impact simulations have often been performed with the Eulerian volume of fluid (VOF) method. The VOF method can be used for the simulation of the free surface interface between two or more immiscible fluids by tracking the volume fraction of each of the fluids in two- or three-dimensional meshes. However, in the process of particle impact a large deformation of the molten particle occurs, from spherical to a thin layer. Therefore, it requires a very fine, or spatially adaptive, mesh discretization over a large area of the simulation domain. While fine meshes require exponentially increasing computational resources, mesh adaptivity is very difficult to implement correctly and still incurs a noticeable performance penalty. Most of the computational cost of VOF algorithms is incurred by the cells that form the interfaces between different fluids [BORRELL et al. 2013].

Other than tracking the position of the free surface, the heat transfer and especially the effect of solidification strongly influences the dynamics of the particle impact process. A popular method to model the effect of solidification is the enthalpy-porosity method [H. HU and ARGYROPOULOS 1996]. Here, heat transfer is solved in enthalpy formulation, with addition of a source term that is dependent on the solid fraction of the semi-solidified fluid in the so-called “mushy zone”, to account for the latent heat of melting and solidification. Furthermore, another related source term is added to the momentum equation to account for increased flow resistance of the fluid in the semi-solidified region, due to growth of dendritic structures. This momentum sink, also called Darcy-term, is dependent on the permeability, which in turn is also dependent on the solid fraction in the “mushy zone”.

Several works have applied the VOF method for modeling particle impact in the thermal spray process. PASANDIDEH-FARD et al. [2002a] developed a 3D model to simulate the impact and solidification of a molten droplet on a flat substrate by applying the fixed velocity approach for solidification, where

the solid is defined as liquid with infinite density and zero velocity. Another example is Y. Z. ZHENG et al. [2014], who developed a 3D particle impact model during plasma spraying utilizing the momentum source method of Ansys Fluent for modeling the solidification.

While these studies helped to increase the understanding of the particle impact and solidification in thermal spraying, a lot of computational time was generally required. In previous works the particle impact and solidification was modeled using a modified momentum source approach [BOBZIN et al. 2019] which was then applied to simulate multiple particle solidification [BOBZIN et al. 2021b]. The most recent work presents a calculation of the effective thermal conductivity using information of inter-splat gaps derived from a simulation of multiple splats of Al_2O_3 droplets [BOBZIN et al. 2021a]. However, while the experimental results showed the characteristic length of the gaps to be $< 1 \mu\text{m}$, the simulation could only resolve gaps with a width of its cell size of $2.25 \mu\text{m}$. This was resolved by numerically smearing the particle boundaries in this model and then enabling an approximately correct calculation of the thermal conductivity. However, this was only an interim solution, not a physically correct representation of the problem.

Due to the high computational cost of VOF-based Eulerian approaches, there has been a range of works applying the Smoothed Particle Hydrodynamics (SPH) method to the simulation of thermal spray deposition. This is often due to the versatility of SPH regarding the simulation of changing domain topology, free surfaces as well as multiple phases.

Since the feedstock material in the thermal spray process consists of particles and at the same time the SPH method is based on discretization particles, first and foremost the terms should be properly distinguished. The term *particle* is from now on used for the SPH discretization particle of the numerical method, while the feedstock particle that is molten in the thermal spray process and projected towards the surface will be termed *droplet*.

FANG et al. [2009] introduce many ideas for the SPH simulation of droplet spreading and solidification. They propose an improved pressure correction scheme and show simulations of droplets impacting on a solid substrate as well as similarities to some images obtained from experiments. In addition, for heat conduction an artificial heat model based on internal energy is used. Particles are classified into liquid, melting and solid, and a source term in the momentum equation accounts for phase change. Results are then compared to an experiment.

A similar approach is pursued by M. ZHANG et al. [2008] without the pressure correction, but also considering melting of the substrate for high thermal conductivities, low thermal capacities and high droplet temperatures. However, a validation of the method was not performed.

FARROKH PANAH et al. [2017, 2021] present a novel method for the simulation of latent heat in SPH with specific application to Suspension Plasma Spraying, which is a variant of the thermal spray process. They use an explicit weakly compressible SPH approach with advected density, as well as an enthalpy-viscosity method to model the process of solidification.

ABUBAKAR and ARIF [2019] introduce a hybrid approach for the simulation of spray deposition. The SPH method is used to model the dynamics of splat formation during the spray process while the Finite Element Method (FEM) is used to model the solidification and compute residual stresses. The results are validated qualitatively by comparison with experimental results in literature. While this approach is quite novel it relies on a very complex system in which numerical errors can occur at many different places (especially during transfers between discretizations). A coupled Eulerian and Lagrangian approach is also pursued by Z. ZHU et al. [2015] to simulate spray deposition of semi-molten ceramic droplets.

Other relevant SPH works, especially regarding heat transfer, melting and solidification can be found in the simulation of arc welding processes [ITO et al. 2015, KOMEN et al. 2018, TRAUTMANN et al. 2018]. KOMEN et al. [2018] in particular also simulate the impact of molten droplets onto a substrate

and investigate the effect on the weld pool, although the droplet impact velocities are significantly smaller and the droplet size is significantly larger than in thermal spraying.

5.2 Method

5.2.1 SPH Discretization

In the following, we will briefly outline our SPH method for the simulation of the droplet impact in thermal spraying. An introduction to the fundamentals of the SPH method can be found in Chapter 2. To recap, the equations typically used for the simulation of incompressible fluids are the continuity equation, Equation (2.6), and the Navier-Stokes equation, Equation (2.3). They are repeated here for quick reference:

$$-\rho \nabla \cdot \mathbf{v} = 0, \quad (5.1)$$

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}_{\text{ext}}. \quad (5.2)$$

Implicit solvers as presented in Section 2.2.2 are used for both pressure [BENDER and KOSCHIER 2015] and viscosity [WEILER et al. 2018], the same as in Chapter 4. The external forces contain both gravity and surface tension. For surface tension we make use of an explicit continuum surface force-based (CSF) approach for consistency with the VOF method comparison. For an overview of notation, the reader is again referred to the table at the beginning of this thesis and to Chapter 2.

Surface Tension

Surface tension computation in SPH is known to be a challenging problem, since it is very difficult to obtain a clear definition of the fluid surface. There exist formulations based on fluid surface curvature, often derived from the Continuum Surface Force (CSF) model by BRACKBILL et al. [1992], as well as formulations based on intermolecular forces. For our purposes, we have implemented the CSF model of M. MÜLLER et al. [2003] based on the model of MORRIS [2000], where the force $\mathbf{f}_{i,\text{st}}$, curvature $\nabla^2 c_i$, and surface normal $\mathbf{n}_i$ are computed from a smoothed color field c_i :

$$c_i = \sum_j \frac{m_j}{\rho_j} W_{ij}, \quad (5.3)$$

$$\mathbf{n}_i = \sum_j \frac{m_j}{\rho_j} (c_j - c_i) \nabla W_{ij}, \quad (5.4)$$

$$\nabla^2 c_i = - \sum_j \frac{m_j}{\rho_j} (c_i - c_j) \frac{2 \|\nabla W_{ij}\|}{\|\mathbf{x}_i - \mathbf{x}_j\|^2 + 0.01 h^2}, \quad (5.5)$$

$$\mathbf{f}_{i,\text{st}} = -\sigma \nabla^2 c_i \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|}. \quad (5.6)$$

Here σ denotes the surface tension coefficient [N m^{-1}] and $0.01 h^2$ is a small factor relative to the smoothing length h that prevents division by zero. The computation of the normals and especially the curvature is documented to be prone to errors due to particle disorder, however we have not observed any significant instabilities in our simulations. This could be due to the extremely small time scale of our simulations as well as due to other forces being more dominant.

Solidification

Solidification is often considered to be one of the main determining factors of the dynamics of the thermal spray process. It depends on the splat thickness, the thermal conductivities of both the sprayed

feedstock material as well as the underlying solid material, and the thermal contact resistance between the flattening droplet and the substrate. It directly affects the deformation behavior, the splat shape and the coating microstructure [PAWLOWSKI 2008]. The fluid of the droplet is cooled upon contact with the wall and the subsequent solidification process is modeled by taking into account a Darcy-term (momentum sink), after the well-known Enthalpy-Porosity method, for modeling the solidification of pure metals [BRENT et al. 1988] and of binary alloys [VOLLER et al. 1990]. The latent heat of melting and solidification is neglected for the initial comparisons between our SPH model and Ansys to keep the comparison as simple as possible. However, as the heat transfer solver is formulated in terms of the enthalpy, it can be easily extended to include the latent heat, which is shown in the simulations presented in Section 5.3.2.

We make use of the same Enthalpy-Porosity model as in Section 4.2.1 of the previous chapter about TIG-welding. The acceleration of this model is given by

$$\mathbf{a}_{\text{porosity}} = \frac{\mathbf{v}^t}{\Delta t} \left(\frac{1}{1 + \Delta t C(1 - f_l(T))} - 1 \right). \quad (5.7)$$

This Darcy-term adds a deceleration to the Navier-Stokes equation which has a strong movement inhibiting effect on the fluid, once the temperature of the fluid becomes low enough. This so-called momentum sink accounts for the semi-liquid state in the so-called mushy-zone, where already some nucleation and dendrite growth has occurred, thereby affecting the properties of the fluid. The effect is controlled by the liquid fraction $f_l(T)$, again modeled as a simple Heaviside function

$$f_l(T) = \begin{cases} 1 & T > T_l, \\ 0 & T_l - \Delta T_l \leq T \leq T_l, \\ - & T \leq T_l - \Delta T_l, \end{cases} \quad (5.8)$$

where the liquidus temperature is denoted by T_l and the temperature range of the mushy-region by ΔT_l , such that the fluid is assumed to be completely solid as soon as $T \leq T_l - \Delta T_l$. The final branch indicates complete solidification by an undefined factor $-$, as particles become frozen in place and unaffected by fluid forces. This is also the main difference to the approach in Chapter 4, where only melting occurred. This chapter instead simulates only solidification. The temperature range ΔT_l essentially lets us control how long the particles will be treated as “liquid” before their position becomes fixed. Furthermore, the semi-implicit formulation generally allows the usage of larger time steps in the simulation without causing instabilities, but comes at the cost of slightly dampening the effect. The parameters for the momentum-sink are shown in Section 5.2.3.

Correction Terms

In our implementation we have found that it is also necessary to add correction terms, which improve the quality of SPH simulations. The first such term was documented by MONAGHAN [1989] and reduces the interpenetration of particles by smoothing the velocity field while conserving linear and angular momentum, without adding dissipation. The acceleration of this correction, also sometimes called XSPH, is given by

$$\mathbf{a}_{i,\text{xsph}} = -\frac{\epsilon}{\Delta t} \sum_{j \in \mathcal{N}_i} \frac{\overline{m}_{ij}}{\overline{\rho}_{ij}} (\mathbf{v}_i - \mathbf{v}_j) W_{ij}, \quad (5.9)$$

where ϵ denotes the (dimensionless) strength of this smoothing, $\overline{m}_{ij}$ the average mass between particle i and j and $\overline{\rho}_{ij}$ the average density. In the original work $\epsilon = 1$ was proposed, yet we have found smaller values, in the range of 0.1 to 0.3, to be preferable in our simulations. Smaller values caused less smoothing and retained more fluid details.

The second correction addresses the issue of tensile instability at the surface of SPH fluids. When using a method which recomputes the density instead of advecting it, it occurs that the density estimate at free surfaces is erroneous due to missing particles and causes an uncontrollable artificial surface tension effect. This is solved in the DFSPH method by only considering “over”-pressures due to larger density values and clamping smaller densities to the rest density. This entirely removes the instability at the surface, but comes at the cost of reduced fluid cohesion. In order to restore fluid cohesion a corrective force of the form

$$\mathbf{a}_{i,\text{cohesion}} = -\frac{\gamma^f}{m_i} \sum_{j \in \mathcal{N}_i^f} \frac{\bar{m}_{ij}}{\bar{\rho}_{ij}} \mathbf{x}_{ij} W_{ij}, \quad (5.10)$$

is employed, where γ^f is a parameter controlling the strength of cohesion [N m^{-1}] and $\mathcal{N}_i^f$ denotes the neighborhood of fluid particle i within the same fluid phase.

Adhesion to other phases and boundaries is formulated analogously

$$\mathbf{a}_{i,\text{adhesion}} = -\frac{\gamma^b}{m_i} \sum_{j \in \mathcal{N}_i^b} \frac{\bar{m}_{ij}}{\bar{\rho}_{ij}} \mathbf{x}_{ij} W_{ij}, \quad (5.11)$$

where γ^b is a parameter controlling the strength of adhesion [N m^{-1}] and $\mathcal{N}_i^b$ denotes the neighborhood of fluid particle i within the boundary phase. It should be noted that for the case that $\gamma^b = \gamma^f$ the net cohesive-adhesive-force at the boundary is zero. Adhesion is therefore modeled by $\gamma^b > \gamma^f$ and repulsion by $\gamma^b < \gamma^f$. By design this correction term only adds forces in regions with particle deficiency and is inspired by the work of MONAGHAN [2000], yet instead of adding an additional repulsion term the attraction is first clamped by the implicit pressure solver and then reintroduced as cohesion and adhesion. A very similar cohesive force was also used by BECKER and TESCHNER [2007].

We note that due to the anti-symmetric nature of these formulations, they conserve angular and linear momentum. In order to further improve the stability and to be able to use larger time steps, we formulate these corrections implicitly in terms of velocity and incorporate them, together with the viscosity force, into a single linear system

$$\begin{aligned} \frac{\mathbf{v}_i^{t+1} - \mathbf{v}_i^t}{\Delta t} = & \frac{\mu}{\rho_i} 2(d+2) \sum_{j \in \mathcal{N}_i} \frac{\bar{m}_{ij}}{\rho_j} \frac{\mathbf{v}_{ij}^{t+1} \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \nabla W_{ij} \\ & - \frac{\epsilon}{\Delta t} \sum_{j \in \mathcal{N}_i} \frac{\bar{m}_{ij}}{\bar{\rho}_{ij}} \mathbf{v}_{ij}^{t+1} W_{ij} \\ & - \frac{\gamma^f}{m_i} \sum_{j \in \mathcal{N}_i^f} \frac{\bar{m}_{ij}}{\bar{\rho}_{ij}} \mathbf{x}_{ij} W_{ij} \\ & - \frac{\gamma^b}{m_i} \sum_{j \in \mathcal{N}_i^b} \frac{\bar{m}_{ij}}{\bar{\rho}_{ij}} \mathbf{x}_{ij} W_{ij}, \end{aligned} \quad (5.12)$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and d is the number of spatial dimensions, i.e., $d = 3$. This linear system is solved using the matrix-free conjugate gradient method.

At the time of development we were not aware of any other works utilizing implicit solvers in SPH simulations to this degree. By doing this, we observe a significant performance improvement because simulations can use large time steps without becoming unstable.

Pseudocode of our full fluid solver is shown in Algorithm 8. The only explicitly computed component is surface tension, while pressure and the remaining forces are integrated implicitly, although in separate steps. Heat transport is governed by the Fourier Equation and solved using the explicit method described in Section 4.2.2. The full SPH solver is summarized in Algorithm 9.

Algorithm 8 Fluid Solver

```

1: procedure FLUIDSOLVE
2:   DFSPH_DIVERGENCEFREE ▷ [BENDER and KOSCHIER 2015]
3:   SOLVEVISCOSITYANDCORRECTIONS ▷ Equation (5.12)
4:   for all particles  $i$  do
5:     COMPUTESURFACETENSION( $i$ ) ▷ Equation (5.6)
6:     SOLVEMOMENTUMSINK( $i$ ) ▷ Equation (5.7)
7:   end for
8:   DFSPH_CONSTANTDENSITY ▷ [BENDER and KOSCHIER 2015]
9:    $\Delta t \leftarrow \text{COMPUTECFLTimestep}()$ 
10:  for all particles  $i$  do
11:     $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \mathbf{a}_i$ 
12:     $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$ 
13:  end for
14: end procedure

```

Algorithm 9 Solver Overview

```

1: procedure FLUIDANDHEATSOLVER
2:    $t \leftarrow t_{\text{start}}$ 
3:   while  $t < t_{\text{end}}$  do
4:     HEATSOLVE() ▷ Algorithm 6
5:     FLUIDSOLVE() ▷ Algorithm 8
6:      $t \leftarrow t + \Delta t$ 
7:   end while
8: end procedure

```

5.2.2 Boundary Conditions

Boundary contributions for all SPH terms are computed using the approach of AKINCI et al. [2012]. The only boundary in our simulation is the substrate, which will be described further in Section 5.2.3. Using the approach of Akinci et al. the boundary is sampled using a single layer of particles on the surface of the boundary. The contribution of the boundary to the SPH summation of fluid particles can be generalized as

$$A_i = A_i^f + A_i^b = \sum_{j \in \mathcal{N}_i^f} V_j A_j W_{ij} + \sum_{j \in \mathcal{N}_i^b} V_j^b A_j^b W_{ij}. \quad (5.13)$$

The superscript f indicates contribution from fluid particles while b indicates contribution from boundary particles within the compact support of particle i . The volume of boundary particles is computed using

$$V_i^b = \frac{1}{\sum_{j \in \mathcal{N}_i^b} W_{ij}}, \quad (5.14)$$

which is an SPH summation over the other boundary particles in the compact support of boundary particle i . The boundary volumes are incorporated into summations of fluid particles by extending fluid quantities into the boundary region. This means, for example, using the rest density of the fluid to compute a mass using the boundary volume and utilizing this contribution for fluid density computations. Similar considerations can be made for all other cases, such as for the Fourier equation. The heat conduction term in Equation (4.11) is extended by the following term for the boundary contribution

$$\nabla \cdot (\lambda \nabla T)_i^b = \sum_{j \in \mathcal{N}_i^b} V_j^b 2\lambda_i (T_i - T_j^b) \frac{\nabla_i W_{ij} \cdot \mathbf{x}_{ij}}{\mathbf{x}_{ij}^2}. \quad (5.15)$$

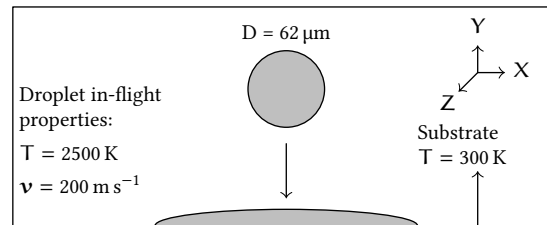
This is equivalent to Equation (4.11), when also using $\lambda_j^b = \lambda_i$ and prescribing the wall temperature T_j^b . This allows specifying a Dirichlet boundary condition on the substrate. According to MOSTAGHIMI

Table 5.1. *Material properties of ceramic droplet and boundary conditions.*(a) *Material properties of the ceramic droplet: Al_2O_3 [BOBZIN et al. 2021a]*

Property	Unit	Value
Droplet diameter	μm	62
Density	kg m^{-3}	3,950
Specific heat	$\text{J kg}^{-1} \text{K}^{-1}$	900
Thermal conductivity	$\text{W m}^{-1} \text{K}^{-1}$	30
Viscosity	$\text{kg m}^{-1} \text{s}^{-1}$	0.055
Surface tension	kg s^{-2}	0.8
Melting temperature	K	2,027

(b) *Wall boundary conditions*

Property	Unit	Value
Wall temperature	K	300
Free-slip boundary condition		

**Figure 5.1.** *Schematic diagram of the simulation domain for the droplet impact.*

and CHANDRA [2018] the estimated heat loss of the droplet to the surrounding gas is roughly three orders of magnitude lower than that of heat conduction into the substrate. Therefore, we assume the free surface of the droplet to be adiabatic, i.e., we neglect heat losses of the droplet into the surrounding gas for our current investigations.

5.2.3 Simulation Domain

In the previous sections, we introduced our simulation model for droplet impact onto a substrate with solidification. Subsequently, we describe the simulation setup used to compare our SPH model with Ansys Fluent. Figure 5.1 shows the simulation domain for droplet impact. This includes the material properties, the initial droplet in-flight properties, the substrate wall properties and boundary conditions. The material properties of the ceramic droplet are listed in Table 5.1. Further, the simulation parameters in Ansys Fluent and for SPH are listed in Table 5.2.

Ansys Fluent

A 3D thermal spray coating build-up model based on a previous publication [BOBZIN et al. 2021a], was created and implemented in Ansys Fluent. In this model the impact of thermally sprayed ceramic droplets onto a flat substrate was simulated. A momentum-source function was used to simplify the calculation of the solidification process, with parameters as seen in Table 5.2c. A laminar viscous model was used and the energy solver was enabled. The dimensions of the spatial domain are $225 \mu\text{m} \times 225 \mu\text{m} \times 75 \mu\text{m}$, incorporating the droplet as well as a surrounding gaseous atmosphere which was assumed to be air. The mesh edge length is $2.25 \mu\text{m}$ which resulted in a calculation mesh of 330,000 cells. The boundaries of the domain consist of an inlet for the droplet on top, with velocity $v_p = 200 \text{ m s}^{-1}$, the substrate as a free-slip wall at the bottom and outlets with pressure $p_{\text{amb}} = 101\,325 \text{ Pa}$ and backflow

Table 5.2. *Simulation parameters***(a)** *Simulation parameters in SPH*

Property	Unit	Value
Particle radius r_{part}	μm	0.4
CFL min time step size	s	1×10^{-12}
CFL max time step size	s	1×10^{-8}
Simulation method	-	DFSPH
XSPH ϵ Equation (5.9)	-	0.3
Cohesion γ^f Equation (5.10)	N m^{-1}	200
Adhesion γ^b Equation (5.11)	N m^{-1}	200

(b) *Simulation parameters in Ansys Fluent*

Property	Unit	Value
Domain size	μm	225×225
Mesh size	μm	2.25
CFL min time step size	s	1×10^{-12}
CFL max time step size	s	1×10^{-7}
Simulation method	-	FVM

(c) *Momentum sink parameters*

Property	Unit	Value
T_l	K	2,027
ΔT_l	K	100
C	s^{-1}	3×10^7

total temperature $T_{\text{out}} = 3000$ K. The interior domain is filled with air at $T_{\text{gas}} = 2400$ K at the start of the simulation. The temperature of the substrate is set to $T_{\text{substrate}} = 300$ K. The free-slip boundary condition was applied for the contact between the droplet and the substrate and a VOF-approach was assumed for the reconstruction of the free surface of the ceramic droplet and the surrounding gas phase. Due to the rapid solidification resulting from the Dirichlet thermal boundary condition, the free-slip boundary condition appeared to be a reasonable assumption in our experiments. The numerical parameters of the simulation in Ansys Fluent are summarized in Table 5.2b. The simulations were performed using Ansys Fluent 2020 R2.

SPH

The same 3D thermal spray coating build-up model was created using our SPH method. The domain of the simulation model is shown in Figure 5.2a. The numerical parameters for the simulation and for the momentum sink are each listed in Table 5.2a and Table 5.2c, respectively. The droplet has a diameter of $d_{\text{droplet}} = 62 \mu\text{m}$ and initial in-flight temperature $T = 2500$ K and downward initial velocity magnitude $\|\mathbf{v}_0\| = 200 \text{ m s}^{-1}$. The droplet was discretized by particles with an individual radius of $r = 0.4 \mu\text{m}$ and consists of 238,310 particles in total. The substrate was modeled as a rigid body with a Dirichlet boundary condition for the temperature of $T_{\text{wall}} = 300$ K and a free-slip boundary condition for the momentum equation. The model was implemented in a custom branch of SPlisHSPlasH [BENDER et al. 2026].

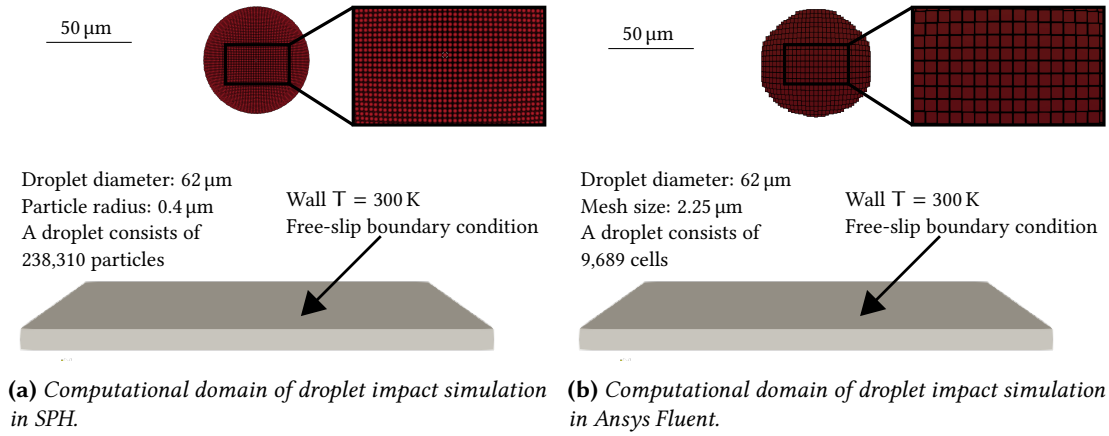


Figure 5.2. Computational domain of droplet impact simulation.

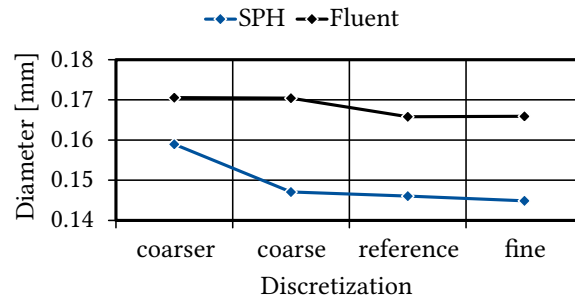


Figure 5.3. Convergence analysis of the simulation models in SPH and Ansys Fluent.

5.3 Results

In this section, we evaluate our SPH model in two different settings. First, we compare the results of our SPH model to the results obtained by a simulation using Ansys Fluent with the VOF method. For this, we consider simplified material parameters and boundary conditions in order to eliminate possible modes of deviation and inaccuracy in both methods. We first attempt to find sufficient discretization densities by conducting a mesh convergence analysis using the diameter of the splat as the convergence indicator value. Afterwards, we take into account temperature-dependent material properties, the latent heat of melting and the heat transfer into the substrate. With this we evaluate splat shapes and spread factors of different diameter droplets and validate these results by comparison with an analytical expression.

5.3.1 Comparison to Ansys-Fluent

Mesh Convergence First, a sensitivity analysis of the mesh and particle resolution was conducted. The droplet impact simulation was run with different mesh sizes and particle resolutions. Both simulations were computed on 32 cores of a high-performance compute cluster. For the simulation in Ansys Fluent, the coarser, coarse, reference, and fine computational meshes consist of 83349, 165099, 330000, and 540000 cells respectively. For the simulation using our SPH method, the coarser, coarse, reference, and fine particle resolutions consist of 60112, 119129, 238310, and 476486 particles, respectively. The results in Figure 5.3 show that as the resolution increases in SPH and Ansys Fluent, the diameters of the splats converge and remain nearly identical at resolutions above these values. In addition, the increasing resolution leads to a longer computation time. The computation time required to solve the simulation in Ansys Fluent from coarser to finer mesh resolution is about 5, 8, 20 and 30 minutes. On the

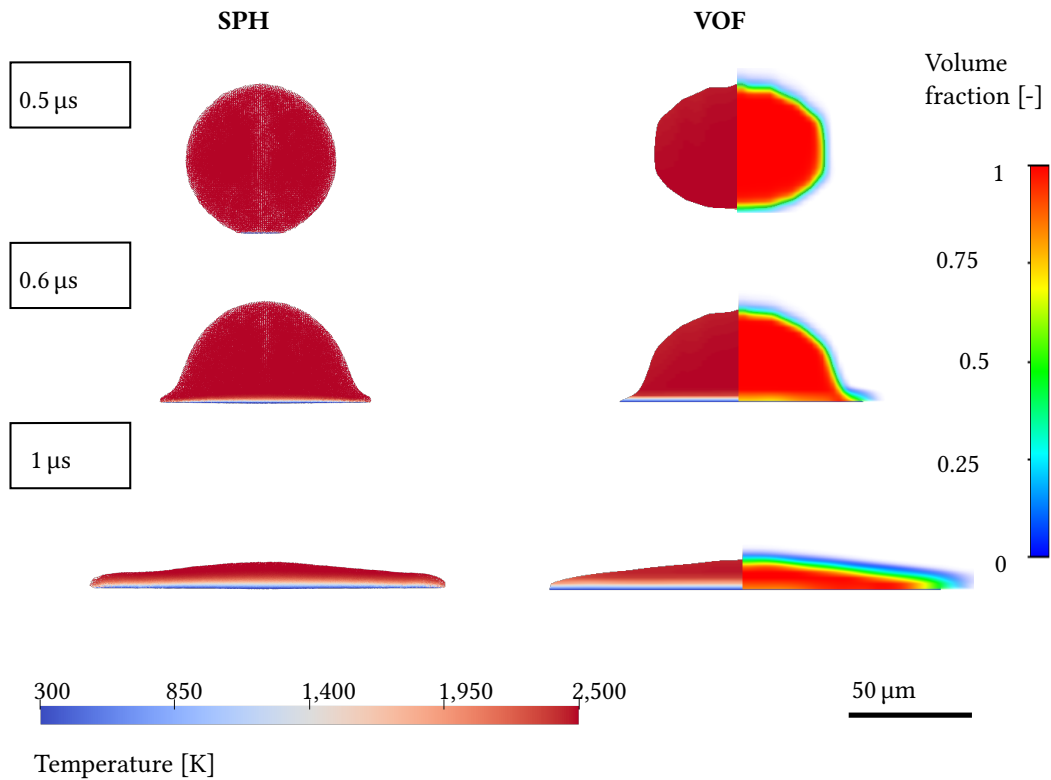


Figure 5.4. Comparison of cross-sections of the droplet impact in SPH (left) and Ansys Fluent/VOF=0.5 (right), for several points in time.

other hand, SPH generally requires less computation time, which is about 1, 4, 5 and 20 minutes from coarser to finer particle resolution. Ultimately, the reference mesh size of $2.25\ \mu\text{m}$ and the reference particle radius of $0.4\ \mu\text{m}$ are used in subsequent simulations for Ansys Fluent and SPH, respectively.

Results Figure 5.4 shows droplet impact, subsequent spreading, and solidification modeled with SPH (left) and Ansys Fluent (right). The main dynamics of the process occur at the three points in time shown. Overall, there is high agreement between the two methods. However, it should be noted that the Ansys Fluent approach is performed with the VOF method for modeling the free surface of the liquid and therefore the presented shape represents the iso-surface of volume fraction 0.5 of the ceramic phase. While the overall resolution of the mesh is quite high, it is relatively coarse in the region of interest, as can be seen in Figure 5.5. As such, the dispersion of the fluid boundary surface can be considerable in the VOF method and the apparent area of the cross section appears somewhat smaller than the area of the cross section in the SPH method, although in both cases the total mass is conserved. Nevertheless, Figure 5.6 shows the decrease of the ratio of mass enclosed by the VOF-0.5 contour with respect to the total fluid mass, as the mass disperses across a larger region. After impact at $t = 0.5\ \mu\text{s}$, the mass share of the VOF-0.5 contour of the droplet decreases steadily and then levels out at $t = 1.4\ \mu\text{s}$.

Another peculiarity is the shape of the droplet in free flight. While in the SPH method, the shape is highly spherical, the VOF droplet appears to be elliptically compressed in the direction of flight in the VOF method. While the surrounding region of the droplet is filled with stagnant air, the droplet itself is immersed in an airstream of the impact velocity in order to avoid compression due to drag. As such, the slightly compressed shape can only be explained by difficulties in using a transient inlet function.

Figure 5.7 presents top views of droplet impact at several points in time. The splat shapes in Fluent and SPH are almost perfectly symmetric. Furthermore, it was observed that the splat seems to cool off faster in Fluent than in SPH. However, it should be noted again that the visible surface in Ansys Fluent

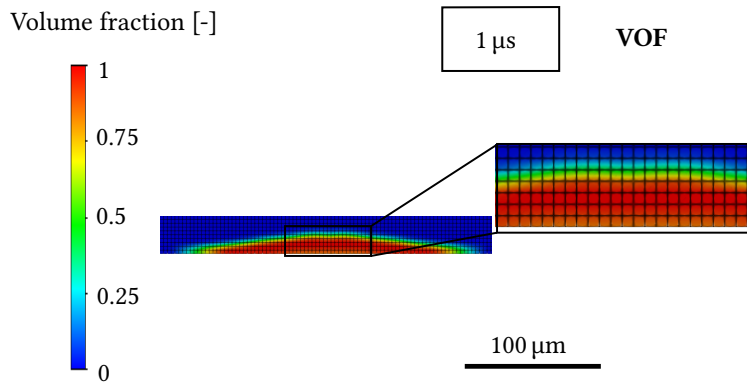


Figure 5.5. Cross-section of the volume fraction of the ceramic phase in Ansys Fluent at $t = 1.0 \mu\text{s}$.

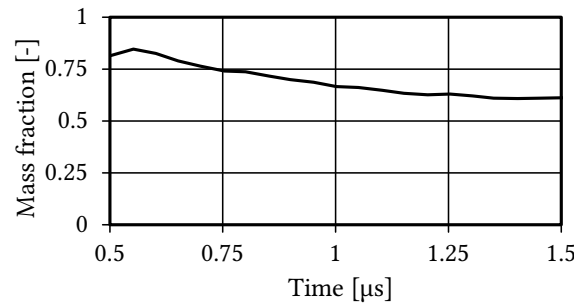


Figure 5.6. Mass share of the ceramic phase enclosed within the volume $\text{VOF} \geq 0.5$ of the total mass of the phase.

corresponds to the volume fraction 0.5 of the ceramic phase. Additionally, for times $t \geq 1.0 \mu\text{s}$ the thickness of the splat became smaller than 5 to 6 mesh cells (see also Figure 5.5), which is problematic in the VOF approach, as it disperses the boundary of the free surface and therefore requires several mesh cells for the transition from one fluid to the other. When the ratio of the number of cells in the transition region to the number of cells with volume fraction 1.0 becomes very large, it becomes more difficult to make accurate observations about the enclosed volume. This is due to the fact that observations about the enclosed volume become highly sensitive to the selection of the contoured volume fraction. It is therefore concluded that the results for $t \geq 1.0 \mu\text{s}$ should be considered with care, but they are included in this figure for reference.

The diameter and the height of the formed splat were compared for SPH and Ansys Fluent for several volume fractions of the ceramic phase (0.1, 0.5, 0.9) over time in Figure 5.8. The droplet impacted the substrate at $0.5 \mu\text{s}$ and subsequently spread out, gaining in diameter and losing in height until it reached a steady state. When examining the diameter and height of the splat calculated with the VOF method in Ansys Fluent, the dimensions vary significantly depending on the volume fraction, consistent with the observation in Figure 5.5. Figure 5.8a shows good agreement in diameter between SPH and the VOF method. The diameter calculated with SPH lies between those at volume fractions 0.5 and 0.9. It should be noted that the cohesion correction parameter (see Section 5.2.1) was adjusted manually to reach this agreement. As discussed in the analysis of Figure 5.7, the presented results of Ansys Fluent for $t \geq 1.0 \mu\text{s}$ do not have a sufficient mesh resolution, despite having a total of 330k cells, and are therefore not discussed further.

The height, as seen in Figure 5.8b shows the same tendency for both approaches but a consistently smaller height was observed for all volume fractions of the ceramic phase in Ansys Fluent when compared to SPH. This is again consistent with the observed decrease in cross-section area in Figure 5.5

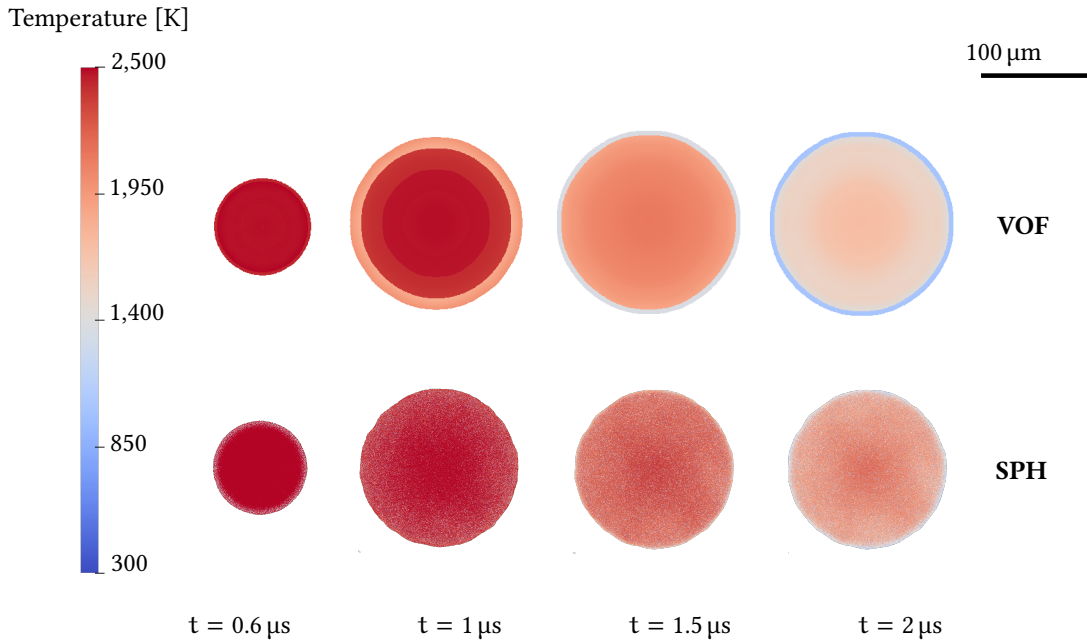


Figure 5.7. Top view of the simulated splats in SPH (bottom) and Ansys Fluent/VOF=0.5 (top) at several times. Times $t = 1.5 \mu\text{s}$ and $t = 2.0 \mu\text{s}$ for Ansys Fluent are included for reference, although the resolution of the mesh is too low for the splat thickness to derive meaningful results.

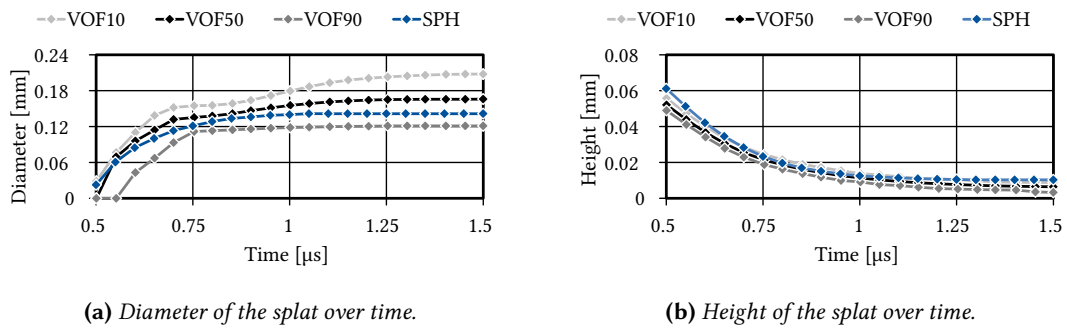
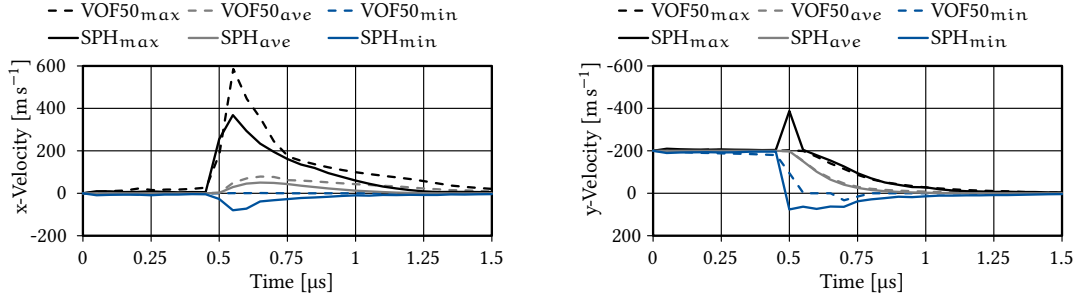


Figure 5.8. Comparison of the diameter and height of the simulated splats over time for SPH and Ansys Fluent at several volume fractions (0.1, 0.5, 0.9) of the ceramic phase.

and in mass share of the droplet in Figure 5.6. However, since both the height as well as the diameter have a strong influence on the heat transfer from the splat to the substrate, this difference is of high significance. We attribute higher confidence to the SPH result regarding the height, because it does not suffer from the aforementioned volume dispersion.

Figure 5.9 shows a comparison of the simulated maximum, minimum, and average velocities over time, evaluated over a half-space in radial (x) and height (y) direction for Ansys Fluent (at VOF = 0.5) and SPH. As the droplet impacts the substrate at $t=0.5 \mu\text{s}$, it can be seen in Figure 5.9a that shortly after impact a very strong increase of the maximum radial velocity from the initial maximum radial velocity of 0 m s^{-1} occurs for both methods. This increase reaches roughly 350 m s^{-1} for SPH, while the maximum radial velocity reaches nearly 600 m s^{-1} in the case of Ansys Fluent. After this initial increase, the maximum radial velocity decreases towards zero for both cases. Compared to this, the average velocity taken over a half-space in both cases shows good agreement, with SPH exhibiting a consistently lower average radial velocity. Furthermore, it can be observed that the minimum velocity in the Ansys Fluent case remains zero for the entire duration, while the minimum velocity in SPH becomes negative, with a small but distinct minimum of approximately -40 m s^{-1} at the moment of impact. During the



(a) Comparison of the maximum, minimum and average radial velocity in Ansys Fluent and SPH, taken over the half-space of a symmetry plane.

(b) Comparison of the maximum, minimum and average axial velocity in Ansys Fluent and SPH; the in-flight velocity of the droplet is directed in the negative y-direction.

Figure 5.9. Comparison of the maximum, minimum and average radial and axial velocity of the droplet.

time shortly after impact, the negative velocities in radial direction are somewhat contrary to the expected dynamics of the process, in which the fluid of the droplet would spread outward (positive velocity in radial direction) to form the splat. Upon further investigations, these particles are generally located near or on the cut plane where due to particle disorder some particles accelerating in negative x-direction may appear.

In Figure 5.9b the maximum, minimum, and average velocity in vertical direction are shown. Please note that the droplet moves towards the substrate, i.e., in negative y-direction. After impact at $t = 0.5 \mu\text{s}$, the maximum vertical velocity decays smoothly to zero in the case of Ansys Fluent. In contrast to this, the observed maximum vertical velocity shows a slightly different behavior in SPH. It has a peak of almost -400 m s^{-1} at the time of impact $t = 0.5 \mu\text{s}$ before decreasing smoothly towards zero. The maximal vertical velocities of both cases for $t \geq 0.6 \mu\text{s}$ show an otherwise excellent agreement. Similarly, the minimum velocity drops rapidly after impact and remains at zero or close to zero in the case of Ansys Fluent, while the minimum velocity simulated in SPH shows a different post-impact behavior. At the time of the impact, the minimum velocity reaches an absolute minimum of roughly 50 m s^{-1} before approaching zero.

It is noticeable that the peak of the maximum velocity precedes the negative peak of the minimum velocity. While this can be understood in terms of a rebound effect, the reason for the large spread between minimum and maximum velocity, as well as the deviation with Ansys Fluent of these observables will be discussed later in detail in the analysis of Figure 5.10. Finally, the average velocities of both simulation methods have an excellent agreement over time, even better than was the case for the radial velocity in Figure 5.9a. The droplet starts with a velocity of -200 m s^{-1} in both methods, then the average velocity in vertical direction decreases gradually after impact and reaches zero at $t = 1.0 \mu\text{s}$.

A more detailed analysis of the apparent disagreement noted in Figure 5.9b can be seen in Figure 5.10 for the moment of impact at $t = 0.5 \mu\text{s}$. It can be seen in Figure 5.10 that a small fraction of particles at the side of the droplet have a very high vertical velocity of -350 m s^{-1} (color-coded in blue). Next to particles that are in contact with the wall, a small fraction of particles experience the rebound effect and their velocity reaches nearly 50 m s^{-1} (color-coded in red), before being counteracted by the bulk movement of the droplet. The main bulk of the particles has a velocity range from 0 m s^{-1} to -200 m s^{-1} , which corresponds to the in-flight velocity of the droplet and solidified particles. This observed velocity peak at the moment of impact is assumed to result from the sudden difference in velocity due to solidification of the fluid and the subsequent increase in local density and pressure. However, this does not necessarily imply an unphysical result, but on the contrary it might actually capture the real conditions even more accurately than the Eulerian method in Ansys Fluent.

Note that the investigation of minimum and maximum velocities are difficult to compare quantita-

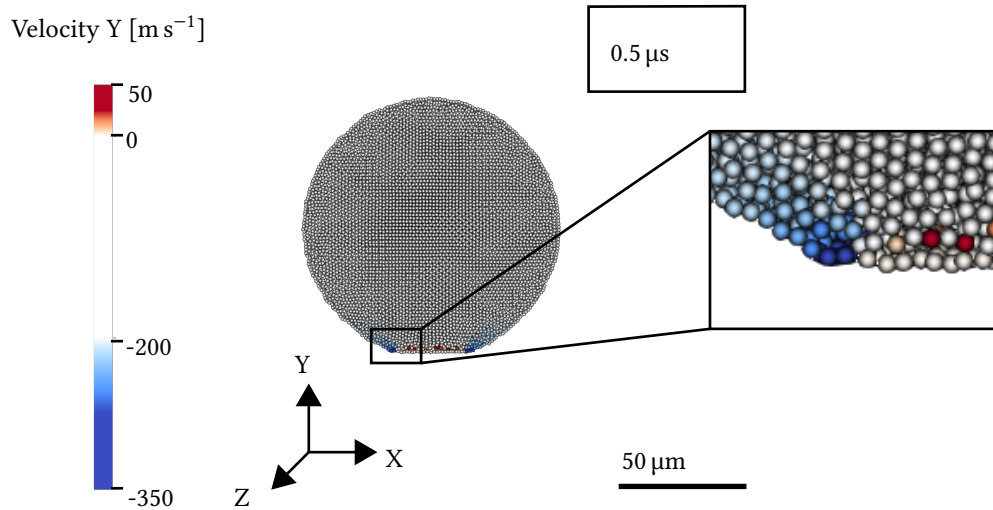


Figure 5.10. Axial velocity at the moment of impact; the in-flight velocity of the droplet is directed in negative y -direction.

Table 5.3. Material properties of ceramic droplet and substrate. Note that the latent heat of melting and the specific heat capacity are included in the specific enthalpy.

(a) Temperature-dependent material properties of the ceramic droplet: Al_2O_3 [BOBZIN et al. 2021a]

Property	Unit	Value
Specific enthalpy	J kg^{-1}	Figure 5.13a (blue)
Thermal conductivity	$\text{W m}^{-1} \text{K}^{-1}$	Figure 5.13b (blue)

(b) Material properties of the substrate: AISI 440C stainless steel

Property	Unit	Value
Density	kg m^{-3}	7,850
Specific enthalpy	J kg^{-1}	Figure 5.13a (grey)
Thermal conductivity	$\text{W m}^{-1} \text{K}^{-1}$	Figure 5.13b (grey)
Viscosity	$\text{kg m}^{-1} \text{s}^{-1}$	5
Melting temperature	K	1,756

tively and are only discussed in order to give better insight into the dynamics of the process for each simulation method. While the actual values differ slightly, the overall trends visible in the minimum and maximum velocities are very similar for both methods.

5.3.2 Simulation with Substrate

We now investigate the spread factor of droplet impacts given different initial diameters. For this we modify previous material properties to be temperature-dependent and to also take into account the latent heat of melting. In addition, we explicitly discretize the substrate and also introduce temperature-dependent material parameters. The modified material properties are summarized in Table 5.3 and Figure 5.13. Note that only modified material properties are listed and that the cohesion γ^f and adhesion factors γ^b were adjusted to $\gamma^f = 300 \text{ N m}^{-1}$ and $\gamma^b = 300 \text{ N m}^{-1}$.

We simulate droplets of three different diameters: $30 \mu\text{m}$, $45 \mu\text{m}$ and $62 \mu\text{m}$. The droplets and substrate are discretized with a particle radius of $0.4 \mu\text{m}$. This resulted in a total of 1908591, 1972736,

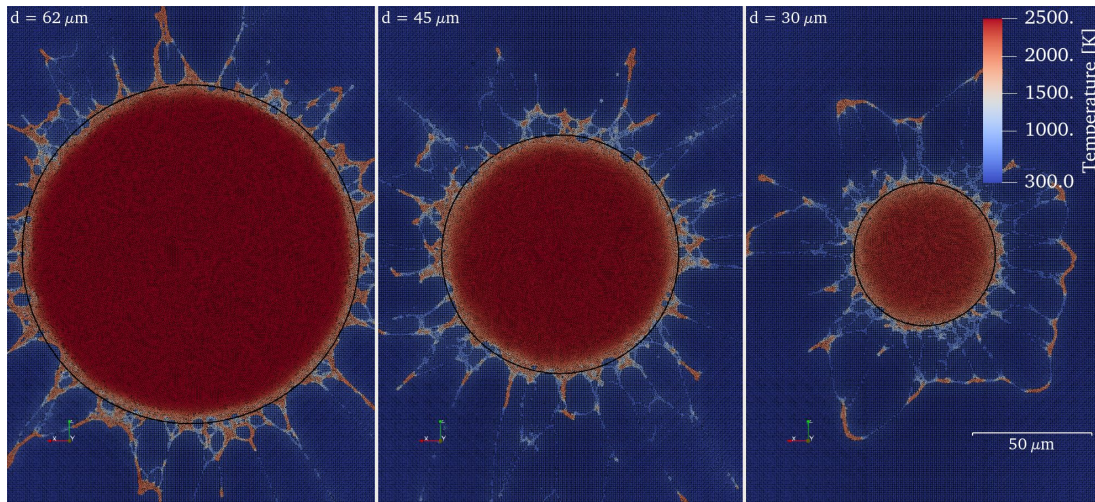


Figure 5.11. Splat shapes of various initial droplet sizes after 2 μ s of simulated time. From left to right: 62 μ m, 45 μ m, 30 μ m. The black circles denote the measured splat diameter. Best viewed at high magnification in the digital version.

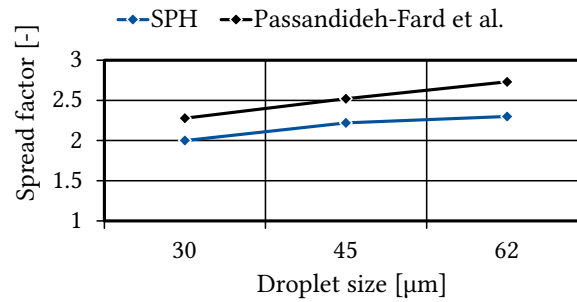
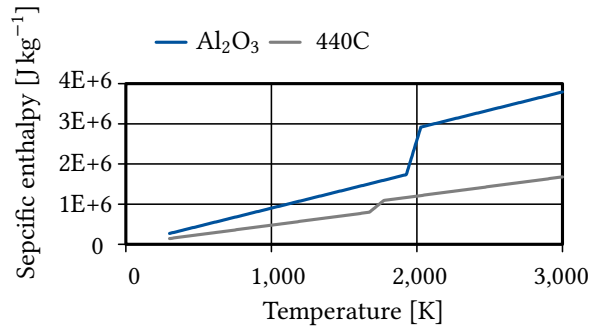


Figure 5.12. Comparison of spread factors of various droplet sizes with maximum spread factors according to PASANDIDEH-FARD et al. [1996].

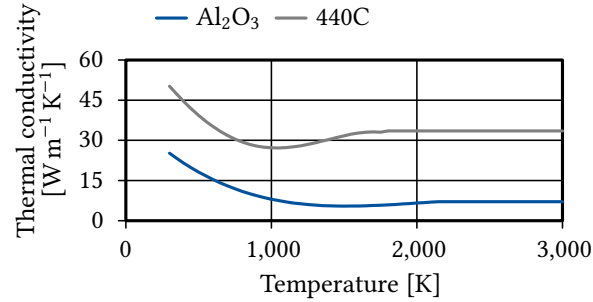
2119910 particles for the three different droplet diameters, respectively. The computation time required to solve the simulation from the smallest to the largest droplet diameter was about 21, 51 and 53 minutes, respectively. The large number of particles in the simulation is mostly due to the discretization of the substrate, which may be further optimized in future work to reduce the computation time. The splat shapes of these droplets are shown in Figure 5.11.

For all splats, movement has ceased after 2 μ s of simulated time and a splat with splashes has formed, a process which is described in the literature for sufficiently large impact velocities [MOSTAGHIMI and CHANDRA 2018]. The only remaining process is the solidification by heat conduction into the substrate. The speed of solidification is well-known to be governed by the thermal contact of droplet and substrate. This thermal contact is naturally modeled by our adhesion method which effectively governs the average distance of particles from the surface of the substrate and the particle density on the surface of the substrate. Higher adhesion values would result in a better thermal contact while smaller values would inhibit thermal contact, which may be interpreted as describing the surface roughness of the sprayed surface on a macro scale.

The maximum spread factor $\xi_{\max} = \frac{D_{\max}}{D_0}$ of the different initial droplet sizes D_0 equal to 30 μ m, 45 μ m and 62 μ m calculated by the SPH solver are shown in Figure 5.12. These droplet diameters correspond to non-dimensional Reynolds numbers of 431, 646, and 891 for $Re = \frac{\rho V_0 D_0}{\mu}$, and Weber numbers 5925, 8888, and 12245 for $We = \frac{\rho V_0^2 D_0}{\sigma}$. According to PASANDIDEH-FARD et al. [1996], since the Weber numbers are much larger than the Reynolds numbers, capillary effects can be neglected.



(a) Specific enthalpy of Al_2O_3 droplet and AISI 440C stainless steel substrate with latent heat of melting as a function of temperature.



(b) Thermal conductivity of Al_2O_3 droplet and AISI 440C stainless steel substrate as a function of temperature.

Figure 5.13. Temperature-dependent material properties of ceramic droplet and substrate.

Therefore, the maximum spread factor can be approximated as $0.5 \text{ Re}^{0.25}$. The maximal diameter of our simulations D_{max} was evaluated by hand to 60 μm , 100 μm and 142 μm , respectively. The corresponding circles are shown in black in Figure 5.11. The results obtained for the maximum spread factor with the SPH method show a good correlation ($R^2 = 0.9565$) with the analytical results approximated by PASANDIDEH-FARD et al. [1996] as well as with the results predicted by FARROKHPANAH et al. [2021] ($R^2 = 0.9960$). For a direct comparison of spread factors between our results and the analytical results see Figure 5.12. We conclude that our model is able to reproduce the droplet spread factor increase that is known to occur with an increase in initial droplet diameter.

5.4 Discussion

In the previous section we compared the results of our SPH simulation against a simulation using the commercial tool Ansys Fluent.

While in the real process the partial melting of the droplet and the heat transfer, including the latent heat of melting and solidification, cannot be neglected, the main goal in this chapter was to compare the performance of fundamentally different numerical methods under droplet-impact conditions in thermal spraying. It is therefore considered justified to initially keep the model as simple as possible, also neglecting most non-linearities like temperature-dependent material parameters for the comparison.

We used identical physical models and parameters for all phenomena, except for the corrective terms employed to improve SPH accuracy. The corrective factors were adjusted to be as small as possible in order to closely match the droplet dimensions computed by Ansys Fluent. From this, we obtained excellent results and good agreement with the Ansys Fluent simulation.

In terms of computational efficiency our proposed SPH method also compares very favorably to Ansys Fluent. In our SPH method, we used a total of 230k particles, while Ansys Fluent used 330k mesh cells. While it may seem at first that the SPH discretization is coarser, the actual discretization density

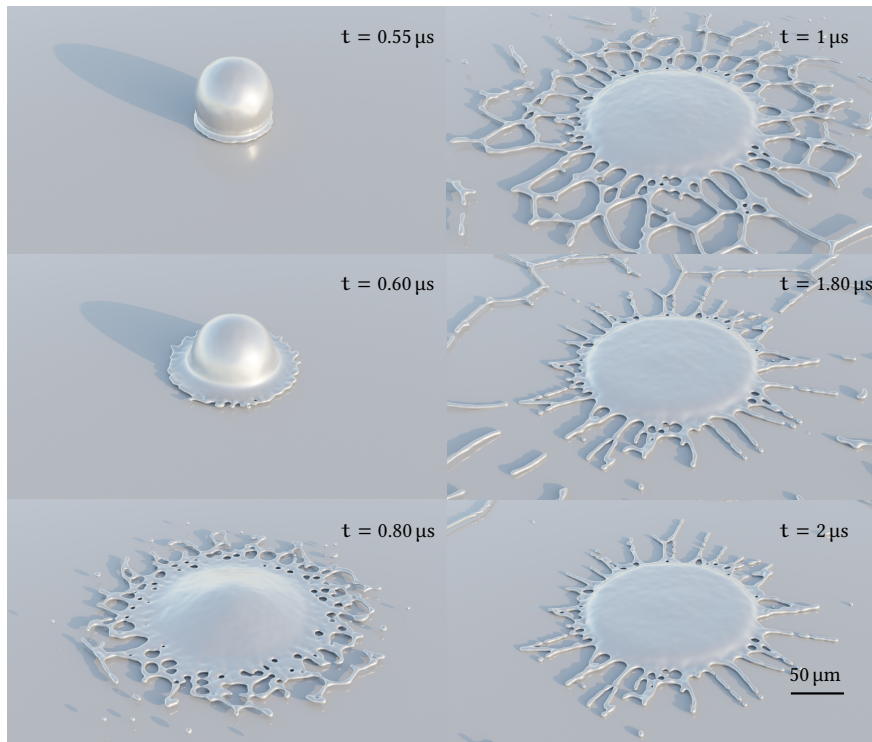


Figure 5.14. Ray-traced rendering of the droplet impact dynamics simulated with SPH.

in the region of interest, i.e., in the droplet, was $\sim 2.8^3 \approx 22$ times higher than the discretization density used by Ansys Fluent, whose mesh edge length was $2.25 \mu\text{m}$, corresponding to 2.8 times the particle diameter of $0.8 \mu\text{m}$. At the same time, our SPH method was able to finish the simulation in roughly 5 minutes, while Ansys Fluent required roughly 20 minutes. This is a remarkable result, as it allows a significant refinement in the region of interest, while reducing the simulation time by a factor of 4. All the more interesting are the scaling implications for the simulation of multiple droplet impacts. The increased performance allows for faster iteration times when simulating multiple droplets as well as significantly more accurate resolution of gaps in the coating on the scale of $< 1 \mu\text{m}$.

Our evaluation also showed that the selection of the liquid fraction for contouring of the ceramic phase has a significant impact on droplet diameter, and especially on droplet height. This has to do with the dispersion of the liquid surface that is ever-present for FVM-simulations using the VOF approach and adds an additional restriction on the mesh resolution. Using our SPH method, we completely avoided this dispersion issue and obtained a high-resolution simulation with a clearly defined surface. A ray-traced rendering of a surface reconstruction of the SPH particle data is shown in Figure 5.14.

Finally, we extended our SPH simulations to explicitly discretize the substrate and to take temperature-dependent material parameters into account. Using this setup, we simulated three droplet impacts with different initial diameters and showed good agreement of the simulated spread factor with the analytical expression proposed by PASANDIDEH-FARD et al. [1996].

5.5 Conclusion

In this chapter, we have shown that it is possible to simulate molten droplet impact in the thermal spray process using nearly identical physical parameters in both an SPH discretization and an FVM (finite volume method) discretization in Ansys Fluent. We performed a quantitative analysis of the simulations by considering droplet height, diameter, and velocity distributions over time. All of these showed good agreement, while the few disparities were isolated and explained. This extends the quantitative

SPH-versus-Eulerian validation from TIG welding (Chapter 4) to microscopic droplet impact.

We introduced a novel SPH model that uses implicit integration for all forces except surface tension. Because of this, our simulations remain stable for a wide range of large time steps. We also showed that our SPH method is an efficient and accurate alternative to the commercial FVM method in Ansys Fluent. Our SPH method is able to have a higher discretization density in the region of interest while only requiring a quarter of the simulation time. Methodologically, this also partially motivated the fully implicit treatment of surface effects developed in Chapter 3.

As a next step, the work in this chapter could be extended by considering multiple droplets of varying size and velocity. When considering multiple droplets, the gaps in the coating may also be evaluated to a higher degree of accuracy than was previously possible. A further extension could enable the simulation of multiple, only partially melted droplets of both varying size as well as varying ratio of solid material at the core. Furthermore, the consideration of rough surfaces and resulting contact angles could also be interesting for future work. These findings and their relation to the broader thesis contributions are summarized in Chapter 8.

Part II

Particle-Based Deep Learning

Foundations of Deep Learning

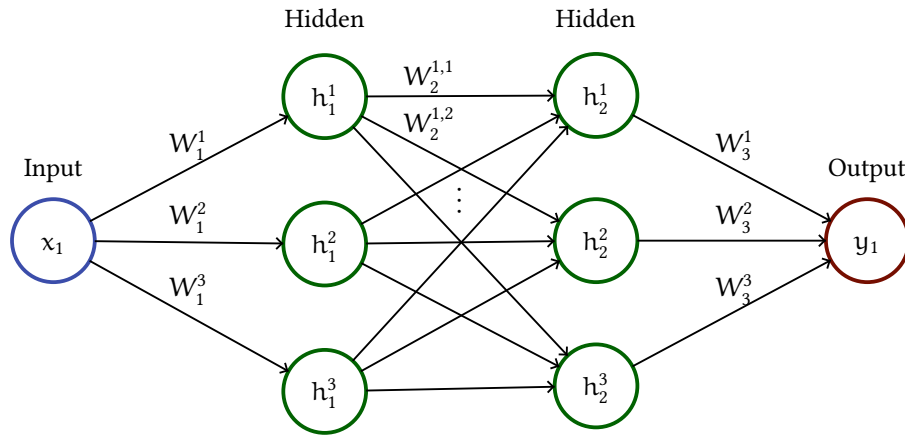


Figure 6.1. A simple multi-layer perceptron (MLP) with one input, two hidden layers of width 3 and one output. All layers are fully-connected.

The previous part of this thesis dealt with fluid simulation using particle representations. This underscored the versatility of how particles can be used in practical applications. In contrast, this part will focus on using particles in the context of deep learning. Specifically, how shapes represented by 3D point clouds can be efficiently processed and encoded into a learned signed distance field, without actually having knowledge of the ground truth shape. To provide some context, this chapter will focus on what deep learning fundamentally represents, how neural networks “learn” and what challenges need to be overcome when learning on three-dimensional data.

Machine Learning (ML) generally describes methods for fitting prediction models to data and it has always been a core field of computer science. Deep Learning (DL), the process of learning patterns from data using deep neural networks, is a subset of machine learning in which enormous progress has been made in recent years. The most recent and noteworthy milestone is the emergence of the transformer architecture [VASWANI et al. 2017] and its use in Large Language Models (LLMs). These are trained on vast amounts of accumulated human knowledge in the form of text and, when scaled to billions or trillions of parameters, exhibit human-like conversational capabilities. The capabilities notably include information retrieval, which makes them immensely useful in many different fields.

The first part of this chapter will focus on introducing Multi-layer Perceptrons (MLPs), which are the fundamental building blocks of general DL methods. Further, it will be outlined how MLPs are

“trained” by adjusting learnable weights through a process called stochastic gradient descent (SGD). This includes how to compute the derivatives needed for the SGD optimizer to be able to run. For these fundamentals, we will follow the foundational work of GOODFELLOW et al. [2016]. The second part of this chapter will give a brief overview of geometric deep learning, with applications to point clouds specifically, and will finally discuss the considerations that should be made when using points within Neural Networks (NN). This part generally follows the works of BRONSTEIN et al. [2021] and Y. XIE et al. [2022]. This will lead into Chapter 7, where a hybrid point and grid neural network architecture for shape reconstruction from point cloud inputs will be presented.

6.1 Multi-layer Perceptrons

Multi-layer Perceptrons (MLPs) are a fundamental building block of deep learning. Traditionally, they are also called feedforward neural networks. Both of these terms describe key characteristics of these networks, one being “multi-layer” which generally means that the computation can be split up into several discrete “layers” that have to be fully computed before the next layer can be evaluated. “Feedforward” refers to the fact that these networks only have forward connections and no connections to earlier parts of the computation. This is in contrast to “recurrent” neural networks which do contain connections to earlier parts of the network, which, prior to the emergence of transformer networks, were state-of-the-art for Natural Language Processing (NLP) tasks. The reader is referred to GOODFELLOW et al. [2016] for further detail as these kinds of networks will not be discussed here.

Now we will go through how to construct a simple neural network and then go into detail about how it is “trained”. A schematic view of a simple feedforward neural network with multiple layers, i.e., an MLP, is shown in Figure 6.1.

This MLP can be interpreted as a composition of functions, where the “depth” of the model is equal to the number of function compositions, three in this case. To be specific, the output y in this illustration is computed by

$$y = ((xW_1 + b_1)^T W_2 + b_2)^T W_3 + b_3, \quad (6.1)$$

which is an entirely linear function. Here W represent learnable parameters, typically called weights, and b represent learnable offsets called biases, which are often implicitly represented as part of the weights. Note, that the biases are not explicitly shown in Figure 6.1. Finally, x denotes the input to the neural network and y denotes the (scalar) output. Using only linear operations in this way causes the entire computation to collapse into a single linear operation with weight matrix $\widehat{W}$ and offset $\widehat{b}$

$$y = x\widehat{W} + \widehat{b}. \quad (6.2)$$

This is a common problem with linear models in machine learning and can be solved in two ways. One is to transform the features x using a specific non-linear function ϕ and fit a linear function in this transformed space. The second, and in deep learning more typical, option is to add a non-linear “activation function” in between individual linear operations. With increasing network depth, this allows the model to “learn” non-linear feature transformations within the model, by essentially preventing linear operations from “collapsing”. As an example, we add activation functions to Equation (6.1) to obtain

$$y = f^{(3)}(f^{(2)}(f^{(1)}(xW_1 + b_1)^T W_2 + b_2)^T W_3 + b_3). \quad (6.3)$$

As long as functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ are non-linear, this function can then represent non-linear functions as a whole. The final composition yields the output of the network, which becomes an interpretable quantity.

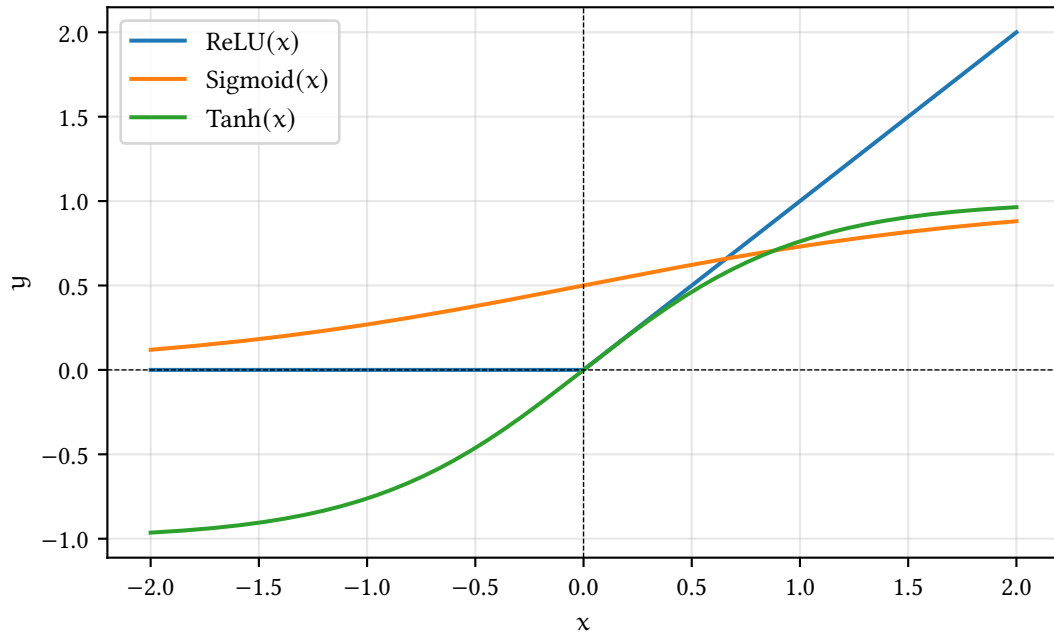


Figure 6.2. Commonly used non-linear activation functions in deep neural networks.

One of the simplest and most widespread activation functions is the so-called rectified linear unit (ReLU), see Figure 6.2. This is essentially a “barely” non-linear function given by

$$\text{ReLU}(x) = \begin{cases} x & \text{for } x > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (6.4)$$

which is linear for positive values and zero for negative values. This simple function is sufficient for deep neural networks to represent highly non-linear functions and is therefore still widely used at the time of writing. Other commonly used functions are, e.g., the sigmoid or tanh functions which are also depicted in Figure 6.2. The sigmoid function output is bounded from 0 to 1 and tanh from -1 to 1. Unfortunately, their gradients can saturate, meaning that the gradient will be approximately zero for very large and very small inputs. This can become a problem when using gradient-based optimization as this effect cancels out any meaningful gradients for these value ranges. In contrast, the ReLU function always either has a gradient of one or zero. While modern optimization methods mitigate this problem to a large degree, it should still be taken into account when choosing appropriate activation functions.

All intermediate states are not necessarily interpretable and typically only used in the process of computing the output value. These states are also called “hidden” states. Regardless of the output dimension of the network, the hidden layers are typically vector-valued. The dimension of the hidden layer vectors is called the width of a network. The hidden layer vectors are often interpreted as many individual “neurons” that take all inputs from the previous layer and compute a single output value through linear combination with adjustable weights. This linear combination is then passed through an activation function. The term “neuron” is inspired by how biological neurons work. However, it should be noted that neural networks do not make it their goal to model how the human brain works, but instead aim to achieve the best-possible statistical generalization. The complexity, and to a certain degree the representational ability, of the neural network composite function is given both by the depth and width of the network. Deeper and wider networks also generally have a larger number of parameters, which is why the number of parameters of a network is generally indicative of the

maximum complexity of the trained function. State-of-the-art open-source neural networks in language modeling are documented to have almost a trillion trainable parameters [DEEPSEEK-AI et al. 2025], with that number expected to already be exceeded by closed-source models. The next section will briefly describe how parameters are algorithmically adjusted to fit a specific goal.

6.2 Training

The process of adjusting the functions of the network to achieve a specific goal is typically referred to as “training”. Training can be achieved by presenting data to the network and prescribing conditions on the output. If a specific value is known and enforced during training, this is called “supervised” training. If some values are known and some are unknown, it is called “semi-supervised,” and if all are unknown, it is called “unsupervised.” In the case of unsupervised learning, constraints typically describe the properties of a valid solution without directly enforcing specific values. One very common example of unsupervised learning is called autoregressive. This means that the network learns to perfectly reconstruct its input. Such a task sounds trivial, but is often combined with a lossy memory “bottleneck” within the neural architecture, such that the model is forced to use patterns in the data to more efficiently store the input and be able to reconstruct it after the bottleneck. It can be used as a pretraining step that is then followed by more specific supervised learning tasks, e.g., classification or segmentation of images. Another example of unsupervised learning that is gaining more popularity is the so-called physics-informed loss [RAISSI et al. 2019, SITZMANN et al. 2020b, TANCIK et al. 2020]. This loss uses autograd functionality of the learning framework to compute derivatives, which are in turn penalized with the residual of a differential equation. In this way, physical operators can be learned without having access to data that is costly to compute, or can even reconstruct physical fields from sparse measurements.

To explain the general training process, we will consider the supervised setting. Note, however, that the concepts of supervised training easily translate to the other cases. Supervised training starts with having a set of data points $\mathcal{D} = \{(\mathbf{x}_i^*, \mathbf{y}_i^*)\}_{i=0}^n$, which are assumed to be drawn independently and identically distributed from an underlying data distribution $\mathcal{P}$ defined over the data domain $\mathbf{x}_i^* \in \mathcal{X}^* \forall i$ and label domain $\mathbf{y}_i^* \in \mathcal{Y}^* \forall i$. In addition, there is assumed to exist an unknown function f^* for which $f^*(\mathbf{x}_i^*) = \mathbf{y}_i^*$ for all $i \in [0, \dots, n]$. The supervised learning problem reduces to finding a parametrized function f_θ that approximates the unknown function f^* as closely as possible by using data pairs $\mathcal{D}$ to adjust the network parameters θ accordingly. But how do we quantify how close our approximation is to the “true” function? One of the most common ways is to use the L2 loss, also called the Mean Squared Error (MSE) loss function:

$$\mathcal{L}_{\text{MSE}}(\mathbf{x}^*, \mathbf{y}^*) = \frac{1}{n} \sum_{i=0}^n \|\mathbf{y}_i^* - f_\theta(\mathbf{x}_i^*)\|^2, \quad (6.5)$$

which is the average of squared differences between the neural network output and ground truth label values. It is a convex function, whose minimum is achieved iff $\mathbf{y}_i^* = f_\theta(\mathbf{x}_i^*)$, $\forall i$. There are other loss functions, but the L2 loss is frequently used as it more strongly penalizes outliers than values already close to the desired solution. This often gives a more desirable solution for a network’s weights than, e.g., the L1 loss, which only penalizes the absolute value of deviation.

6.3 Backpropagation and Stochastic Gradient Descent

The loss function can also give us information about how the parameters θ , i.e., the weights of the network, need to be changed to reduce the loss and yield a closer approximation of the desired function.

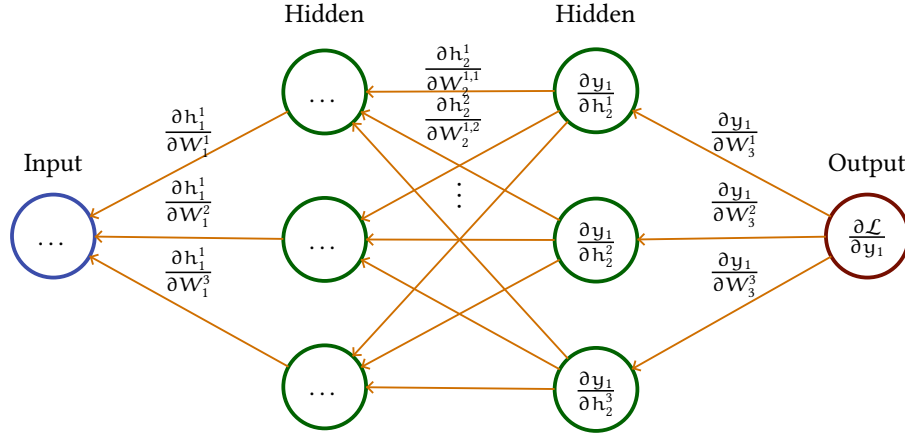


Figure 6.3. The backpropagation algorithm as used frequently in deep learning frameworks. The computational graph of the forward pass (Figure 6.1) is traversed in reverse order, and derivatives of the loss with respect to individual parts of the network are accumulated. The dots denote summations over the respective partial derivatives.

Specifically, the weights can be adjusted in the direction of the negative gradient of the loss function with respect to the network parameters $-\frac{\partial \mathcal{L}}{\partial W_{i,j}^{k,j}}$. This expression can be expanded using the chain rule, such that a single “backward-pass” of the computational graph is sufficient to compute all the derivatives. However, this is only true for scalar loss values, which is why multiple losses are generally aggregated by summation or weighted averaging. The backward pass for the computational graph of Figure 6.1 is shown in Figure 6.3. For example, the expression for $\frac{\partial \mathcal{L}}{\partial W_3^3}$ can be expanded using the chain rule to

$$\frac{\partial \mathcal{L}}{\partial W_3^3} = \frac{\partial \mathcal{L}}{\partial y_1} \frac{\partial y_1}{\partial W_3^3}, \quad (6.6)$$

where for $\mathcal{L} = \mathcal{L}_{\text{MSE}}$

$$\frac{\partial \mathcal{L}}{\partial y_1} = 2(y_1^* - y_1), \quad (6.7)$$

and

$$\frac{\partial y_1}{\partial W_3^3} = f^{(3)'} h_2^3. \quad (6.8)$$

The last expression is yet another application of the chain rule to Equation (6.3), where $f^{(3)'}$ is the derivative of the last activation function with respect to its input and $h_2 = f^{(2)}(f^{(1)}(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)^T \mathbf{W}_2 + \mathbf{b}_2)$, such that h_2^3 is the third component of this expression.

As mentioned previously, having the derivatives of the loss function with respect to the network parameters allows us to adjust the parameters in a way that reduces the loss. This process is called gradient descent and modifies the weights using

$$W_{i,j}^{\text{new}} = W_{i,j}^{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial W_{i,j}^{\text{old}}}, \quad (6.9)$$

where η is called the “learning rate” and is often in the range of $\eta \in [0.0001, 0.1]$. Since the available data is not presented to the network all at once but instead in smaller, and typically randomly selected so-called “batches”, this process is called stochastic gradient descent (SGD) and is known as the most straightforward and reliable way to optimize the weights of neural networks. Nevertheless, the Adam [KINGMA and BA 2017] and AdamW [LOSHCHILOV and HUTTER 2019] optimizers are a lot more popular nowadays, mainly due to their superior speed and general robustness.

These derivatives are not computed by hand in practice. Instead, a process called automatic differentiation is used in almost all deep learning frameworks today. During network evaluation, a

computational graph (a directed acyclic graph to be specific) is stored automatically, representing all computations. At the time when derivatives need to be computed, this graph is traversed in reverse order and so-called adjoint mode derivatives computed. For scalar loss functions this only requires a single backward pass to compute all derivatives of the loss function with respect to the network parameters. Recently, however, popular frameworks such as PyTorch [PASZKE et al. 2019] started implementing ahead-of-time compilation for forward and backward code, which can considerably improve runtime and reduce memory usage, allowing larger models to be trained efficiently. For more information on automatic differentiation, the reader is referred to the book by NAUMANN [2012].

6.4 Deep Learning on Particles

2D concepts Deep learning on high-dimensional data is fundamentally very challenging. Already in 3D Euclidean space it becomes very difficult to learn meaningful features and representations, due to the sheer variety of possible shapes. However, geometric deep learning as described by BRONSTEIN et al. [2021] teaches us that we can leverage regularities, i.e., symmetries in the domain, by imposing geometric priors to improve the learning process. This has been used in the field of computer vision for image processing for decades. The convolutional neural network (CNN) architecture with its shared weights is a translation-equivariant operator, meaning that the output shifts along with any input translations. Local pooling operators separate different scales, increasing the “receptive field” of the network. Global pooling is entirely translation invariant, meaning that the output is always the same, regardless of any input translations. All of these components together impose a strong geometric prior that allows CNNs to learn meaningful features in 2D image space. Accounting for translational invariance/equivariance within network architecture essentially reduces the feature space and functionally “maps” different data to the same features.

Transfer to 3D While this is a well-understood concept for 2D Euclidean geometry (images), it becomes exponentially harder for 3D geometry. Already the increase in memory usage when storing dense 3D feature grids can make working on neural networks for 3D processing prohibitively expensive. In addition, the geometric representation of 3D geometry is not always given by grids, but is frequently given as point clouds, originating, for example, from LiDAR scanners. While this is often a more memory-efficient representation since only non-empty space is represented in memory, it comes with its own set of challenges. Namely, how do we apply the geometric deep learning blueprint of combining translation invariant, translation equivariant and scale-separating operations on point clouds? We also face another problem, namely that the ordering of the points in the point cloud can possibly affect the output of the neural network. As such, we face the extra challenge of ensuring that permutation of the input points does not affect the output of the neural network.

Point-network components To repeat the problem statement more plainly: how can a neural network that traditionally processes grid-based data of a fixed dimension be generalized so that it can process unordered point clouds of varying sizes? How can it be ensured that passing the points in a different order to the network results in a correspondingly permuted ordering of output values? For this Qi et al. [2017b] came up with three simple operations, which are shown in Figure 6.4.

The first, shown in Figure 6.4 (a), consists of the steps “Subsample” and “Pooling”. Subsample selects a specific number of points from the input, e.g., by randomly sampling a fixed ratio of points from the input set. Qi et al. [2017b] also suggest using what they call Farthest-Point Sampling (FPS), which is supposed to select a number N of the most meaningful points. With this selection (orange points), all other points are assigned to at least one orange point based on, e.g., a radius graph. The features

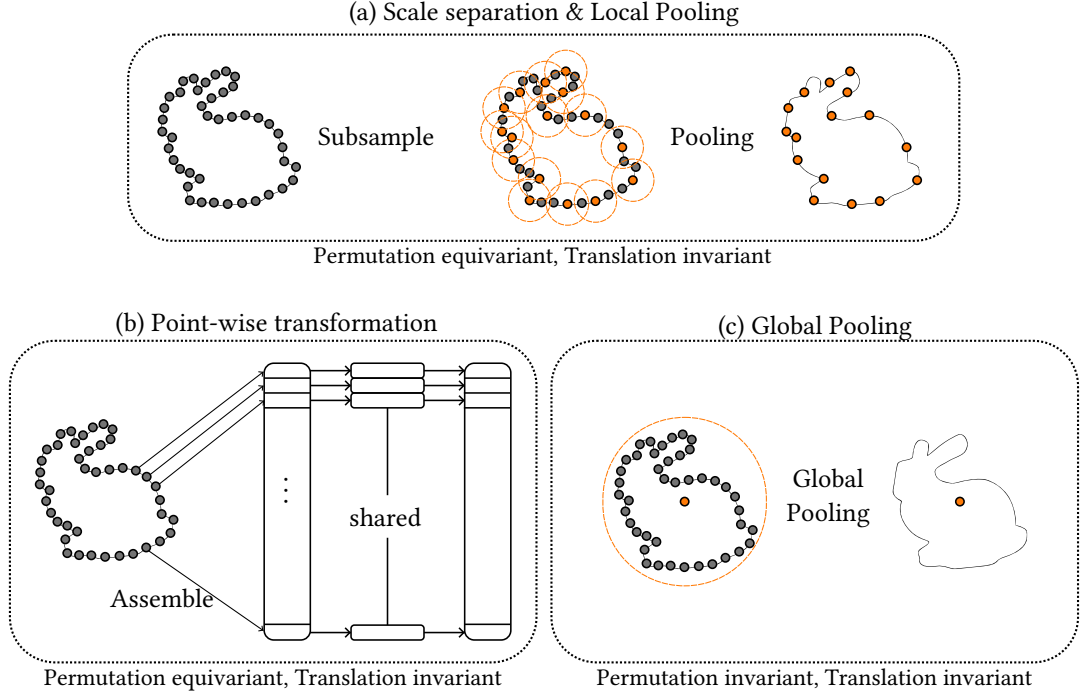


Figure 6.4. *Neural Network components for imposing geometric priors on point cloud inputs.*

from the other points are then collected into the orange points by a permutation invariant aggregation method, such as summation, averaging or simply taking the maximum value for each feature dimension. This operator is permutation equivariant, because the ordering of the output changes alongside the ordering of the input. It is translation invariant, since only relative positions are taken into account. This operator increases the “receptive field” of the networks, because each point effectively represents a whole region of points, thereby changing the viewing scale on the data.

The second operator in Figure 6.4 (b) is a simple point-wise transformation. It simply applies the same function to each input point individually. As such, it is trivially permutation equivariant, such that the output ordering changes alongside the input ordering. It is also translation invariant, as long as no absolute positions are used as features within the network. This operator is the traditional learning “backbone” of point networks, containing the majority of learnable parameters.

The final operator in Figure 6.4 (c) is the global pooling operator. In contrast to the local pooling operator in (a), it simply aggregates all the points into a single point, again using, e.g., summation. This operator is trivially both permutation invariant and translation invariant, since the ordering of points and their absolute translation has no impact on the output. The drawback of this operator is that it frequently underfits the data [BUTEREZ et al. 2022]. In other words, this means that a lot of information is lost in the global pooling process. Networks using the features from global pooling struggle to accurately use patterns in the data in their respective tasks.

Generalization to message passing The different approaches shown in Figure 6.4 can be unified in the “message-passing” framework, such as the one proposed by FEY and LENSSEN [2019], which is shown in Figure 6.5. The purpose of this framework is to provide a unified way of describing operations on point clouds and graphs, such as the ones described above. This shared mathematical notation is given by

$$\mathbf{x}_i^{\text{new}} = \gamma \left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \phi(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{i,j}; \theta_\phi); \theta_\gamma \right), \quad (6.10)$$

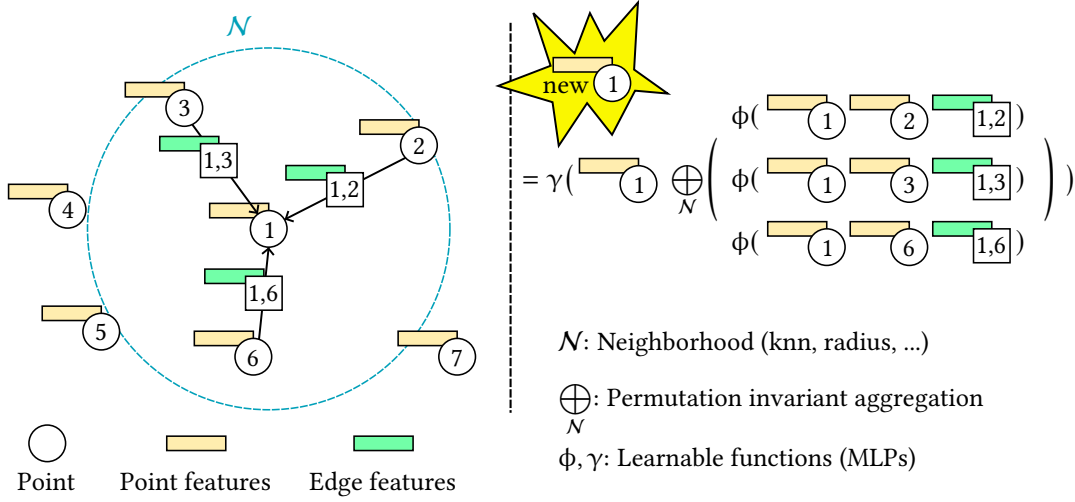


Figure 6.5. Generic message passing mechanism as employed for graph neural networks (GNNs). After constructing a connectivity, such as k -nearest neighbors (k -NN) or radius graph, this can also be used for learning on point clouds.

where $\mathbf{x}_i$ denotes the point feature of point i and $\mathbf{e}_{i,j}$ denotes the (optional) edge feature between nodes i and j . In addition, $\gamma(\cdots; \theta_\gamma)$ and $\phi(\cdots; \theta_\phi)$ denote learnable functions parametrized by θ_γ and θ_ϕ , respectively. Finally, $\mathcal{N}_i$ denotes the neighborhood of point i , which could, for example, be obtained through a k -nearest-neighbor (k -NN) or radius-neighbor query algorithm. Neighborhoods are typically stored as edges of a graph, i.e., pairs of point indices for easier (batch) processing. All of the operations displayed in Figure 6.4 can be expressed using the message passing paradigm. The local (a) and global (c) pooling operations can be entirely expressed by defining the neighborhood $\mathcal{N}$ and using the respective aggregation operator without any learnable networks. The point-wise transformation can be expressed by simply not using a neighborhood, i.e., $\mathcal{N} = \emptyset$ and defining $\mathbf{x}_i^{\text{new}} = \gamma(\mathbf{x}_i; \theta_\gamma)$, so that the new point features only depend on its own previous point features. Nevertheless, the message passing paradigm allows much more sophisticated notions of graph learning to be expressed and implemented.

The attentive reader might also be able to draw a parallel between the message-passing framework and the SPH approximation presented in Chapter 2, particularly Equation (2.14). In fact, the SPH function approximation can also be written using message passing, by setting

$$\bigoplus = \sum_{j \in \mathcal{N}_i}, \quad (6.11)$$

$$\phi(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{i,j}; \theta_\phi) = \frac{m_j}{\rho_j} f_j W(\mathbf{x}_i - \mathbf{x}_j; h), \quad (6.12)$$

$$\theta_\phi = h, \quad (6.13)$$

$$\gamma(\mathbf{x}_i, \mathbf{y}_i; \theta_\gamma) = \mathbf{y}_i. \quad (6.14)$$

Abusing notation a little, the features $\mathbf{x}$ are briefly reinterpreted as the concatenation of position $\mathbf{x}$, mass m , density ρ , and function f . This should simply serve as an illustration of the versatility of the message passing framework and has no further relevance with respect to SPH simulation.

Alternatives to point-based networks While research into point-based deep learning, and more generally graph-based deep learning, is still ongoing, another option is to simply discretize the point-based data onto a regular voxel grid. This is done frequently in practice, and can be as simple as initializing an indicator function, 0 for empty and 1 for filled, on a voxel grid underlying the points, as is shown in Figure 6.6. The decisive advantage of this approach is the ability to then use the

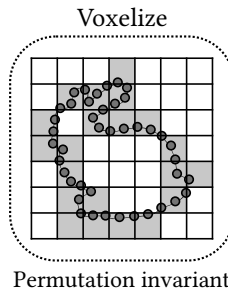


Figure 6.6. *Voxelization of point features onto a background grid.*

plethora of well-established grid-based neural network architectures for further processing. In addition, networks utilizing dense 3D grids are typically very fast to evaluate, since the underlying dense matrix multiplication operations are computational workloads that are particularly well suited for modern graphics cards. They follow predictable storage access patterns and constitute uniform computational loads, ideal for massively parallel execution. However, this can quickly result in memory issues, as storing learned features on multiple dense 3D grids can become prohibitively expensive.

To become more scalable, there are multiple possible solutions that can be investigated. For one, adaptive hierarchical data structures can be used, such as sparse voxel octrees [TAKIKAWA et al. 2021, P.-S. WANG et al. 2022]. The challenge then becomes to construct, store, and process these sparse data structures efficiently on accelerator hardware (GPUs). Another option is to forego the use of grids entirely and simply work purely on point data. The drawback to this, however, is that pure point-based networks generally have worse performance than grid-based networks and show slightly inferior results on average [Y. XIE et al. 2022]. The natural remaining option — and also the approach that we will investigate in Chapter 7 — is to combine point-based networks with a few dense grids on multiple levels of detail in a hybrid architecture. By doing this, one could combine the benefits of both representations while mitigating the drawbacks.

6.5 Encoder-Decoder Networks

Finally, a specific network architecture relevant to our work, namely the encoder-decoder network, will be briefly presented. Figure 6.7 shows this type of network conceptually. As the name implies, it consists of two main components, namely the encoder and the decoder. Individually, they consist of smaller building blocks, such as the point-network components discussed above. The goal of the encoder is to take the input of the network, and to transform it into a so-called “latent” representation. In other words, the encoder should compress the input data, typically into a fixed-size vector. The decoder then takes the latent representation as input and decodes it into the final output. For images, this could be, e.g., per-pixel classification or global image classification. Figure 6.7 shows a special case for the encoder-decoder, called an autoencoder. This implies two properties. First, that the encoder-decoder is trained in an unsupervised manner, and second, that it should exactly reproduce the input data.

While it might sound nonsensical to train a neural network to simply reproduce what is already known, the really interesting part here is the space of latent representations, the so-called “latent space.” This contains a compressed view of our data distribution and can be used in a variety of ways. For instance, latent spaces can be sampled to generate new data points that adhere to the patterns set forth by the training data, thus generating, for example, previously unseen shapes or images. Latent spaces of multiple data instances can also be combined or interpolated to result in smooth transitions between previously discrete points. Finally, the latent representation can be used as a starting point for solving further downstream tasks. In other words, the encoder is used as a fixed component to train a different

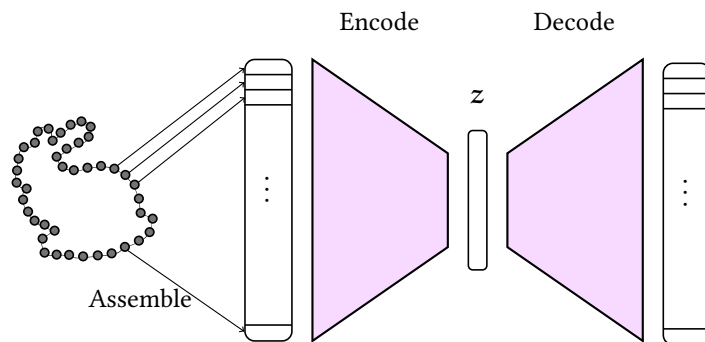


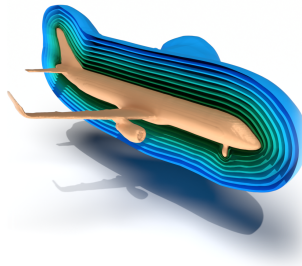
Figure 6.7. Encoder-Decoder network architecture for point-based networks. The point data is compressed by the encoder into a latent representation z of fixed dimensionality and subsequently decompressed by the decoder.

network using the latent code. Overall, encoder-decoder networks are a highly relevant and useful architecture for applying deep learning to practical problems. More detail on autoencoders can be found in Chapter 14 of the book by GOODFELLOW et al. [2016].

6.6 Summary

In this chapter, we looked at the fundamental concepts of deep learning. Specifically, we have seen how neural networks at their core are constructed, evaluated, and trained to fulfill a specific goal. Namely, backpropagation and gradient descent are the core mechanisms by which all neural networks are trained. Furthermore, there is a large variety of other neural network components, such as activation functions, optimizers, loss functions, and general network architecture that also substantially impact the potential performance of a neural network. We also looked at the task of learning on point clouds in particular. For this kind of data, it is of interest to derive learning operators that can leverage regularities in the training data to improve the learning process. To this end, learning operators that are translationally invariant and permutation equivariant/invariant were described, which emulate well-studied convolutional operators for grid inputs. However, arbitrary geometric data is challenging to work with and the development of neural network operators on this kind of data is still an area of active research [BRONSTEIN et al. 2021]. Finally, we have briefly introduced a simple but very influential neural architecture called the encoder-decoder, which will be used in the next chapter.

HYVE: Hybrid Vertex Encoder for Neural Distance Fields



Algorithms processing 3D geometric data have become omnipresent and an integral part of many systems. These include, for example, systems evaluating LiDAR sensor data, game engines, 3D asset visualization, and physical simulation used in engineering prototypes. In recent years, deep learning methods have been increasingly investigated to assist in solving problems pertaining to 3D geometry. Within this thesis, this chapter addresses the second main objective defined in Chapter 1, namely improving the efficiency of point-based representations in a data-driven way. The required background on deep learning and point-based learning is summarized in Chapter 6.

Neural shape representation is the task of using neural networks to predict shape occupancies or surface distances at arbitrary spatial coordinates. Recent works have shown the ability to capture intricate details of 3D geometry with ever-increasing fidelity [LONG et al. 2023, Y. WANG et al. 2022]. However, a significant number of such works employ an auto-decoder-based architecture, which requires solving an optimization problem when representing new geometry. Additionally, the auto-decoder still has to be evaluated for all query points individually, which can become very costly when evaluating these systems for high-resolution reconstruction [Y. XIE et al. 2022]. Finally, many of these methods also require annotated and densely sampled ground truth data. Meta-learning approaches, e.g., by SITZMANN et al. [2020a] and OUSFI and BOUKHAYMA [2022], mitigate this problem, but also have to run several iterations of gradient descent to specialize the network for each new model before inference. Encoder-decoders can instead encode the shape in a single forward pass [BOULCH and MARLET 2022, CHIBANE et al. 2020a, B. ZHANG et al. 2023], and typically employ computationally cheaper decoder networks for evaluating query points. Nevertheless, these approaches also often require elaborate data preprocessing pipelines, and rely on labeled training data.

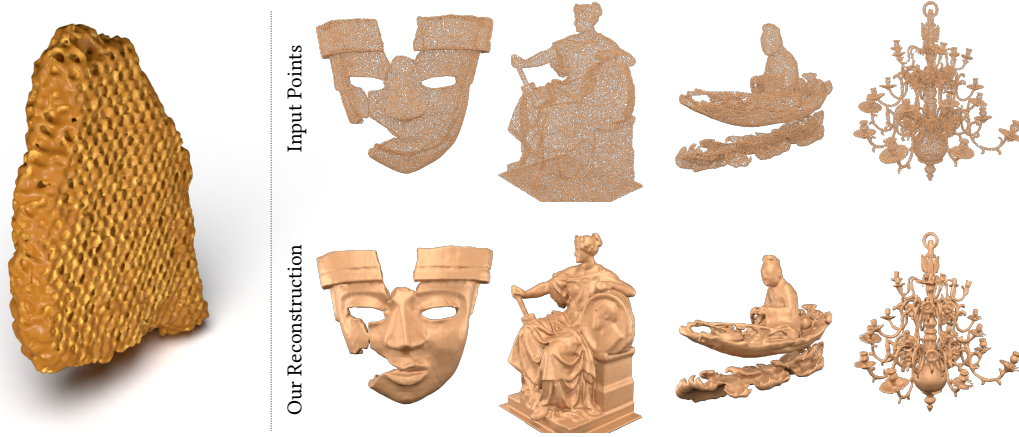


Figure 7.1. The reconstruction of a 3D scan of a beehive (left) and comparisons between input points and our corresponding reconstructions on the Objaverse dataset (right) [DEITKE et al. 2023]. These examples show the capability of our neural distance field encoder to capture a large amount of detail in a single forward pass, using only oriented point clouds as input.

One of the core motivations for this work was to develop a neural distance field-based method that could be used for signed distance computation of both rigid and deforming shapes. Both are represented as points and the latter can change often, requiring frequent recomputation of the distance field. The deformations and dynamics of these shapes are not explicitly learned by the network but instead implicitly captured through geometric changes in the input shape. As a result, we wanted to keep the decoder network computationally efficient and, instead of latent optimization, we wanted to develop an encoder capable of generating fast and accurate latent codes for this decoder network. The encoder-decoder architecture specifically is briefly presented in Section 6.5.

Therefore, we propose an end-to-end learnable encoder-decoder system that is not bound by previous data preprocessing constraints and can be trained using only the zero-level set, i.e., surface samples as labeled data, which are readily available with a minimal amount of preprocessing. This kind of training was previously introduced for auto-decoders [SITZMANN et al. 2020b], and is enabled by using the eikonal equation as a training target. Similar approaches have been attempted for encoders by ATZMON and LIPMAN [2020a,b], using global encoders augmented by latent-optimization at test-time. Yet to the best of our knowledge, this kind of training is not well-established using encoders without latent-optimization.

We summarize our contributions as follows:

- We derive an encoder architecture for representing 3D shapes. The key component of this encoder is the hybrid and interleaved execution of graph-level convolutions and 3D grid convolutions, as well as back-and-forth feature projections. Our motivation for this hybrid approach is the ability to accurately *extract* information from surface points and to efficiently *process* information in grid (Euclidean) space.
- We introduce a novel way of voxelizing point features within network architectures, in which the pooling operator is replaced by projection of the entire “feature-field” onto the grid. We show that this results in less-noisy and more detailed surfaces, particularly when point normals are used as additional input.
- We show that the accuracy of our architecture is intuitively controllable. Using a model with as little as 38K parameters (including the decoder) can already achieve excellent visual quality while

being very fast to evaluate. This makes it potentially useful for practical applications within resource-constrained computing platforms.

- We show the feasibility of training encoder-decoder networks on the eikonal equation for high-fidelity 3D shape encoding. We also propose a simple yet effective modification to the loss function that can gracefully handle poorly oriented surface normals in the training data, e.g., caused by non-manifold or non-watertight geometry.

In the evaluation, we show that we are able to reconstruct better quality surfaces than other state-of-the-art methods. Sample reconstructions of our method are shown in Figure 7.1.

7.1 Related Work

Neural fields have become an integral part of research in geometric deep learning, with hundreds of papers published in recent years. A comprehensive overview is given by Y. XIE et al. [2022]. One of the seminal works on deep learning for unstructured data is the introduction of PointNet [QI et al. 2017a]. From today’s perspective, one of the major limitations of this work is the difficulty of learning high-frequency functions from low-dimensional data [RAHAMAN et al. 2019, XU et al. 2019]. The solution to the problem is addressed by more recent approaches such as NeRFs [MILDENHALL et al. 2020] and Fourier Feature Networks [TANCIK et al. 2020]. In essence, the idea is to use positional encoding, inspired by the embeddings proposed by VASWANI et al. [2017] for transformer networks. These embeddings compute a mapping from low dimensional positional information (typically 2D or 3D) into higher dimensional spaces, e.g., using a specific number of Fourier basis functions [TANCIK et al. 2020]. A concurrent work shows that using periodic activation functions inside an MLP also significantly improves reconstruction quality and surface detail [SITZMANN et al. 2020b], the single layer of which can again be seen as a kind of positional encoding [Y. XIE et al. 2022]. Subsequent works improve the usage of positional encodings, e.g., by controlling the frequency through a feedback loop [HERTZ et al. 2021] or modulating the periodic activations using a separate ReLU-activated MLP [MEHTA et al. 2021]. Other benefits of using periodic activations are the ability to better learn high-frequency mappings and the continuous differentiability of these activations which is useful for evaluating network derivatives as training targets [P. Y. CHEN et al. 2023, RAISSI et al. 2019, Y. WANG et al. 2022].

There are many approaches for representing 3D shapes using neural networks. For clarity of exposition we will classify them into global-prior and local-prior based methods. In addition, we will discuss overfitting-based methods, general unsupervised shape encoding and neural scene representation. Finally, we will differentiate our work from previous works.

Global methods These do not make use of geometric structures in the network itself, and can generally be used irrespective of the discretized representation of the geometry. Typically auto-decoder methods, in which the latent representation is optimized during training and testing, are in this category [ATZMON and LIPMAN 2020a,b, GROPP et al. 2020, MEHTA et al. 2021, MESCHER et al. 2019, PARK et al. 2019, SITZMANN et al. 2020b, Y. WANG et al. 2022]. The network can then be queried using both the latent feature and a 3D coordinate to evaluate either a distance metric or an occupancy value. Meta learning approaches also fall into this category. A few iterations of gradient descent are used to specialize the weights of a generalized network to a new shape [OUASFI and BOUKHAYMA 2022, SITZMANN et al. 2020a]. An approach that has both discretization-dependent and independent components is presented by P. Y. CHEN et al. [2023], where the discretization-dependent encoder is typically discarded during inference. The amount of encoded details by these methods is naturally bounded by the number of

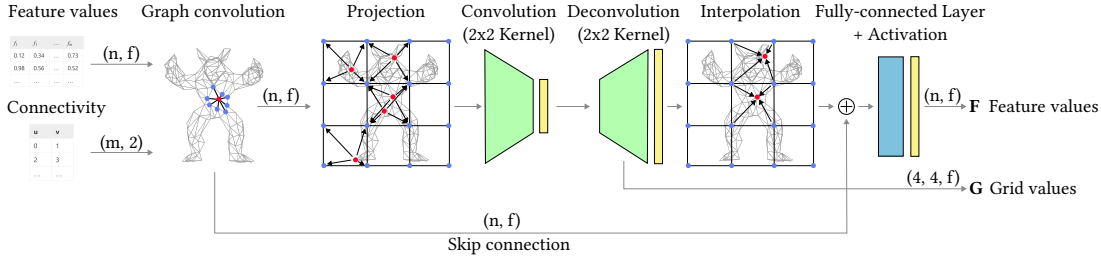


Figure 7.2. Convolution block that extracts features for a specific grid resolution. For clarity of illustration, a 2D rather than 3D grid is shown here. The input is a set of vertices (with position / feature data) and edges (encoded as vertex indices). The $+$ denotes element-wise vector addition. The block has two outputs, feature values on the vertices and grid values for each grid cell. For all resolutions, a 2×2 convolution kernel is used. n : number of vertices. m : number of edges. f : number of features (on the first level, the features are the spatial coordinate of each vertex).

network weights. Unfortunately, pooling-based set encoders for global conditioning frequently underfit the data [BUTEREZ et al. 2022].

Local methods This group of methods typically relies on using spatial structures, namely spatial grids or point clouds, within the network itself for the extraction of meaningful information [BOULCH and MARLET 2022, CHABRA et al. 2020, CHIBANE et al. 2020a, HUANG et al. 2022, 2023, C. JIANG et al. 2020, LOMBARDI et al. 2019, PENG et al. 2020, REN et al. 2024, TANG et al. 2021, P.-S. WANG et al. 2022, WILLIAMS et al. 2022, B. ZHANG et al. 2022, 2023]. This is a valuable approach since it is quite difficult for neural networks to encode the high-frequency functions needed to represent detailed fields in 3D. Previous works make efficient use of discretized structures, e.g., point clouds, meshes or voxel-grids as either inputs or outputs [BUTEREZ et al. 2022, HERTZ et al. 2021, QI et al. 2017a, TAKIKAWA et al. 2021]. There are also exceptions using other discretized structures. One such work uses triplanes instead of volumetric grids for latent diffusion 3D generation tasks [GUPTA et al. 2023], while another makes use of grid “patches” representing local geometric features that are merged through a global octree structure [LIN et al. 2023]. A similar approach is used by YARIV et al. [2024], which uses patches of small volumetric grids covering the object surface. For encoder-type methods, extracting local features generally improves network performance over global ones.

More recently, kernel-based methods combine local features in spatial structures with an inference-time sparse linear solve [HUANG et al. 2022, 2023, WILLIAMS et al. 2022]. This allows local methods to more strictly adhere to the input point cloud and scale to larger inputs, at the cost of the linear solve in addition to inference with a “convolutional-backbone” network. Finally, T. MÜLLER et al. [2022] propose an efficient way to use sparsity through multiresolution hashing and demonstrate its potential applicability for many of the local methods discussed here.

Unsupervised shape encoding A number of recent works investigate the ability to train networks for shape representation without direct supervision [ATZMON and LIPMAN 2020a,b, C. CHEN et al. 2025, CHIBANE et al. 2020b, GROPP et al. 2020, S. LI et al. 2025, LONG et al. 2023, MA et al. 2021, PENG et al. 2021, SITZMANN et al. 2020b]. Most of these works focus on auto-decoders and generally use optimization of latent codes during inference, while others “overfit” small networks to specific shapes [MA et al. 2021]. Some also use unsigned distance fields, which enables training on non-manifold geometry but increases reconstruction difficulty. While TANG et al. [2021] also use sign-agnostic optimization of occupancy fields, they still require ground-truth occupancy values for pre-training. Notably, GROPP et al. [2020] introduce the formulation of the unsupervised eikonal loss, which was further refined in the work of

SITZMANN et al. [2020b]. In our work we extend this loss to improve training on data with inconsistent normal orientations.

Neural scene representations and diffusion-based 3D generation Apart from our focus on shape reconstruction, recent neural scene representations reconstruct scenes from posed multi-view images. Prominent approaches include neural implicit fields, often neural radiance fields (NeRFs) [BARRON et al. 2023, FRIDOVICH-KEIL et al. 2023, Z. LI et al. 2023, MILDENHALL et al. 2020] and explicit primitive-based methods such as 3D Gaussian splatting [Y. JIANG et al. 2024, KERBL et al. 2023, G. WU et al. 2024]. The dominant objective is photorealistic novel-view synthesis and interactive editing. While geometric reconstruction is possible (e.g., implicit-surface methods like Neuralangelo), most pipelines prioritize rendering quality and speed over watertight mesh extraction.

For generating 3D shapes or assets from text or images, diffusion models have become the most common paradigm [CHENG et al. 2023, Z. WANG et al. 2023, ZHAO et al. 2025]. Nevertheless, a recent autoregressive model shows competitive performance compared with prevalent diffusion models for shape generation [WEI et al. 2025].

Differentiation from Related Works Many previous works focus on either purely point-based or grid-based approaches with prior voxelization of input points. Early works by Z. LIU et al. [2019, 2022] explore the use of hybrid point-voxel architectures for object segmentation and classification and show promising results in terms of accuracy and efficiency. In the latter work, the authors particularly focus on the efficiency aspect, presenting a method to find an optimal neural architecture to fit into a specific computational budget, again with application to classification and segmentation of 3D scenes. In contrast to previous work, we expand the capabilities of hybrid approaches to the unsupervised 3D reconstruction task and explore important design decisions. This includes details on network construction and the often-overlooked point-to-grid feature transfer methods to maintain consistency between grid and points.

As opposed to other works operating purely on point-based data [BOULCH and MARLET 2022, B. ZHANG et al. 2023], we find the use of a grid beneficial in terms of both efficiency and accuracy. We also use graph-convolutions at the point-level, extracting information at the smallest possible scale, while P.-S. WANG et al. [2022] use graph-convolutions at the voxel-level, as a surrogate for sparse grid-convolutions. Our formulation also retains information about the local positioning of the points in the grid cells by *not* voxelizing the input as is common with purely grid-based approaches [CHIBANE et al. 2020a, REN et al. 2024, P.-S. WANG et al. 2022]. Using these components we propose and ablate an architecture that outperforms previous baselines on a number of popular datasets and behaves predictably when changing internal resolutions and latent sizes. We further propose to approximate a continuous feature field that adheres to the input points by coupling point-centered and grid-centered convolutions through feature projections. In contrast to recent works [HUANG et al. 2023, PARK et al. 2019, WILLIAMS et al. 2022], we maintain this “continuity” without any additional optimization or linear solves at inference-time.

7.2 Zero-Level-Set Encoder

In the following, we present our encoder-decoder architecture, including our *convolution block* for the encoder, the decoder structure, the loss function along with our modification to support non-manifold geometries, and the overall training process. We will also propose a new way to transfer information from points to voxels within neural networks, which is inspired by hybrid Eulerian-Lagrangian physical simulation methods.

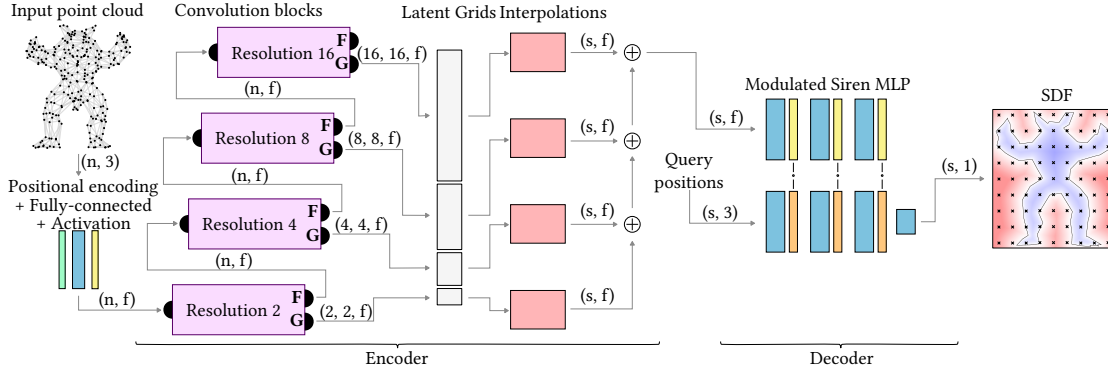


Figure 7.3. The encoder-decoder architecture of our network. The encoder computes vertex and volumetric features at multiple resolutions. By passing the feature vector through the convolution blocks, neighbor information is collected. The implementation of the convolution blocks is shown in Figure 7.2. After the last block, the point feature vector is discarded. The $+$ denotes element-wise vector addition. n : number of vertices. f : number of features. s : number of SDF sample points.

Our inputs are surface point clouds of 3D objects, given by a set of points $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^N$, $\mathbf{p}_i \in \mathbb{R}^3$. In order to use graph convolutions, we create edges $\mathcal{E} \subseteq \{1, \dots, N\} \times \{1, \dots, N\}$ between points, using e.g. k -nearest neighbors (k -NN) or radius graph connectivity. Within the network, the surface points store abstract f -dimensional feature vectors ($\mathbf{f}_i \in \mathbb{R}^f$), rather than 3D coordinates. This input representation also allows for utilization of known point connectivity, e.g., for partially meshed point cloud inputs or triangle soups.

7.2.1 Encoder-Decoder Architecture

Convolution block We introduce our hybrid point-grid convolution block in Figure 7.2 as the main building block for our encoder. This is in contrast to many previous encoder approaches that use either only point data [BOULCH and MARLET 2022, MESCHER et al. 2019, QI et al. 2017a, B. ZHANG et al. 2023] or transfer point data to voxel grids for further processing [CHIBANE et al. 2020a, PENG et al. 2020]. To get the best of both, we instead interleave these approaches. The intuition behind this idea is that the points represent exactly the shape that should be encoded. In order to reason about the surrounding space of the object, the features are projected onto a grid and propagated by using shallow convolutional layers. Then, to extract as much information as possible about the shape, features are interpolated from the grid to the input points and propagated between the nodes. The use of shallow 3D convolutions makes the network faster to evaluate and use less intermediate storage.

First, a graph convolution operator (e.g., EdgeConv by Y. WANG et al. [2019]) transforms each point feature $\mathbf{f}_i$ using the edge connectivity information. Next, we project the feature vectors onto a grid. We do this using our projection method described in Section 7.2.2. A $2 \times 2 \times 2$ convolution and subsequent deconvolution with activation (where the feature count is doubled for the latent space to retain information) is then used to exchange information between neighboring cells. We map the features back onto the points through tri-linear interpolation using the 8 closest cell centers. Here, they are combined with the original output of the graph convolution before finally being processed through a single per-point fully-connected layer. This output serves as the input of the next convolution block, while the deconvolved grid values are cached for later use.

We reason that the grid is suitable for distances in Euclidean space, because the regular grid structure implicitly assumes regular distances between cells and across convolution weights. This information can be used implicitly to more accurately locate patches of points, projected to individual grid cells, within the global structure. In turn, the graph convolution can maximize information extraction from the input points at a smaller scale. Connecting the points based on geometric proximity and using the

relative distance in graph convolutions makes the information extraction more “fine-grained” within local patches of points. Combining both multiple times throughout the network makes the feature extraction time and memory efficient.

Encoder The overall encoder-decoder architecture is shown in Figure 7.3. At the beginning of the encoder, the input point positions $\mathcal{P}$ are transformed to per-point features $\mathbf{f}_i$ through positional encoding [TANCIK et al. 2020] and a single linear layer with ReLU activations. We feed the encoded positions through a fixed number (typically 4-5) of our hybrid convolution blocks (as described above) with increasing resolution (contrary to the usual decreasing order of convolutional neural networks). The concatenated outputs of the grids form the grid vector (the point-level feature output of the last convolutional block can be discarded). Values from each level of the grid vector are extracted for each of the s SDF sample positions by tri-linear interpolation. The results are summed element-wise to form the final latent vector. We note that when latent vectors for additional sample positions are computed, there is no need to recompute the grid vector — it can be cached for future use.

Decoder The decoder receives the sample features and sample positions as input, see Section 7.2.4, and processes them individually to compute a single signed distance value for each sample. As architecture we make use of the proposed decoder of MEHTA et al. [2021], which is an augmentation of the SIREN layer proposed by SITZMANN et al. [2020b]. The feature vector is passed through a ReLU-activated MLP and the sample position through a sine-activated MLP. The activations of the ReLU MLP are then used to modulate the activations of the sine MLP whose output is the final SDF value. In our experiments, we have found sine-based decoders to consistently outperform ReLU-based decoders. In our testing, regular SIREN layers often had instabilities, which was remedied by using the modulation proposed by MEHTA et al. [2021].

The decoder is specifically chosen to be a smaller network, since per our motivation, it should be efficient to evaluate for a very large number of query points.

7.2.2 Point-to-Grid Transfer

Instead of the usual voxelization schemes applied in neural networks – typically occupancy indicators, max pooling or average pooling – we propose to use the particle-in-cell interpolation approach [HARLOW 1962, C. JIANG et al. 2015]. This approach is commonly used in hybrid physical simulation methods in order to transfer momentum from particles to an enclosing grid. The concept of this transfer is sketched in Figure 7.2. As opposed to commonly-used pooling operations, which only transfer features to the closest grid node, our projection operation splits up the feature vector across the eight closest grid nodes (for linear basis functions), shown by the black arrows. This intuitively allows for the reconstruction of a more accurate and consistent “feature-field” by using a more meaningful geometric operation. The feature vector $\mathbf{f}_i$ at each grid point i can be computed using

$$\hat{\mathbf{f}}_i = \sum_p w_{ip} m_p \mathbf{f}_p, \quad w_i = \sum_p w_{ip}, \quad \mathbf{f}_i = \frac{\hat{\mathbf{f}}_i}{w_i},$$

where $\mathbf{f}_p \in \mathbb{R}^f$ denotes the feature vector at point p . We further assume unit masses $m_p = 1$, meaning that each point carries the same amount of “weight” as all others. This quantity could theoretically be used to relatively weight contributions according to local point density or variability, or even to learn via, e.g., an attention-based mechanism. Lastly,

$$w_{ip} = N\left(\frac{x_p - x_i}{h}\right) N\left(\frac{y_p - y_i}{h}\right) N\left(\frac{z_p - z_i}{h}\right),$$

Table 7.1. Metadata about each of our datasets. $\mathcal{O}$ denotes a roughly even split of manifold and non-manifold instances.

(a) Number of shapes and vertices				
	Trained Shapes	Tested Shapes	Trained Vertices	Tested Vertices
Dragon	2,400	300	2,210	2,210
Armadillo	2,400	300	25,441	25,441
DFAUST	6,258	2,038	30,000	100,000
ScanNet	1,513	100	30,000	100,000
Thing10k	2,000	200	4 - 4,995	282 - 4,890
ShapeNet v2 planes	1,632	421	434 - 14,879	457 - 13,888
Objaverse	2,000	200	50,000	100,000

(b) Dataset properties			
	Manifold	Noisy	Sparse
Dragon	✓	✗	✗
Armadillo	✓	✗	✗
DFAUST	✗	✓	✗
ScanNet	✗	✓	✓
Thing10k	✓	✗	✓
ShapeNet v2 planes	✗	✗	✗
Objaverse	$\mathcal{O}$	✗	✗

denotes the interpolation weights, h the grid spacing and N the linear interpolation function given by

$$N(x) = \begin{cases} 1 - |x|, & 0 \leq |x| < 1 \\ 0, & 1 \leq |x|. \end{cases}$$

Using this interpolation scheme instead of a pooling approach allows for a smoother transition between graph-convolutions and grid-convolutions, since both of them now approximate the same continuous “feature-field”. We show in Section 7.3.3 that this improves reconstruction quality and reduces noise in reconstructed surfaces when compared to max pooling, especially when used in conjunction with input normals.

This projection scheme can also be interpreted as a kind of “cross-attention” layer, that computes grid features using a weighted sum of point features. Namely, the dot-product between key and query is replaced by the geometric weights w_{ip} computed from the interpolation function N . In contrast to regular scaled dot product attention, one can directly obtain grid features from point features, without “initializing” the grid features in any way.

7.2.3 Loss Function

An SDF can be defined as the unique solution Φ of the following eikonal equation:

$$\begin{aligned} \|\nabla\Phi(x)\| &= 1 \text{ for } x \in \Omega \setminus \Omega_S \subset \mathbb{R}^3, \\ \Phi(x) &= 0 \text{ for } x \in \Omega_S, \end{aligned} \tag{7.1}$$

where Ω is the SDF domain and Ω_S is the surface of the object. Related to the work of SMITH et al. [2021], SITZMANN et al. [2020b] proposed the following loss function as a measure of how well the

eikonal equation is satisfied

$$\begin{aligned}\mathcal{L}_{\text{eikonal}} = & \int_{\Omega} ||\nabla\Phi(\mathbf{x})|| - 1| \, d\mathbf{x} + \int_{\Omega_S} |\Phi(\mathbf{x})| \, d\mathbf{x} \\ & + \int_{\Omega_S} (1 - \langle \nabla\Phi(\mathbf{x}), \mathbf{n}_x \rangle) \, d\mathbf{x} \\ & + \int_{\Omega \setminus \Omega_S} \exp(-\alpha|\Phi(\mathbf{x})|) \, d\mathbf{x}.\end{aligned}\tag{7.2}$$

Here $\Phi(\mathbf{x})$ denotes the predicted signed distance value of our neural network at position $\mathbf{x} \in \Omega$, and $\mathbf{n}_x$ denotes the target surface normal. The exponential in the last term is a weighting function which “pushes” signed distance values away from zero for points not on the surface. The original work notes that the value of the constant α should be chosen as $\alpha \gg 1$, however, we have found that this is sometimes detrimental to the desired effect, while choosing a lower value of $\alpha \approx 10$ yielded the best results.

If the input surface is non-manifold or self-intersecting, the inside and outside of the object’s volume are not well defined and normal vectors do not necessarily point in the right direction. For these cases we introduce a simple modification that ignores the sign of the normal:

$$\mathcal{L}_{\text{surface normal}} = \int_{\Omega_S} (1 - |\langle \nabla\Phi(\mathbf{x}), \mathbf{n}_x \rangle|) \, d\mathbf{x}.\tag{7.3}$$

This change alone still allows for the result to be a valid *signed* distance field. As we discuss in Section 7.3.2, this greatly improves performance on meshes with poor quality. An example of multiple reconstructed SDF contours from a model trained using all of these losses on the ShapeNet v2 planes dataset is shown in Figure A.2 in the appendix for this chapter.

7.2.4 Training

A major advantage of our proposed architecture is that ground truth SDF values for the input point cloud *never* have to be computed. However, it does need sample points to train the network. While these could theoretically be generated on the fly, we sample the evaluation points in advance to reduce the training overhead. In order to evaluate the eikonal loss, only the sample positions and classification into (1) surface and (2) non-surface samples are required. After normalizing the input point clouds to the unit cube, the training samples can be trivially generated.

We sample them in the following ways: (1) Surface points. If the source is a triangle mesh rather than a point cloud, its surface is randomly sampled to create additional on-surface samples. If it is a point cloud, we either take all available points or subsample to a reasonable number (typically 100k-200k). (2a) Random points in the surrounding volume are sampled as off-surface samples (no check is performed if these points actually lie on the surface per chance since the surface has zero measure as a volume). (2b) If surface normal information is available, additional close-to-surface samples are generated from random points on the surface by displacing them along the surface normal. These additional samples are not strictly needed but aid the training process, as the SDF is most detailed close to the surface.

We train our encoder-decoder architecture end to end by inputting batches of surface point clouds and sample points and computing output signed distance values. The set of input points is always disjoint from the set of sample points. Automatic differentiation is used to provide spatial derivatives for the SDF gradients in $\mathcal{L}_{\text{eikonal}}$.

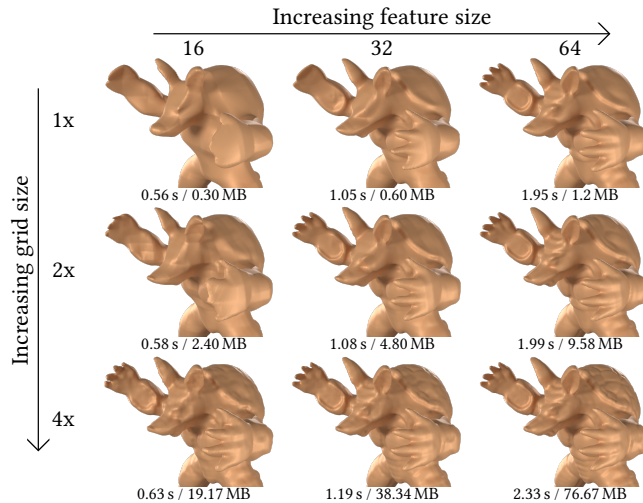


Figure 7.4. From top to bottom, we increase the size of all latent grids, while from left to right the latent feature size is increased. Below each figure, the inference time for 17M points (256^3 regular grid) and the storage requirements for the latent grid are shown. The number of trainable network parameters for feature sizes 16, 32, and 64 are 38K, 151K, and 604K, respectively, regardless of the specific grid sizes.

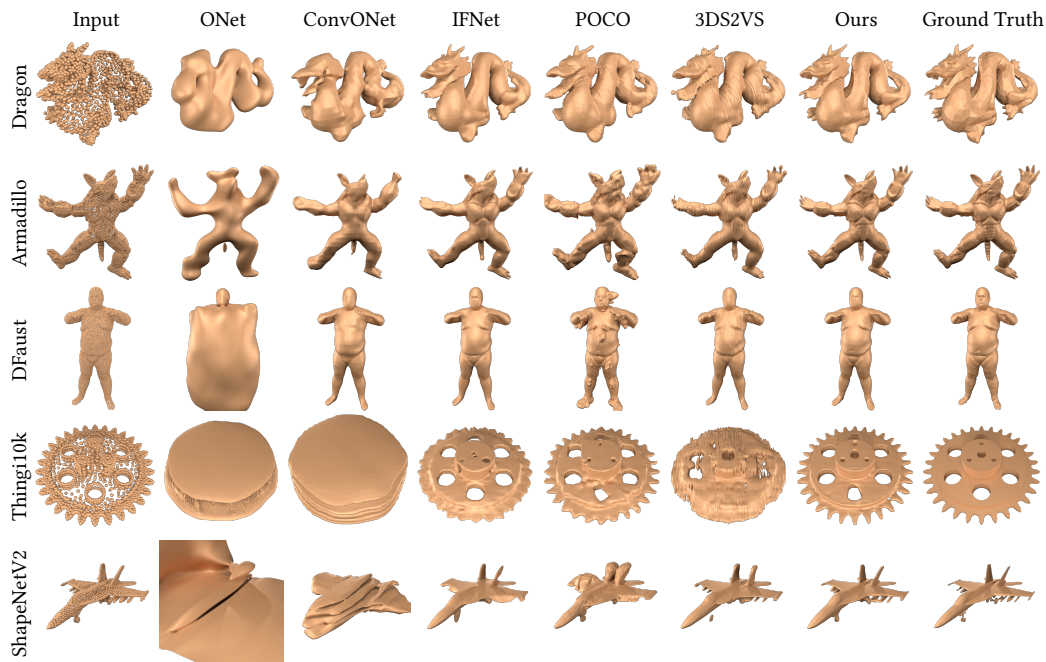


Figure 7.5. Comparing our method to related baselines. All methods use a variable number of input points for the Thingi10k and ShapeNet v2 datasets, while POCO and 3DS2VS use a fixed number of input points. Please refer to the accompanying video¹ for a more immersive comparison.

7.3 Results

The evaluation of our method is split into three parts: First, we compare different model sizes (i.e., number of learned parameters and grid resolutions) with respect to the amount of detail they can reconstruct. Second, we compare our results to a number of recent encoder architectures to highlight the advantages of our approach. Finally, we showcase the improvement of using our proposed point-to-grid feature projection over pooling-based voxelization. Details on general model ablation can be found in Section A.1 of the appendix for this chapter.

¹<https://youtu.be/ykKsjeX2-0M>

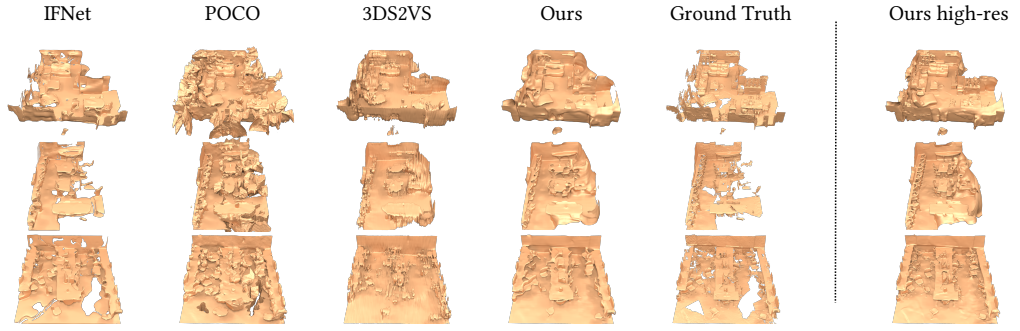


Figure 7.6. Comparing our method to related baselines on the indoor scene scan dataset ScanNet [DAI et al. 2017]. The results of ONet and ConvONet are omitted due to poor quality. In the rightmost column, we additionally show the results obtained with our method utilizing double resolution grid sizes and oriented point clouds. Best viewed under magnification in the digital document. Please refer to the accompanying video² for a more immersive comparison.

Datasets To cover a variety of relevant scenarios, we use seven different datasets, which are summarized in Table 7.1. The first two datasets consist of synthetic deformations of two different simulated solid objects, a low-resolution dragon and a high-resolution armadillo, which were computed using the IPC simulation framework [M. LI et al. 2020]. We additionally use the raw point scan data of the dynamic FAUST (DFAUST) dataset [BOGO et al. 2017], which consists of temporally deforming 3D shapes and the ScanNet dataset [DAI et al. 2017] which contains a large variety of scanned indoor scenes. Due to the origin of these datasets, they contain varying amounts of holes and noise. To analyze performance on variable point count as well as single-class and multi-class encoding, we employ the Thingi10k dataset as provided by Y. HU et al. [2018] as well as the “planes” category of the ShapeNet v2 dataset [CHANG et al. 2015]. Finally, we use the public-domain models of the Objaverse dataset [DEITKE et al. 2023] to test the ability of our method to fit more complex shapes.

Baselines We focus our comparison on other encoder-based 3D shape representation methods, as they are most similar to our work. To that end we implemented the encoder components of Occupancy Networks (ONet) [MESCHER et al. 2019], Convolutional Occupancy Networks (ConvONet) [PENG et al. 2020], Implicit Feature Networks (IFNet) [CHIBANE et al. 2020a], Point Convolution for Surface Reconstruction (POCO) [BOULCH and MARLET 2022] and 3DShape2VecSet (3DS2VS) [B. ZHANG et al. 2023]. To ensure a fair comparison, we use the same training and testing data for all models as well as the same modulated-SIREN decoder as discussed in Section 7.2. In addition, we use the same eikonal loss for all baselines, which trains all models to compute signed distance fields. We should note that while we use 3DS2VS for reconstruction, it is more strongly targeted towards latent compression and generation. Furthermore, the implementations of POCO and 3DS2VS were unfortunately not trivially adjustable to account for a variable number of input points during training, which is why they are evaluated using a fixed number of input points for our versions of the Thingi10K and ShapeNet v2 datasets. Specifically, we sample the mean number of points for each geometry across the respective datasets to train these models. For all comparisons with the other methods, our model utilizes five layers of latent grids with resolutions [4,8,16,32,64], a latent size of 64, $k = 8$ for k -NN connectivity and EdgeConv [Y. WANG et al. 2019] as point convolution operator. Please refer to Section A.1 in the appendix for this chapter for details on model component ablation.

Metrics We use four different metrics in our evaluation: The widely used Chamfer distance (CD) in L1 and L2 norm, normal consistency (NC), intersection-over-union (IoU) and F-score. The Chamfer

²<https://youtu.be/ykKsjeX2-0M>

Table 7.2. A summary of the comparisons between our encoder method, ONet [MESCHER et al. 2019], ConvONet [PENG et al. 2020], IFNet [CHIBANE et al. 2020a], POCO [BOULCH and MARLET 2022] and 3DS2VecSet [B. ZHANG et al. 2023]. The table shows the L1 Chamfer distance (CD-L1), L2 Chamfer distance (CD-L2), normal consistency (NC), intersection-over-union (IoU) and F-score (F). The arrow indicates whether lower or higher values are better. The best score is marked using a bold font. All values are given scaled to $\cdot 10^{-2}$. Footnotes for specific values: ¹ Evaluated on a subset of 200 random shapes due to computational constraints. ² Better metric but less desirable solution. ³ Trained and evaluated using a fixed number of input points instead of a variable number.

Metric	Method	Dragon	Armadillo	DFAUST	ScanNet	Thing10k	ShapeNet v2 w/ Eq. (7.3)	ShapeNet v2 w/o Eq. (7.3)
CD _{L1} ↓ mean	ONet	5.581	4.451	16.50	15.64	12.69	34.89	10.02
	ConvONet	3.960	2.377	1.627	10.69	10.62	3.734	4.424
	IFNet	2.681	1.259	1.015	1.559 ²	3.888	2.180	2.724
	POCO	2.981	2.594	3.480 ¹	29.82	3.235 ³	3.843 ³	4.382 ³
	3DS2VS	2.823	1.291	1.181	7.722	6.196 ³	2.362 ³	6.104 ³
	Ours	2.406	0.934	0.662	4.721	3.222	1.888	3.547
CD _{L2} ↓ mean	ONet	0.281	0.322	3.667	3.000	2.298	19.76	1.161
	ConvONet	0.143	0.096	0.070	1.421	1.407	0.164	0.235
	IFNet	0.056	0.048	0.101	0.022 ²	0.135	0.035	0.055
	POCO	0.067	0.067	0.217 ¹	14.53	0.120 ³	0.264 ³	0.243 ³
	3DS2VS	0.062	0.012	0.049	1.520	0.596 ³	0.050 ³	0.600 ³
	Ours	0.049	0.005	0.004	0.679	0.112	0.027	0.092
NC ↑ mean	ONet	82.20	83.35	82.70	64.51	62.16	59.61	58.11
	ConvONet	88.89	90.80	94.50	63.91	67.11	75.43	72.31
	IFNet	95.63	95.87	96.52	83.36 ²	92.27	82.34	81.61
	POCO	92.39	89.06	84.93 ¹	65.13	93.57 ³	78.01 ³	73.66 ³
	3DS2VS	92.64	94.12	95.33	74.22	79.74 ³	83.35 ³	70.11 ³
	Ours	96.33	98.22	97.41	82.27	94.26	86.75	78.67
IoU _{0.01} ↑ mean	ONet	32.41	32.60	20.90	9.428	11.65	17.70	11.97
	ConvONet	51.12	53.72	56.78	8.687	14.51	36.68	29.93
	IFNet	74.20	76.84	76.25	52.26	50.71	55.10	39.69
	POCO	63.80	37.25	39.52 ¹	31.83	78.42 ³	38.76 ³	30.57 ³
	3DS2VS	66.79	71.32	72.96	35.49	38.66 ³	52.54 ³	23.98 ³
	Ours	87.73	89.76	85.82	52.29	80.02	66.96	27.30
F _{0.01} ↑ mean	ONet	48.04	49.56	23.44	16.15	16.59	18.29	18.85
	ConvONet	68.44	74.68	82.04	13.54	21.30	56.87	45.84
	IFNet	94.29	96.86	96.50	73.10 ²	68.32	79.17	58.06
	POCO	86.53	56.65	68.94 ¹	20.15	91.73 ³	61.94 ³	43.52 ³
	3DS2VS	87.75	93.12	95.31	52.33	54.14 ³	78.80 ³	40.32 ³
	Ours	99.47	99.82	97.89	71.01	93.86	93.23	40.44

Table 7.3. Number of parameters and inference time of the different encoder baselines compared in Table 7.2.

	ONet	ConvONet	IFNet	POCO	3DS2VS	Ours
Params	626K	1.1M	1.9M	12.8M	106M	747K
Time	2.45 s	0.7 s	15.3 s	120 s to 330 s	12.3 s to 20.1 s	2.34 s

distance compares distances between two point clouds A and B by finding the distance to the closest point in B for each point in A and taking the average of the L1 and L2 norms, respectively. While the values typically correlate strongly, the L2 Chamfer distance penalizes outliers more strongly than the L1 Chamfer distance and is therefore also reported for completeness. The normal consistency instead compares orientations of normals by also finding the closest point in B for each point in A, but instead computing the cosine similarity of the normal vectors and taking the average. The same is done for the reverse direction and the results are summed together in the case of the Chamfer distance, and averaged in the case of the normal consistency.

The IoU and F-score are computed using a “thick-shell” surface representation of prediction and

ground truth surfaces. For the IoU, a voxel grid of 256^3 is sampled and cells marked as 1 that are within a distance of 0.01 for both the prediction and ground truth. The IoU is then the number of cells that are 1 in both grids divided by the number of cells that are 1 in either grid.

The F-score is computed in a similar manner, but uses sampled points instead of a regular grid. In specific, we sample 100k points on the surface of the predicted and ground-truth mesh, respectively. Recall is computed as the number of ground-truth points that are within 0.01 distance to the predicted surface, while precision is the reverse, i.e., the number of predicted points within 0.01 to the ground-truth surface. The F-score is finally computed as usual.

For all tests we use 100-2000 individual reconstructions and report mean values for all metrics.

Hardware and Framework We train all models on RTX 4000 Ada GPUs with 20 GB of memory (they also fit on an RTX 2080 Ti which increases the training times reported below by $\approx 30\%$). The training time for 400 epochs of our model using a batch size of 16 on a single GPU is between 1-2 days. Most other models trained in similar time, apart from IFNet, POCO, and 3DS2VS which required between 3-16 days on a single GPU. We implemented all of the models in this chapter in the PyTorch framework [PASZKE et al. 2019]. In addition, we make use of PyTorch-Geometric for its 3D graph and set learning capabilities [FEY and LENSSEN 2019]. All models were trained using the Adam optimizer using a learning rate of $5e-4$ and a batch size of 16. We have also implemented the tri-linear interpolation and point-to-grid projection using custom CUDA kernels. For the former, this includes the double backward pass, which is not supported by the native PyTorch interpolation function, but necessary because we are using network gradients in our training target. We limit the impact of the additional GPU memory and compute-load of this double-backward pass, by only differentiating the “small” decoder network twice, instead of the whole encoder-decoder structure.

7.3.1 Model Size

The number of features per point and the resolutions of the grids are tunable parameters of our architecture. Increasing them enables the network to capture more surface details, at the cost of increased storage requirements or computation time. The results are shown in Figure 7.4. We find that the network behaves very intuitively and that results degrade gracefully when the model size is reduced. Most notably, the models in the first column with a latent size of 16 all contain only 38K network parameters in total, of which the decoder contains just 1.4K. High-frequency features vanish first which makes small model sizes particularly appealing for applications such as approximate collision tests, e.g., in physical simulation, or deployment in resource constrained environments, e.g., edge devices.

7.3.2 Comparison to Related Work

We now discuss comparisons to the other baseline shape encoder methods on six datasets. They are shown for all methods and datasets in the accompanying video³, as well as in Figure 7.5, in Figure 7.6, in Table 7.2, and in Table 7.3. We can summarize that our method outperforms the other baselines with respect to reconstructed surface detail in terms of all metrics on nearly all datasets. There is however one exception, where IFNet seemingly performs better on ScanNet. On closer inspection, this is due to IFNet finding a less-desirable minimum to the eikonal loss, by “wrapping” thin sheets around the sparsely distributed geometry. While this results in smaller loss values, this is clearly a less desirable and less useful result. The visual comparison in Figure 7.6 and the accompanying video⁴ confirm this observation.

³<https://youtu.be/ykKsjeX2-0M>

⁴<https://youtu.be/ykKsjeX2-0M>

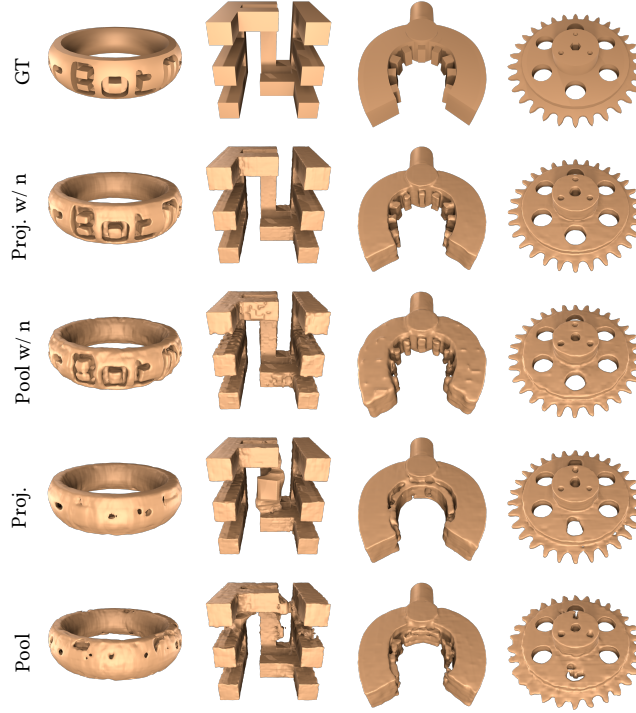


Figure 7.7. Visual comparison of our architecture using pooling-based and projection-based point-to-grid transfers. Rows two and three from the top use point normals as extra input.

A potential risk of using k-nearest-neighbor (k-NN) connectivity in our method is for fine detail to become “connected”. However, out of all the baselines, only our method was able to reconstruct the individual teeth of the cog wheel in the Thingi10k dataset. In addition, some methods also struggle with the hand of the armadillo and the claws of the dragon, while our method is able to reproduce them in better detail. We attribute this to the effectiveness of our proposed architecture and point-to-grid feature projection at extracting point-level information and detail. This assumption is supported by the results in Figure 7.7.

From Table 7.3 we can see, that our method also is able to reconstruct a dense grid of approx. 17M points in only 2.35 s, while IFNet takes more than 6x longer. Only ConvONet is faster on the reconstruction task, however at significantly lower quality. We can conclude that our approach offers the highest quality per parameter.

Our model consistently exhibits very low L2 Chamfer distance on all datasets. This indicates that there are fewer outliers, which means that the performance of our network is more predictable for a given level of accuracy. This is underlined by the comparison in Figure 7.4, which shows the effect of changing latent size and grid resolutions. Very small networks with our architecture can already have remarkable representational abilities.

Finally, we introduced a simple modification to the loss function, Equation (7.3), to be able to compute distance fields for surfaces with inconsistent surface normals, which we tested on the ShapeNet v2 dataset. Results without this modification are shown in the grey column in Table 7.2. We observed an improvement on all metrics for five of the six tested methods, with the exception of ONet [MESCHEDER et al. 2019], which partially performed better using the original loss. Although it should be noted that the model did not seem to be able to deal well with the changing number of vertices in the dataset, which is why we interpret this result as an outlier.

7.3.3 Point-to-Grid Feature Projection

As described in Section 7.2.2, we propose a physically-inspired method to project particle features to the convolutional grid. We find that the feature projection generally makes reconstructions smoother and details sharper, as can be seen in Figure 7.7. We further compare our method using projection with the well-established max pooling operator in the top two rows for each metric in Table A.5 of the appendix for this chapter. Simply exchanging the pooling operator for our projection already results in improved metrics for most datasets. This exchange comes at identical network complexity and memory requirements and nearly identical training and inference speed.

In the case that the input point clouds contain information about orientation, i.e., point normals, these can be used as additional inputs to our network. To do so, they are concatenated to the surface-point positions (see Figure 7.3). Results using normals are shown in the bottom two rows for each metric in Table A.5 for the pooling-based and projection-based variants of our network, respectively. Again, projection measurably improves performance, outperforming almost all other variants of our model across all datasets. The reconstructed surfaces in Figure 7.7 again show improved surface smoothness and detail sharpness. We also apply projection-based pooling using input normals to the ScanNet dataset, using twice the resolution of the baseline model, and show the results in the right column of Figure 7.6. It is again possible to see improved surface detail and smoothness. The reader is encouraged to view examples from the other datasets in the second half of the accompanying video⁵.

Since projection maps a single point to eight grid cells in contrast to the one-to-one mapping of pooling, we have at times found it useful to apply dropout within the projection operator. In practice we have found a larger dropout factor to be necessary, when the shapes within a dataset are more diverse. For the Objaverse dataset, we in fact only keep one of the eight points during training, whereas for the dragon and armadillo datasets we keep all eight.

7.4 Conclusion

We have shown that our hybrid approach of interleaving graph and grid convolutions, including point-to-grid projection and feature interpolation, is capable of outperforming other state-of-the-art encoders on the surface reconstruction task from point cloud inputs. We are able to do this using only surface information and additional unlabeled samples in the surrounding volume, which prevents us from having to solve for a ground truth signed distance field in advance. We have also introduced a physically-inspired projection operator for particle-to-grid feature transfers in neural networks and shown its effectiveness. Using this projection enables the reconstruction of both more detailed and less noisy surfaces, by improving “feature-field” consistency between point-based and grid-based representations. This chapter addresses the second main objective defined in Chapter 1, namely improving the efficiency of point-based representations in a data-driven way. The contribution complements the simulation-focused SPH chapters and is summarized jointly in Chapter 8.

We believe that this work could provide a foundation for further research on efficient models for 3D shape encoding. For instance, exploring the sparsity of our latent grids could enable the use of ever-higher latent resolutions, resulting in accurate encoding of almost arbitrarily complex shapes. Combined with small latent sizes, this could result in a more scalable architecture, where the accuracy can be determined by available compute capabilities. Another interesting direction of further study is the use of the latent grid as a basis for shape generation. Multiple latent representations could also be blended, or operators could be learned on the latent codes to achieve specific effects. Finally, in the spirit of using traditional geometric techniques within neural architectures, it could be explored

⁵<https://youtu.be/ykKsjeX2-0M>

whether projecting surface features to outer grid cells using methods such as fast marching or Laplacian smoothing could further improve predictions at distant points. The reader is referred to the project website hyve.physics-simulation.org for implementation details.

Part III

Summary

Concluding Remarks

This thesis used the simplest geometric primitive, points, for two purposes. These overarching goals were defined in Chapter 1, as *extending and improving the fidelity of point-based simulations* in complex melt-driven scenarios and *improving the efficiency of using point-based representations* for point-based surfaces.

8.1 Summary

The first of these goals was addressed in Part I on particle-based fluid simulation, specifically in Chapters 3 to 5. Answering the first two research questions, we can conclude that complex melt-driven processes can be captured with satisfying accuracy using the SPH method. We show that it achieves comparable accuracy to established methods for both the tungsten inert gas (TIG) welding process and the thermal spraying process. In addition, we are able to resolve the splat in thermal spraying with higher resolution than a comparable volume of fluid (VOF) approach, while also improving computational performance. This is done by incorporating a free enthalpy-based heat transfer model and implementing an implicit momentum sink for melting and solidification. All of these components result in a system that quantifiably improves the fidelity and accuracy of the SPH method for complex melt-driven scenarios. Focusing on strong surface tension effects, we found a way to formulate surface tension forces as an implicit linear system. We achieve this by selectively linearizing a cohesion-based surface tension model. This enables us to stably simulate a wide range of tension coefficients, outperforming previous work both in terms of stability and computational efficiency. As the resulting implicit system is a primal system solving for updated velocities, it naturally permits strong coupling between surface tension and viscosity. Strongly coupling these effects results in overall improved performance and accuracy if both forces are used.

The second goal was addressed in Chapter 7 of Part II, which considers deep learning methods as a way to improve the efficiency of point-based representations. Addressing the final research question, we construct a hybrid neural architecture using both points and enclosing grids to extract information from the input point cloud. This information is aggregated into a grid-based latent representation that can be efficiently queried at new positions to compute signed distance fields. In this way, we show that we are able to quickly compute SDFs, both for a single physically deforming shape and for the more general case of shape reconstruction from input points. Our proposed method outperforms existing architectures and approaches in terms of both reconstruction accuracy and computational efficiency, thus achieving our overall research goal.

All in all, points are powerful geometric primitives, both because their geometric simplicity makes them easy and efficient to use and because of the breadth of available models. In this thesis, we looked at specific methods to improve the fidelity and efficiency of these representations, using the Smoothed Particle Hydrodynamics (SPH) method for fluid simulation and a hybrid point-grid architecture for deep learning. Ultimately, we show that points can be used accurately in both continuum mechanics and geometry.

8.2 Outlook

The work on particles in this thesis underscored the remarkable effectiveness of this representation. Nevertheless, there remain a number of questions and open problems that result from the presented works. First, considering SPH fluid simulations, we identify several avenues for improvement with increasing perceived level of difficulty. To start, the thermal model for melt-driven processes, while robust and well-founded, did not take into account factors such as latent heat, whose inclusion would significantly improve the accuracy of the captured process. Furthermore, while temperature-dependent parameters were modeled as non-linear through tabulation, the temperature solve was explicitly integrated and decoupled from the fluid solve. Combining these solvers in a monolithic system could further increase the fidelity of the simulation, but would result in a massively complex system depending on implementation.

The surface tension force in our work of Chapter 3 was based on cohesion forces. This resulted in visually correct results and is a great tool for high-surface tension scenes. However, we believe that a curvature-based approach using the continuum surface force (CSF) model might yield more consistent results if implemented correctly. Currently, CSF models are often unstable and very prone to errors, resulting from highly fluctuating second derivatives, which are used to compute the curvature of the surface. ZORILLA et al. [2020] show that a robust geometric estimate of curvature values can remedy many of the problems with the method and capture intricate surface tension effects, but still falls prey to instabilities caused by explicit integration for high stiffnesses. We believe that attempting to formulate an implicit CSF model with a geometric curvature estimate could be a worthwhile direction of further research.

This leads to the final, and possibly most difficult, problem that this thesis identifies as possible follow-up work for SPH simulation. One of the most fundamental limitations in formulating the implicit surface tension force for this work was the constraint that the implicit system had to be linear. This is a purely practical limitation born from computational concerns, since there is no actual mathematical restriction. Let us assume we use Newton’s method to solve for primal variables, velocity and position. Then we need to take the first and second derivatives of the function we aim to minimize, invariably a function that is based on an SPH sum as in Equation (2.14). Taking the derivative with respect to positions automatically involves iterating not just over a particle’s immediate neighborhood but also over the neighbors of all particle neighbors, because positions $\mathbf{x}_i$ are also contained in the kernel for computations of all neighbor densities ρ_j . This alone results in a significant increase in computational requirements. In addition, running a Newton iteration would require computing a new particle state every iteration, including neighborhood search. This twofold problem alone makes investigating non-linear solvers for SPH a challenging proposition. Possible remedies could include adapting neighborhood search algorithms to include an uncertainty radius and computing slightly larger neighborhoods, as well as simplified non-linear systems that keep positions fixed while solving only for velocities. In addition, quasi-Newton non-linear solvers such as the limited-memory BFGS solver could be explored for more efficient solutions.

In the deep learning work we showed that we are able to use a latent representation to effectively

encode geometry. The logical next step is to explore the structure of this latent representation for a variety of downstream tasks, such as generating new shapes or learning latent editing operators to modify and combine existing shapes directly in latent space. For shape generation, we could further explore attention mechanisms to improve information extraction from the input point cloud, such as Point Transformers [X. Wu et al. 2024], or cross-attention between grid and point representations. Finally, the efficiency aspect of the model could be further explored by using the inherent sparsity of latent grids, which could enable scaling to even larger model sizes and ease deployment on compute-constrained devices.

Bibliography

- AANJANEYA, MRIDUL, SAKET PATKAR, and RONALD FEDKIW [2013]. “A Monolithic Mass Tracking Formulation for Bubbles in Incompressible Flow”. In: *Journal of Computational Physics*. DOI: [10.1016/j.jcp.2013.03.048](https://doi.org/10.1016/j.jcp.2013.03.048).
- ABUBAKAR, ABBA A. and ABUL FAZAL M. ARIF [2019]. “A Hybrid Computational Approach for Modeling Thermal Spray Deposition”. In: *Surface and Coatings Technology*. DOI: [10.1016/j.surfcoat.2019.02.010](https://doi.org/10.1016/j.surfcoat.2019.02.010).
- ADCOCK, STEWART A. and J. ANDREW McCAMMON [2006]. “Molecular Dynamics: Survey of Methods for Simulating the Activity of Proteins”. In: *Chemical Reviews*. DOI: [10.1021/cr040426m](https://doi.org/10.1021/cr040426m).
- AKINCI, NADIR, GIZEM AKINCI, and MATTHIAS TESCHNER [2013]. “Versatile Surface Tension and Adhesion for SPH Fluids”. In: *ACM Trans. Graph.* DOI: [10.1145/2508363.2508395](https://doi.org/10.1145/2508363.2508395).
- AKINCI, NADIR, MARKUS IHMSEN, GIZEM AKINCI, BARBARA SOLENTHALER, and MATTHIAS TESCHNER [2012]. “Versatile Rigid-Fluid Coupling for Incompressible SPH”. In: *ACM Transactions on Graphics*. DOI: [10.1145/2185520.2335413](https://doi.org/10.1145/2185520.2335413).
- ALBADAWI, A., D.B. DONOGHUE, A.J. ROBINSON, D.B. MURRAY, and Y.M.C. DELAURÉ [2013]. “Influence of Surface Tension Implementation in Volume of Fluid and Coupled Volume of Fluid with Level Set Methods for Bubble Growth and Detachment”. In: *International Journal of Multiphase Flow*. DOI: [10.1016/j.ijmultiphaseflow.2013.01.005](https://doi.org/10.1016/j.ijmultiphaseflow.2013.01.005).
- ATZMON, MATAN and YARON LIPMAN [2020a]. “SAL: Sign Agnostic Learning of Shapes From Raw Data”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.
- [2020b]. “SALD: Sign Agnostic Learning with Derivatives”. In: 9th International Conference on Learning Representations, ICLR 2021.
- AZENCOT, OMRI, ORESTIS VANTZOS, MAX WARDETZKY, MARTIN RUMPF, and MIRELA BEN-CHEN [2015]. “Functional Thin Films on Surfaces”. In: *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. Los Angeles, California and New York, NY, USA. DOI: [10.1145/2786784.2786793](https://doi.org/10.1145/2786784.2786793).
- BARECASCO, AGRA, HANIFA TERRISSA, and CHRISTIAN FREDY NAA [2013]. “Simple Free-Surface Detection in Two and Three-Dimensional Sph Solver”. arXiv: [1309.4290v1](https://arxiv.org/abs/1309.4290v1) [[physics.flu-dyn](https://arxiv.org/abs/1309.4290v1)].
- BARRON, JONATHAN T., BEN MILDENHALL, DOR VERBIN, PRATUL P. SRINIVASAN, and PETER HEDMAN [2023]. “Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields”. In: Proceedings of the IEEE/CVF International Conference on Computer Vision.
- BATCHELOR, G. K. [2010]. *An Introduction to Fluid Dynamics*. 14. print. Cambridge. 615 pp.
- BATTY, CHRISTOPHER, ANDRES URIBE, BASILE AUDOLY, and EITAN GRINSFUD [2012]. “Discrete Viscous Sheets”. In: *ACM Transactions on Graphics*. DOI: [10.1145/2185520.2185609](https://doi.org/10.1145/2185520.2185609).
- BECKER, MARKUS and MATTHIAS TESCHNER [2007]. “Weakly Compressible SPH for Free Surface Flows”. In: *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Goslar, DEU.
- BENDER, JAN et al. [2026]. *SPlisHSPlasH*. URL: <https://github.com/InteractiveComputerGraphics/SPlisHSPlasH>.

- BENDER, JAN and DAN KOSCHIER [2015]. “Divergence-Free Smoothed Particle Hydrodynamics”. In: *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.
- [2017]. “Divergence-Free SPH for Incompressible and Viscous Fluids”. In: *IEEE Transactions on Visualization and Computer Graphics*. DOI: [10.1109/TVCG.2016.2578335](https://doi.org/10.1109/TVCG.2016.2578335).
- BENDER, JAN, TASSILO KUGELSTADT, MARCEL WEILER, and DAN KOSCHIER [2019]. “Volume Maps: An Implicit Boundary Representation for SPH”. In: *Motion, Interaction and Games*.
- BENDER, JAN, LUKAS WESTHOFEN, and STEFAN RHYS JESKE [2023]. “Consistent SPH Rigid-Fluid Coupling”. In: *Vision, Modeling, and Visualization*, pp. 209–217. DOI: [10.2312/VMV.20231244](https://doi.org/10.2312/VMV.20231244).
- BERGOU, MIKLÓS, BASILE AUDOLY, ETIENNE VOUGA, MAX WARDETZKY, and EITAN GRINSUN [2010]. “Discrete Viscous Threads”. In: *ACM Transactions on Graphics*. DOI: [10.1145/1833351.1778853](https://doi.org/10.1145/1833351.1778853).
- BOBZIN, K, M ÖTE, M A KNOCH, I ALKHASLI, and S R DOKHANCHI [2019]. “Modelling of Particle Impact Using Modified Momentum Source Method in Thermal Spraying”. In: *IOP Conference Series: Materials Science and Engineering*. DOI: [10.1088/1757-899x/480/1/012003](https://doi.org/10.1088/1757-899x/480/1/012003).
- BOBZIN, K., W. WIETHEGER, H. HEINEMANN, and F. WOLF [2021a]. “Simulation of Thermally Sprayed Coating Properties Considering the Splat Boundaries”. In: *IOP Conference Series: Materials Science and Engineering*. DOI: [10.1088/1757-899x/1147/1/012026](https://doi.org/10.1088/1757-899x/1147/1/012026).
- BOBZIN, KIRSTEN, WOLFGANG WIETHEGER, HENDRIK HEINEMANN, and ILKIN ALKHASLI [2021b]. “Simulation of Multiple Particle Impacts in Plasma Spraying”. In: *Enhanced Material, Parts Optimization and Process Intensification*. Cham.
- BOGO, FEDERICA, JAVIER ROMERO, GERARD PONS-MOLL, and MICHAEL J. BLACK [2017]. “Dynamic FAUST: Registering Human Bodies in Motion”. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). DOI: [10.1109/CVPR.2017.591](https://doi.org/10.1109/CVPR.2017.591).
- BORRELL, R., ORIOL LEHMKUHL, and JESÚS CASTRO [2013]. “Parallelization Strategy for the Volume-of-Fluid Method on Unstructured Meshes”. In: *Procedia Engineering*. DOI: [10.1016/j.proeng.2013.08.003](https://doi.org/10.1016/j.proeng.2013.08.003).
- BOULCH, ALEXANDRE and RENAUD MARLET [2022]. “POCO: Point Convolution for Surface Reconstruction”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.
- BOYD, LANDON and ROBERT BRIDSON [2012]. “MultiFLIP for Energetic Two-Phase Fluid Simulation”. In: *ACM Trans. Graph.* DOI: [10.1145/2159516.2159522](https://doi.org/10.1145/2159516.2159522).
- BRACKBILL, J. U, D. B KOTHE, and C ZEMACH [1992]. “A Continuum Method for Modeling Surface Tension”. In: *Journal of Computational Physics*. DOI: [10.1016/0021-9991\(92\)90240-Y](https://doi.org/10.1016/0021-9991(92)90240-Y).
- BRENT, A. D., V. R. VOLLER, and K. J. REID [1988]. “Enthalpy-Porosity Technique for Modeling Convection-Diffusion Phase Change: Application to the Melting of a Pure Metal”. In: *Numerical Heat Transfer*. DOI: [10.1080/10407788808913615](https://doi.org/10.1080/10407788808913615).
- BRONSTEIN, MICHAEL M., JOAN BRUNA, TACO COHEN, and PETAR VELIČKOVIĆ [2021]. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. DOI: [10.48550/arXiv.2104.13478](https://doi.org/10.48550/arXiv.2104.13478). arXiv: [2104.13478](https://arxiv.org/abs/2104.13478) [cs]. Pre-published.
- BROOKSHAW, L. [1985]. “A Method of Calculating Radiative Heat Diffusion in Particle Simulations”. In: *Publications of the Astronomical Society of Australia*. DOI: [10.1017/S1323358000018117](https://doi.org/10.1017/S1323358000018117).
- BUTEREZ, DAVID, JON PAUL JANET, STEVEN J. KIDDLE, DINO OGLIC, and PIETRO LIÒ [2022]. “Graph Neural Networks with Adaptive Readouts”. In: *Advances in Neural Information Processing Systems*.
- CADIOU, S., M. COURTOIS, M. CARIN, W. BERCKMANS, and P. LE MASSON [2020]. “3d Heat Transfer, Fluid Flow and Electromagnetic Model for Cold Metal Transfer Wire Arc Additive Manufacturing (Cmt-Waam)”. In: *Additive Manufacturing*. DOI: [10.1016/j.addma.2020.101541](https://doi.org/10.1016/j.addma.2020.101541).
- CAHN, JOHN W. and JOHN E. HILLIARD [1958]. “Free Energy of a Nonuniform System. I. Interfacial Free Energy”. In: *The Journal of Chemical Physics*. DOI: [10.1063/1.1744102](https://doi.org/10.1063/1.1744102).

- CHABRA, ROHAN, JAN E. LENSSEN, EDDY ILG, TANNER SCHMIDT, JULIAN STRAUB, STEVEN LOVEGROVE, and RICHARD NEWCOMBE [2020]. “Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction”. In: *Computer Vision – ECCV 2020*. Cham. doi: [10.1007/978-3-030-58526-6_36](https://doi.org/10.1007/978-3-030-58526-6_36).
- CHANDRA, SANJEEV and PIERRE FAUCHAIS [2009]. “Formation of Solid Splats during Thermal Spray Deposition”. In: *Journal of Thermal Spray Technology*. doi: [10.1007/s11666-009-9294-5](https://doi.org/10.1007/s11666-009-9294-5).
- CHANG, ANGEL X., THOMAS FUNKHOUSER, LEONIDAS GUIBAS, PAT HANRAHAN, QIXING HUANG, ZIMO LI, SILVIO SAVARESE, MANOLIS SAVVA, SHURAN SONG, HAO SU, JIANXIONG XIAO, LI YI, and FISHER YU [2015]. *ShapeNet: An Information-Rich 3D Model Repository*. Stanford University – Princeton University – Toyota Technological Institute at Chicago.
- CHEN, CHAO, YU-SHEN LIU, and ZHIZHONG HAN [2025]. “NeuralTPS: Learning Signed Distance Functions Without Priors From Single Sparse Point Clouds”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. doi: [10.1109/TPAMI.2024.3476349](https://doi.org/10.1109/TPAMI.2024.3476349).
- CHEN, JINGYU, VICTORIA KALA, ALAN MARQUEZ-RAZON, ELIAS GUEIDON, DAVID A. B. HYDE, and JOSEPH TERAN [2021]. “A Momentum-Conserving Implicit Material Point Method for Surface Tension with Contact Angles and Spatial Gradients”. In: *ACM Transactions on Graphics*. doi: [10.1145/3450626.3459874](https://doi.org/10.1145/3450626.3459874).
- CHEN, YI-LU, JONATHAN MEIER, BARBARA SOLENTHALER, and VINICIUS C. AZEVEDO [2020]. “An Extended Cut-Cell Method for Sub-Grid Liquids Tracking with Surface Tension”. In: *ACM Transactions on Graphics*. doi: [10.1145/3414685.3417859](https://doi.org/10.1145/3414685.3417859).
- CHEN, PETER YICHEN, JINXU XIANG, DONG HEON CHO, YUE CHANG, G A PERSHING, HENRIQUE TELES MAIA, MAURIZIO M CHIARAMONTE, KEVIN THOMAS CARLBERG, and EITAN GRINSPUN [2023]. “CROM: Continuous Reduced-Order Modeling of PDEs Using Implicit Neural Representations”. In: *The Eleventh International Conference on Learning Representations*.
- CHENG, YEN-CHI, HSIN-YING LEE, SERGEY TUYAKOV, ALEX SCHWING, and LIANGYAN GUI [2023]. “SDFu-sion: Multimodal 3D Shape Completion, Reconstruction, and Generation”. In: *CVPR*.
- CHIBANE, JULIAN, THIEMO ALLDIECK, and GERARD PONS-MOLL [2020a]. “Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- CHIBANE, JULIAN, MOHAMAD AYMEN MIR, and GERARD PONS-MOLL [2020b]. “Neural Unsigned Distance Fields for Implicit Function Learning”. In: *Advances in Neural Information Processing Systems*.
- CHO, WON-IK and SUK-JOO NA [2021]. “Impact of Driving Forces on Molten Pool in Gas Metal Arc Welding”. In: *Welding in the World*. doi: [10.1007/s40194-021-01138-8](https://doi.org/10.1007/s40194-021-01138-8).
- CHORIN, A. J. and J. E. MARSDEN [1990]. *A Mathematical Introduction to Fluid Mechanics*. New York, NY. doi: [10.1007/978-1-4684-0364-0](https://doi.org/10.1007/978-1-4684-0364-0).
- CHORIN, ALEXANDRE JOEL [1969]. “On the Convergence of Discrete Approximations to the Navier-Stokes Equations”. In: *Mathematics of Computation*. doi: [10.1090/S0025-5718-1969-0242393-5](https://doi.org/10.1090/S0025-5718-1969-0242393-5).
- CLAVET, SIMON, PHILIPPE BEAUDOIN, and PIERRE POULIN [2005]. “Particle-Based Viscoelastic Fluid Simulation”. In: *Symposium on Computer Animation*. doi: [10.2312/SCA/SCA05/219-228](https://doi.org/10.2312/SCA/SCA05/219-228).
- DA, FANG, CHRISTOPHER BATTY, CHRIS WOJTAN, and EITAN GRINSPUN [2015]. “Double Bubbles sans Toil and Trouble: Discrete Circulation-Preserving Vortex Sheets for Soap Films and Foams”. In: *ACM Transactions on Graphics*. doi: [10.1145/2767003](https://doi.org/10.1145/2767003).
- DAI, ANGELA, ANGEL X. CHANG, MANOLIS SAVVA, MACIEJ HALBER, THOMAS FUNKHOUSER, and MATTHIAS NIESSNER [2017]. “ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- DAS, R. and P.W. CLEARY [2016]. “Three-Dimensional Modelling of Coupled Flow Dynamics, Heat Transfer and Residual Stress Generation in Arc Welding Processes Using the Mesh-Free SPH Method”. In: *Journal of Computational Science*. doi: [10.1016/j.jocs.2016.03.006](https://doi.org/10.1016/j.jocs.2016.03.006).

- DAVIDSON, P.A. [2001]. *An Introduction to Magnetohydrodynamics*.
- DAVIES, J.R. [2004]. *Handbook of Thermal Spray Technology*. Materials Park, OH.
- DEEPSEEK-AI et al. [2025]. *DeepSeek-V3 Technical Report*. Version 2. DOI: [10.48550/arXiv.2412.19437](https://doi.org/10.48550/arXiv.2412.19437). arXiv: [2412.19437](https://arxiv.org/abs/2412.19437) [cs]. Pre-published.
- DEITKE, MATT, DUSTIN SCHWENK, JORDI SALVADOR, LUCA WEIHS, OSCAR MICHEL, ELI VANDERBILT, LUDWIG SCHMIDT, KIANA EHSANI, ANIRUDDHA KEMBHAVI, and ALI FARHADI [2023]. “Objaverse: A Universe of Annotated 3D Objects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- DESBRUN, MATHIEU and MARIE-PAULE GASCUEL [1996]. “Smoothed Particles: A New Paradigm for Animating Highly Deformable Bodies”. In: *Eurographics*. DOI: [10.1007/978-3-7091-7486-9_5](https://doi.org/10.1007/978-3-7091-7486-9_5).
- FANG, H. S., K. BAO, J. A. WEI, H. ZHANG, E. H. WU, and L. L. ZHENG [2009]. “Simulations of Droplet Spreading and Solidification Using an Improved SPH Model”. In: *Numerical Heat Transfer, Part A: Applications*. DOI: [10.1080/10407780802603139](https://doi.org/10.1080/10407780802603139).
- FARROKHMANAH, AMIRSAMAN, MARKUS BUSSMANN, and JAVAD MOSTAGHIMI [2017]. “New Smoothed Particle Hydrodynamics (SPH) Formulation for Modeling Heat Conduction with Solidification and Melting”. In: *Numerical Heat Transfer, Part B: Fundamentals*. DOI: [10.1080/10407790.2017.1293972](https://doi.org/10.1080/10407790.2017.1293972).
- FARROKHMANAH, AMIRSAMAN, JAVAD MOSTAGHIMI, and MARKUS BUSSMANN [2021]. “Nonlinear Enthalpy Transformation for Transient Convective Phase Change in Smoothed Particle Hydrodynamics (SPH)”. In: *Numerical Heat Transfer, Part B: Fundamentals*. DOI: [10.1080/10407790.2021.1929295](https://doi.org/10.1080/10407790.2021.1929295).
- FEY, MATTHIAS and JAN E. LENSSEN [2019]. “Fast Graph Representation Learning with PyTorch Geometric”. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
- FRIDOVICH-KEIL, SARA, GIACOMO MEANTI, FREDERIK RAHBÆK WARBURG, BENJAMIN RECHT, and ANGJOO KANAZAWA [2023]. “K-Planes: Explicit Radiance Fields in Space, Time, and Appearance”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- GEIGER, A, P LENZ, C STILLER, and R URTASUN [2013]. “Vision Meets Robotics: The KITTI Dataset”. In: *Int. J. Rob. Res.* DOI: [10.1177/0278364913491297](https://doi.org/10.1177/0278364913491297).
- GHAFOURI-AZAR, R., J. MOSTAGHIMI, S. CHANDRA, and M. CHARMCHI [2003]. “A Stochastic Model to Simulate the Formation of a Thermal Spray Coating”. In: *Journal of Thermal Spray Technology*. DOI: [10.1361/105996303770348500](https://doi.org/10.1361/105996303770348500).
- GINGOLD, R. a. and J.J. MONAGHAN [1977]. “Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars”. In: *Monthly Notices of the Royal Astronomical Society*. DOI: [10.1093/mnras/181.3.375](https://doi.org/10.1093/mnras/181.3.375).
- GISSLER, CHRISTOPH, ANDREAS HENNE, STEFAN BAND, ANDREAS PEER, and MATTHIAS TESCHNER [2020]. “An Implicit Compressible SPH Solver for Snow Simulation”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3386569.3392431](https://doi.org/10.1145/3386569.3392431).
- GISSLER, CHRISTOPH, ANDREAS PEER, STEFAN BAND, JAN BENDER, and MATTHIAS TESCHNER [2019]. “Interlinked SPH Pressure Solvers for Strong Fluid-Rigid Coupling”. In: *ACM Trans. Graph.* DOI: [10.1145/3284980](https://doi.org/10.1145/3284980).
- GOODFELLOW, IAN, YOSHUA BENGIO, and AARON COURVILLE [2016]. *Deep Learning*. Cambridge, Massachusetts. 775 pp.
- GOUTIER, SIMON and P. FAUCHAIS [2011]. “Last Developments in Diagnostics to Follow Splats Formation during Plasma Spraying”. In: *Journal of Physics: Conference Series*. DOI: [10.1088/1742-6596/275/1/012003](https://doi.org/10.1088/1742-6596/275/1/012003).
- GROPP, AMOS, LIOR YARIV, NIV HAIM, MATAN ATZMON, and YARON LIPMAN [2020]. “Implicit Geometric Regularization for Learning Shapes”. In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning.

- GUPTA, ANCHIT, WENHAN XIONG, YIXIN NIE, IAN JONES, and BARLAS OĞUZ [2023]. *3DGen: Triplane Latent Diffusion for Textured Mesh Generation*. DOI: [10.48550/arXiv.2303.05371](https://doi.org/10.48550/arXiv.2303.05371). arXiv: [2303.05371](https://arxiv.org/abs/2303.05371) [cs]. Pre-published.
- HAIDAR, J and JJ LOWKE [1996]. “Predictions of Metal Droplet Formation in Arc Welding”. In: *Journal of Physics D: Applied Physics*.
- HARLOW, FRANCIS [1962]. *The Particle-in-Cell Method for Numerical Solution of Problems in Fluid Dynamics*. DOI: [10.2172/4769185](https://doi.org/10.2172/4769185).
- HE, XIAOWEI, HUAMIN WANG, FENGJUN ZHANG, HONGAN WANG, GUOPING WANG, and KUN ZHOU [2015]. “Robust Simulation of Sparsely Sampled Thin Features in SPH-Based Free Surface Flows”. In: *ACM Trans. Graph.* DOI: [10.1145/2682630](https://doi.org/10.1145/2682630).
- HERTEL, M, U FÜSSEL, and M SCHNICK [2014]. “Numerical Simulation of the Plasma – Mig Process – Interactions of the Arcs, Droplet Detachment and Weld Pool Formation”. In: *Welding in the World*.
- HERTZ, AMIR, OR PEREL, RAJA GIRYES, OLGA SORKINE-HORNUNG, and DANIEL COHEN-OR [2021]. “SAPE: Spatially-Adaptive Progressive Encoding for Neural Optimization”. In: *Advances in Neural Information Processing Systems*.
- HOLZ, DANIEL, STEFAN RHYS JESKE, FABIAN LÖSCHNER, JAN BENDER, YIN YANG, and SHELDON ANDREWS [2025]. “Multiphysics Simulation Methods in Computer Graphics”. In: *Computer Graphics Forum* 44.2, e70082. DOI: [10.1111/cgf.70082](https://doi.org/10.1111/cgf.70082).
- HONG, JEONG-MO and CHANG-HUN KIM [2003]. “Animation of Bubbles in Liquid”. In: *Computer Graphics Forum*. DOI: [10.1111/1467-8659.00672](https://doi.org/10.1111/1467-8659.00672).
- [2005]. “Discontinuous Fluids”. In: *ACM Trans. Graph.* DOI: [10.1145/1073204.1073283](https://doi.org/10.1145/1073204.1073283).
- HONG, JEONG-MO, HO-YOUNG LEE, JONG-CHUL YOON, and CHANG-HUN KIM [2008]. “Bubbles Alive”. In: *ACM Transactions on Graphics*. DOI: [10.1145/1360612.1360647](https://doi.org/10.1145/1360612.1360647).
- HOPP-HIRSCHLER, MANUEL and ULRICH NIEKEN [2019]. “Fully Implicit Time Integration in Truly Incompressible SPH”. In: *The European Physical Journal Special Topics*. DOI: [10.1140/epjst/e2019-800152-6](https://doi.org/10.1140/epjst/e2019-800152-6).
- HU, H. and S. ARGYROPOULOS [1996]. “Mathematical Modelling of Solidification and Melting: A Review”. In: *Modelling and Simulation in Materials Science and Engineering*.
- HU, X.Y. and N.A. ADAMS [2006]. “A Multi-Phase SPH Method for Macroscopic and Mesoscopic Flows”. In: *Journal of Computational Physics*. DOI: [10.1016/j.jcp.2005.09.001](https://doi.org/10.1016/j.jcp.2005.09.001).
- HU, YIXIN, QINGNAN ZHOU, XIFENG GAO, ALEC JACOBSON, DENIS ZORIN, and DANIELE PANOZZO [2018]. “Tetrahedral Meshing in the Wild”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3197517.3201353](https://doi.org/10.1145/3197517.3201353).
- HUANG, JIAHUI, HAO-XIANG CHEN, and SHI-MIN HU [2022]. “A Neural Galerkin Solver for Accurate Surface Reconstruction”. In: *ACM Trans. Graph.* DOI: [10.1145/3550454.3555457](https://doi.org/10.1145/3550454.3555457).
- HUANG, JIAHUI, ZAN GOJCIC, MATAN ATZMON, OR LITANY, SANJA FIDLER, and FRANCIS WILLIAMS [2023]. “Neural Kernel Surface Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- HUBER, MARKUS, STEFAN REINHARDT, DANIEL WEISKOPF, and BERNHARD EBERHARDT [2015]. *Evaluation of Surface Tension Models for SPH-based Fluid Animations Using a Benchmark Test*. The Eurographics Association. DOI: [10.2312/VRIPHYS.20151333](https://doi.org/10.2312/VRIPHYS.20151333).
- HYDE, DAVID A. B., STEVEN W. GAGNIERE, ALAN MARQUEZ-RAZON, and JOSEPH TERAN [2020]. “An Implicit Updated Lagrangian Formulation for Liquids with Large Surface Energy”. In: *ACM Trans. Graph.* DOI: [10.1145/3414685.3417845](https://doi.org/10.1145/3414685.3417845).
- IHMSEN, MARKUS, JENS CORNELIS, BARBARA SOLENTHALER, CHRISTOPHER HORVATH, and MATTHIAS TESCHNER [2014a]. “Implicit Incompressible SPH”. In: *IEEE Transactions on Visualization and Computer Graphics*. DOI: [10.1109/TVCG.2013.105](https://doi.org/10.1109/TVCG.2013.105).

- IHMSEN, MARKUS, JENS ORTHMANN, BARBARA SOLENTHALER, ANDREAS KOLB, and MATTHIAS TESCHNER [2014b]. “SPH Fluids in Computer Graphics”. In: *Eurographics (State of the Art Reports)*. DOI: [10.2312/egst.20141034](https://doi.org/10.2312/egst.20141034).
- ISHIDA, SADASHIGE, PETER SYNAK, FUMIYA NARITA, TOSHIYA HACHISUKA, and CHRIS WOJTAN [2020]. “A Model for Soap Film Dynamics with Evolving Thickness”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3386569.3392405](https://doi.org/10.1145/3386569.3392405).
- ISHIDA, SADASHIGE, MASAFUMI YAMAMOTO, RYOICHI ANDO, and TOSHIYA HACHISUKA [2017]. “A Hyperbolic Geometric Flow for Evolving Films and Foams”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3130800.3130835](https://doi.org/10.1145/3130800.3130835).
- ISSA, R. I [1986]. “Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting”. In: *Journal of Computational Physics*. DOI: [10.1016/0021-9991\(86\)90099-9](https://doi.org/10.1016/0021-9991(86)90099-9).
- ITO, M., S. IZAWA, Y. FUKUNISHI, and M. SHIGETA [2012]. “Sph Simulation of Gas Arc Welding Process”. In: *Proc. Seventh International Conference on Computational Fluid Dynamics (Hawaii, 2012)*, ICCFD7-3706.
- ITO, MASUMI, YU NISHIO, SEIICHIRO IZAWA, YU FUKUNISHI, and MASAYA SHIGETA [2015]. “Numerical Simulation of Joining Process in a TIG Welding System Using Incompressible SPH Method”. In: *QUARTERLY JOURNAL OF THE JAPAN WELDING SOCIETY*. DOI: [10.2207/qjws.33.34s](https://doi.org/10.2207/qjws.33.34s).
- JESKE, STEFAN RHYS, JAN BENDER, KIRSTEN BOBZIN, HENDRIK HEINEMANN, KEVIN JASUTYN, MAREK SIMON, OLEG MOKROV, RAHUL SHARMA, and UWE REISGEN [2022a]. “Application and Benchmark of SPH for Modeling the Impact in Thermal Spraying”. In: *Computational Particle Mechanics*. DOI: [10.1007/s40571-022-00459-9](https://doi.org/10.1007/s40571-022-00459-9).
- JESKE, STEFAN RHYS, MAREK SEBASTIAN SIMON, OLEKSII SEMENOV, JAN KRUSKA, OLEG MOKROV, RAHUL SHARMA, UWE REISGEN, and JAN BENDER [2022b]. “Quantitative Evaluation of SPH in TIG Spot Welding”. In: *Computational Particle Mechanics*. DOI: [10.1007/s40571-022-00465-x](https://doi.org/10.1007/s40571-022-00465-x).
- JESKE, STEFAN RHYS, LUKAS WESTHOFEN, FABIAN LÖSCHNER, JOSÉ ANTONIO FERNÁNDEZ-FERNÁNDEZ, and JAN BENDER [2023]. “Implicit Surface Tension for SPH Fluid Simulation”. In: *ACM Trans. Graph.* 43.1, 13:1–13:14. DOI: [10.1145/3631936](https://doi.org/10.1145/3631936).
- JIANG, CHENFANFU, CRAIG SCHROEDER, ANDREW SELLE, JOSEPH TERAN, and ALEXEY STOMAKHIN [2015]. “The Affine Particle-in-Cell Method”. In: *ACM Transactions on Graphics*. DOI: [10.1145/2766996](https://doi.org/10.1145/2766996).
- JIANG, CHIYU, AVNEESH SUD, AMEESH MAKADIA, JINGWEI HUANG, MATTHIAS NIESSNER, and THOMAS FUNKHOUSER [2020]. “Local Implicit Grid Representations for 3D Scenes”. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). DOI: [10.1109/CVPR42600.2020.00604](https://doi.org/10.1109/CVPR42600.2020.00604).
- JIANG, YING, CHANG YU, TIANYI XIE, XUAN LI, YUTAO FENG, HUAMIN WANG, MINCHEN LI, HENRY LAU, FENG GAO, YIN YANG, and CHENFANFU JIANG [2024]. “VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality”. In: *ACM SIGGRAPH 2024 Conference Papers*. New York, NY, USA. DOI: [10.1145/3641519.3657448](https://doi.org/10.1145/3641519.3657448).
- KANG, MYUNGJOO, BARRY MERRIMAN, and STANLEY OSHER [2008]. “Numerical Simulations for the Motion of Soap Bubbles Using Level Set Methods”. In: *Computers & Fluids*. DOI: [10.1016/j.compfluid.2007.07.002](https://doi.org/10.1016/j.compfluid.2007.07.002).
- KERBL, BERNHARD, GEORGIOS KOPANAS, THOMAS LEIMKUEHLER, and GEORGE DRETTAKIS [2023]. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3592433](https://doi.org/10.1145/3592433).
- KIM, J and P MOIN [1985]. “Application of a Fractional-Step Method to Incompressible Navier-Stokes Equations”. In: *Journal of Computational Physics*. DOI: [10.1016/0021-9991\(85\)90148-2](https://doi.org/10.1016/0021-9991(85)90148-2).
- KINGMA, DIEDERIK P. and JIMMY BA [2017]. *Adam: A Method for Stochastic Optimization*. DOI: [10.48550/arXiv.1412.6980](https://doi.org/10.48550/arXiv.1412.6980). arXiv: [1412.6980](https://arxiv.org/abs/1412.6980) [cs]. Pre-published.
- KOMEN, HISAYA, MASAYA SHIGETA, and MANABU TANAKA [2018]. “Numerical Simulation of Molten Metal Droplet Transfer and Weld Pool Convection during Gas Metal Arc Welding Using Incompressible

- Smoothed Particle Hydrodynamics Method”. In: *International Journal of Heat and Mass Transfer*. DOI: [10.1016/j.ijheatmasstransfer.2018.01.059](https://doi.org/10.1016/j.ijheatmasstransfer.2018.01.059).
- KOMEN, HISAYA, MANABU TANAKA, and HIDENORI TERASAKI [2020]. “Three-Dimensional Simulation of Gas Metal Arc Welding Process Using Particle-Grid Hybrid Method”. In: *QUARTERLY JOURNAL OF THE JAPAN WELDING SOCIETY*. DOI: [10.2207/qjws.38.25s](https://doi.org/10.2207/qjws.38.25s).
- KOSCHIER, DAN and JAN BENDER [2017]. “Density Maps for Improved SPH Boundary Handling”. In: *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. New York, NY, USA. DOI: [10.1145/3099564.3099565](https://doi.org/10.1145/3099564.3099565).
- KOSCHIER, DAN, JAN BENDER, BARBARA SOLENTHALER, and MATTHIAS TESCHNER [2019]. “Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids”. In: *EUROGRAPHICS 2019 Tutorials*.
- [2022]. “A Survey on SPH Methods in Computer Graphics”. In: *Computer Graphics Forum*. DOI: [10.1111/cgf.14508](https://doi.org/10.1111/cgf.14508).
- KOU, SINDO and D. K. SUN [1985]. “Fluid Flow and Weld Penetration in Stationary Arc Welds”. In: *Metallurgical Transactions A*. DOI: [10.1007/bf02816047](https://doi.org/10.1007/bf02816047).
- KUGELSTADT, TASSILO, JAN BENDER, JOSÉ ANTONIO FERNÁNDEZ-FERNÁNDEZ, STEFAN RHYS JESKE, FABIAN LÖSCHNER, and ANDREAS LONGVA [2021a]. “Fast Corotated Elastic SPH Solids with Implicit Zero-Energy Mode Control”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. DOI: [10.1145/3480142](https://doi.org/10.1145/3480142).
- KUGELSTADT, TASSILO, ANDREAS LONGVA, NILS THUEREY, and JAN BENDER [2021b]. “Implicit Density Projection for Volume Conserving Liquids”. In: *IEEE Transactions on Visualization and Computer Graphics*. DOI: [10.1109/TVCG.2019.2947437](https://doi.org/10.1109/TVCG.2019.2947437).
- LANDAU, L. D. and E. M. LIFSHITZ [2013]. *Fluid Mechanics*. 554 pp.
- LENNARD-JONES, J E [1931]. “Cohesion”. In: *Proceedings of the Physical Society*. DOI: [10.1088/0959-5309/43/5/301](https://doi.org/10.1088/0959-5309/43/5/301).
- LI, MINCHEN, ZACHARY FERGUSON, TESEO SCHNEIDER, TIMOTHY LANGLOIS, DENIS ZORIN, DANIELE PANOZZO, CHENFANFU JIANG, and DANNY M. KAUFMAN [2020]. “Incremental Potential Contact: Intersection-and Inversion-Free, Large-Deformation Dynamics”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3386569.3392425](https://doi.org/10.1145/3386569.3392425).
- LI, SHENGTAO, GE GAO, YUDONG LIU, MING GU, and YU-SHEN LIU [2025]. “Implicit Filtering for Learning Neural Signed Distance Functions from 3D Point Clouds”. In: *Computer Vision – ECCV 2024*. Cham. DOI: [10.1007/978-3-031-72658-3_14](https://doi.org/10.1007/978-3-031-72658-3_14).
- LI, WEI, YIXIN CHEN, MATHIEU DESBRUN, CHANGXI ZHENG, and XIAOPEI LIU [2020]. “Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling”. In: *ACM Trans. Graph.* DOI: [10.1145/3386569.3392400](https://doi.org/10.1145/3386569.3392400).
- LI, ZHAOSHUO, THOMAS MÜLLER, ALEX EVANS, RUSSELL H. TAYLOR, MATHIAS UNBERATH, MING-YU LIU, and CHEN-HSUAN LIN [2023]. “Neuralangelo: High-Fidelity Neural Surface Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- LIN, GUYING, LEI YANG, CONGYI ZHANG, HAO PAN, YUHAN PING, GUODONG WEI, TAKU KOMURA, JOHN KEYSER, and WENPING WANG [2023]. *Patch-Grid: An Efficient and Feature-Preserving Neural Implicit Surface Representation*. DOI: [10.48550/arXiv.2308.13934](https://doi.org/10.48550/arXiv.2308.13934). arXiv: 2308.13934 [cs]. Pre-published.
- LIU, M. B. and G. R. LIU [2010]. “Smoothed Particle Hydrodynamics (SPH): An Overview and Recent Developments”. In: *Archives of Computational Methods in Engineering*. DOI: [10.1007/s11831-010-9040-7](https://doi.org/10.1007/s11831-010-9040-7).
- LIU, SHUSEN, XIAOWEI HE, WENCHENG WANG, and ENHUA WU [2022]. “Adapted SIMPLE Algorithm for Incompressible SPH Fluids with a Broad Range Viscosity”. In: *IEEE Transactions on Visualization and Computer Graphics*. DOI: [10.1109/tvcg.2021.3055789](https://doi.org/10.1109/tvcg.2021.3055789).

- LIU, ZHIJIAN, HAOTIAN TANG, YUJUN LIN, and SONG HAN [2019]. “Point-Voxel CNN for Efficient 3D Deep Learning”. In: *Advances in Neural Information Processing Systems*.
- LIU, ZHIJIAN, HAOTIAN TANG, SHENGYU ZHAO, KEVIN SHAO, and SONG HAN [2022]. “PVNAS: 3D Neural Architecture Search With Point-Voxel Convolution”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. DOI: [10.1109/TPAMI.2021.3109025](https://doi.org/10.1109/TPAMI.2021.3109025).
- LOMBARDI, STEPHEN, TOMAS SIMON, JASON SARAGIH, GABRIEL SCHWARTZ, ANDREAS LEHRMANN, and YASER SHEIKH [2019]. “Neural Volumes: Learning Dynamic Renderable Volumes from Images”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3306346.3323020](https://doi.org/10.1145/3306346.3323020).
- LONG, XIAOXIAO, CHENG LIN, LINGJIE LIU, YUAN LIU, PENG WANG, CHRISTIAN THEOBALT, TAKU KOMURA, and WENPING WANG [2023]. “NeuralUDF: Learning Unsigned Distance Fields for Multi-View Reconstruction of Surfaces With Arbitrary Topologies”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- LÖSCHNER, FABIAN, TIMNA BÖTTCHER, STEFAN RHYS JESKE, and JAN BENDER [2023]. “Weighted Laplacian Smoothing for Surface Reconstruction of Particle-based Fluids”. In: 9 pages. DOI: [10.2312/VMV.20231245](https://doi.org/10.2312/VMV.20231245).
- LÖSCHNER, FABIAN, ANDREAS LONGVA, STEFAN JESKE, TASSILO KUGELSTADT, and JAN BENDER [2020]. “Higher-Order Time Integration for Deformable Solids”. In: *Computer Graphics Forum*. DOI: [10.1111/cgf.14110](https://doi.org/10.1111/cgf.14110).
- LOSHCHILOV, ILYA and FRANK HUTTER [2019]. *Decoupled Weight Decay Regularization*. DOI: [10.48550/arXiv.1711.05101](https://doi.org/10.48550/arXiv.1711.05101). arXiv: [1711.05101](https://arxiv.org/abs/1711.05101) [cs]. Pre-published.
- LUCY, L. B. [1977]. “A Numerical Approach to the Testing of the Fission Hypothesis”. In: *Astronomical Journal*. DOI: [10.1086/112164](https://doi.org/10.1086/112164).
- LYU, CHAOYANG, WEI LI, MATHIEU DESBRUN, and XIAOPEI LIU [2021]. “Fast and Versatile Fluid-Solid Coupling for Turbulent Flow Simulation”. In: *ACM Trans. Graph.* DOI: [10.1145/3478513.3480493](https://doi.org/10.1145/3478513.3480493).
- MA, BAORUI, ZHIZHONG HAN, YU-SHEN LIU, and MATTHIAS ZWICKER [2021]. “Neural-Pull: Learning Signed Distance Function from Point Clouds by Learning to Pull Space onto Surface”. In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning.
- MEDALE, MARC, SYLVAIN RABIER, and CHARLINE XHAARD [2004]. “A Thermo-Hydraulic Numerical Model for High Energy Welding Processes”. In: *Revue Européenne des Eléments*.
- MEDALE, MARC, CHARLINE TOUVREY, and RÉMY FABBRO [2008]. “An Axi-Symmetric Thermo-Hydraulic Model to Better Understand Spot Laser Welding”. In: *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*.
- MEHTA, ISHIT, MICHAËL GHARBI, CONNELLY BARNES, ELI SHECHTMAN, RAVI RAMAMOORTHY, and MANMOHAN CHANDRAKER [2021]. “Modulated Periodic Activations for Generalizable Local Functional Representations”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA. DOI: [10.1109/ICCV48922.2021.01395](https://doi.org/10.1109/ICCV48922.2021.01395). arXiv: [2104.03960](https://arxiv.org/abs/2104.03960) [cs].
- MESCHEDER, LARS, MICHAEL OECHSLE, MICHAEL NIEMEYER, SEBASTIAN NOWOZIN, and ANDREAS GEIGER [2019]. “Occupancy Networks: Learning 3D Reconstruction in Function Space”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). DOI: [10.1109/CVPR.2019.00459](https://doi.org/10.1109/CVPR.2019.00459).
- MILDENHALL, BEN, PRATUL P. SRINIVASAN, MATTHEW TANCIK, JONATHAN T. BARRON, RAVI RAMAMOORTHY, and REN NG [2020]. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *Computer Vision – ECCV 2020*. Cham. DOI: [10.1007/978-3-030-58452-8_24](https://doi.org/10.1007/978-3-030-58452-8_24).
- MISZTAL, MAREK KRZYSZTOF, KENNY ERLEBEN, ADAM BARGTEIL, JENS FURSUND, BRIAN BUNCH CHRISTENSEN, J. ANDREAS BÆRENTZEN, and ROBERT BRIDSON [2012]. *Multiphase Flow of Immiscible Fluids on Unstructured Moving Meshes*. The Eurographics Association. DOI: [10.2312/SCA/SCA12/097-106](https://doi.org/10.2312/SCA/SCA12/097-106).

- MISZTAL, MAREK KRZYSZTOF, KENNY ERLEBEN, ADAM BARGTEIL, JENS FURSUND, BRIAN BUNCH CHRISTENSEN, JAKOB ANDREAS BAERENTZEN, and ROBERT BRIDSON [2014]. “Multiphase Flow of Immiscible Fluids on Unstructured Moving Meshes”. In: *IEEE Transactions on Visualization and Computer Graphics*. DOI: [10.1109/tvcg.2013.97](https://doi.org/10.1109/tvcg.2013.97).
- MOKROV, OLEG, MAREK SIMON, IVAN SHVARTC, RAHUL SHARMA, and UWE REISGEN [2021]. “Validation of the EDACC Model for GMAW Process Simulation by Weld Pool Dimension Comparison”. In: *Lecture Notes in Mechanical Engineering*. DOI: [10.1007/978-3-030-70332-5_5](https://doi.org/10.1007/978-3-030-70332-5_5).
- MONAGHAN, J.J. [1989]. “On the Problem of Penetration in Particle Methods”. In: *Journal of Computational Physics*. DOI: [10.1016/0021-9991\(89\)90032-6](https://doi.org/10.1016/0021-9991(89)90032-6).
- [1992]. “Smoothed Particle Hydrodynamics”. In: *Annual Review of Astronomy and Astrophysics*. DOI: [10.1146/annurev.aa.30.090192.002551](https://doi.org/10.1146/annurev.aa.30.090192.002551).
- [2000]. “SPH without a Tensile Instability”. In: *Journal of Computational Physics*. DOI: [10.1006/jcph.2000.6439](https://doi.org/10.1006/jcph.2000.6439).
- MORRIS, JOSEPH P. [2000]. “Simulating Surface Tension with Smoothed Particle Hydrodynamics”. In: *International Journal for Numerical Methods in Fluids*. DOI: [10.1002/1097-0363\(20000615\)33:3<333::aid-fld11>3.0.co;2-7](https://doi.org/10.1002/1097-0363(20000615)33:3<333::aid-fld11>3.0.co;2-7).
- MOSTAGHIMI, JAVAD and SANJEEV CHANDRA [2018]. “Droplet Impact and Solidification in Plasma Spraying”. In: *Handbook of Thermal Science and Engineering*. Cham. DOI: [10.1007/978-3-319-26695-4_78](https://doi.org/10.1007/978-3-319-26695-4_78).
- MÜLLER, MATTHIAS, DAVID CHARYPAR, and MARKUS GROSS [2003]. “Particle-Based Fluid Simulation for Interactive Applications”. In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Goslar, DEU.
- MÜLLER, THOMAS, ALEX EVANS, CHRISTOPH SCHIED, and ALEXANDER KELLER [2022]. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *ACM Trans. Graph.* DOI: [10.1145/3528223.3530127](https://doi.org/10.1145/3528223.3530127).
- NAUMANN, UWE [2012]. *The Art of Differentiating Computer Programs: An Introduction to Algorithmic Differentiation*. Philadelphia, Pa. 340 pp.
- NELKON, MICHAEL [1969]. *Mechanics and Properties of Matter*.
- NGUYEN, MINH CHIEN, MARC MEDALE, OLIVIER ASSERIN, STEPHANE GOUNAND, and PHILIPPE GILLES [2017]. “Sensitivity to Welding Positions and Parameters in Gta Welding with a 3d Multiphysics Numerical Model”. In: *Numerical Heat Transfer, Part A: Applications*.
- NI, XINGYU, BO ZHU, BIN WANG, and BAOQUAN CHEN [2020]. “A Level-Set Method for Magnetic Substance Simulation”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3386569.3392445](https://doi.org/10.1145/3386569.3392445).
- OUASFI, AMINE and ADNANE BOUKHAYMA [2022]. “Few ‘Zero Level Set’-Shot Learning of Shape Signed Distance Functions in Feature Space”. In: *Computer Vision – ECCV 2022*. Cham. DOI: [10.1007/978-3-031-19824-3_33](https://doi.org/10.1007/978-3-031-19824-3_33).
- PARK, JEONG JOON, PETER FLORENCE, JULIAN STRAUB, RICHARD NEWCOMBE, and STEVEN LOVEGROVE [2019]. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). DOI: [10.1109/CVPR.2019.00025](https://doi.org/10.1109/CVPR.2019.00025).
- PASANDIDEH-FARD, M., S. CHANDRA, and J. MOSTAGHIMI [2002a]. “A Three-Dimensional Model of Droplet Impact and Solidification”. In: *International Journal of Heat and Mass Transfer*.
- PASANDIDEH-FARD, M., V. PERSHIN, S. CHANDRA, and J. MOSTAGHIMI [2002b]. “Splat Shapes in a Thermal Spray Coating Process: Simulations and Experiments”. In: *Journal of Thermal Spray Technology*.
- PASANDIDEH-FARD, M., Y. M. QIAO, S. CHANDRA, and J. MOSTAGHIMI [1996]. “Capillary Effects during Droplet Impact on a Solid Surface”. In: *Physics of Fluids*. DOI: [10.1063/1.868850](https://doi.org/10.1063/1.868850).

- PASZKE, ADAM, SAM GROSS, FRANCISCO MASSA, ADAM LERER, JAMES BRADBURY, GREGORY CHANAN, TREVOR KILLEEN, ZEMING LIN, NATALIA GIMELSHEIN, LUCA ANTIGA, ALBAN DESMAISON, ANDREAS KOPF, EDWARD YANG, ZACHARY DeVITO, MARTIN RAISON, ALYKHAN TEJANI, SASANK CHILAMKURTHY, BENOIT STEINER, LU FANG, JUNJIE BAI, and SOUMITH CHINTALA [2019]. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems*.
- PATKAR, SAKET, MRIDUL AANJANEYA, DMITRIY KARPMAN, and RONALD FEDKIW [2013]. “A Hybrid Lagrangian-Eulerian Formulation for Bubble Generation and Dynamics”. In: *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. DOI: [10.1145/2485895.2485912](https://doi.org/10.1145/2485895.2485912).
- PAWLOWSKI [2008]. *Science and Engineering of Thermal 2e*. 660 pp.
- PEER, ANDREAS, CHRISTOPH GISSLER, STEFAN BAND, and MATTHIAS TESCHNER [2018]. “An Implicit SPH Formulation for Incompressible Linearly Elastic Solids”. In: *Computer Graphics Forum*. DOI: [10.1111/cgf.13317](https://doi.org/10.1111/cgf.13317).
- PENG, SONGYOU, CHIYU JIANG, YIYI LIAO, MICHAEL NIEMEYER, MARC POLLEFEYS, and ANDREAS GEIGER [2021]. “Shape As Points: A Differentiable Poisson Solver”. In: *Advances in Neural Information Processing Systems*.
- PENG, SONGYOU, MICHAEL NIEMEYER, LARS MESCHEDER, MARC POLLEFEYS, and ANDREAS GEIGER [2020]. “Convolutional Occupancy Networks”. In: *Computer Vision – ECCV 2020*. Cham. DOI: [10.1007/978-3-030-58580-8_31](https://doi.org/10.1007/978-3-030-58580-8_31).
- POPINET, STÉPHANE [2018]. “Numerical Models of Surface Tension”. In: *Annual Review of Fluid Mechanics*. DOI: [10.1146/annurev-fluid-122316-045034](https://doi.org/10.1146/annurev-fluid-122316-045034).
- PRICE, DANIEL J. [2012]. “Smoothed Particle Hydrodynamics and Magnetohydrodynamics”. In: *Journal of Computational Physics*. DOI: [10.1016/j.jcp.2010.12.011](https://doi.org/10.1016/j.jcp.2010.12.011).
- QI, CHARLES R., HAO SU, KAICHUN MO, and LEONIDAS J. GUIBAS [2017a]. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- QI, CHARLES RUIZHONGTAI, LI YI, HAO SU, and LEONIDAS J. GUIBAS [2017b]. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *Advances in Neural Information Processing Systems*.
- RAHAMAN, NASIM, ARISTIDE BARATIN, DEVANSH ARPIT, FELIX DRAXLER, MIN LIN, FRED HAMPRECHT, YOSHUA BENGIO, and AARON COURVILLE [2019]. “On the Spectral Bias of Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning*. International Conference on Machine Learning.
- RAISSI, M., P. PERDIKARIS, and G. E. KARNIAKAKIS [2019]. “Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations”. In: *Journal of Computational Physics*. DOI: [10.1016/j.jcp.2018.10.045](https://doi.org/10.1016/j.jcp.2018.10.045).
- REN, XUANCHI, JIAHUI HUANG, XIAOHUI ZENG, KEN MUSETH, SANJA FIDLER, and FRANCIS WILLIAMS [2024]. “XCube: Large-Scale 3D Generative Modeling Using Sparse Voxel Hierarchies”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- RUAN, LIANGWANG, JINYUAN LIU, BO ZHU, SHINJIRO SUEDA, BIN WANG, and BAOQUAN CHEN [2021]. “Solid-Fluid Interaction with Surface-Tension-Dominant Contact”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3450626.3459862](https://doi.org/10.1145/3450626.3459862).
- SCHENK, OLAF and KLAUS GÄRTNER [2004]. “Solving Unsymmetric Sparse Systems of Linear Equations with Pardiso”. In: *Future Generation Computer Systems*.
- SCHROEDER, CRAIG, WEN ZHENG, and RONALD FEDKIW [2012]. “Semi-Implicit Surface Tension Formulation with a Lagrangian Surface Mesh on an Eulerian Simulation Grid”. In: *Journal of Computational Physics*. DOI: [10.1016/j.jcp.2011.11.021](https://doi.org/10.1016/j.jcp.2011.11.021).

- SEMENOV, O, V DEMCHENKO, I KRIVTSUN, et al. [2012]. “Modelling of the Droplet Formation Process in GMA Welding”. In: *Proc. of 10th Int. Sem. on Numerical Analysis of Weldability*.
- SEMENOV, OP, IV KRIVTSUN, AV LYKHOSHVA, OI HLUCHENKYI, and OI BONDAR [2021]. “Comparative Analysis of the Results of Computer Simulation of Heat Transfer and Hydrodynamic Processes in the Metal Being Welded by Means of Different Software Tools”. In: *Paton Welding Journal*. DOI: [10.37434/as2021.01.04](https://doi.org/10.37434/as2021.01.04).
- SEMYONOV, AP [2014]. “Methods of Mathematical Modelling of the Processes of Electrode Metal Drop Formation and Transfer in Consumable Electrode Welding”. In: *Paton Welding Journal*.
- SITZMANN, VINCENT, ERIC CHAN, RICHARD TUCKER, NOAH SNAVELY, and GORDON WETZSTEIN [2020a]. “MetaSDF: Meta-Learning Signed Distance Functions”. In: *Advances in Neural Information Processing Systems*.
- SITZMANN, VINCENT, JULIEN MARTEL, ALEXANDER BERGMAN, DAVID LINDELL, and GORDON WETZSTEIN [2020b]. “Implicit Neural Representations with Periodic Activation Functions”. In: *Advances in Neural Information Processing Systems*.
- SMITH, JONATHAN D., KAMYAR AZIZZADENESHELI, and ZACHARY E. ROSS [2021]. “EikoNet: Solving the Eikonal Equation With Deep Neural Networks”. In: *IEEE Transactions on Geoscience and Remote Sensing*. DOI: [10.1109/TGRS.2020.3039165](https://doi.org/10.1109/TGRS.2020.3039165).
- SOLENTHALER, B. and R. PAJAROLA [2008]. “Density Contrast SPH Interfaces”. In: *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Goslar, DEU.
- [2009]. “Predictive-Corrective Incompressible SPH”. In: *ACM Trans. Graph.* DOI: [10.1145/1531326.1531346](https://doi.org/10.1145/1531326.1531346).
- TAKIKAWA, TOWAKI, JOEY LITALIEN, KANGXUE YIN, KARSTEN KREIS, CHARLES LOOP, DEREK NOWROUZEZAHRAI, ALEC JACOBSON, MORGAN MCGUIRE, and SANJA FIDLER [2021]. “Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- TANCIK, MATTHEW, PRATUL P. SRINIVASAN, BEN MILDENHALL, SARA FRIDOVICH-KEIL, NITHIN RAGHAVAN, UTKARSH SINGHAL, RAVI RAMAMOORTHY, JONATHAN T. BARRON, and REN NG [2020]. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA.
- TANG, JIAPENG, JIABAO LEI, DAN XU, FEIYING MA, KUI JIA, and LEI ZHANG [2021]. “SA-ConvONet: Sign-Agnostic Optimization of Convolutional Occupancy Networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
- TARTAKOVSKY, ALEXANDRE and PAUL MEAKIN [2005]. “Modeling of Surface Tension and Contact Angles with Smoothed Particle Hydrodynamics”. In: *Physical Review E*. DOI: [10.1103/physreve.72.026301](https://doi.org/10.1103/physreve.72.026301).
- TRAUTMANN, M., M. HERTEL, and U. FÜSSEL [2017]. “Numerical Simulation of TIG Weld Pool Dynamics Using Smoothed Particle Hydrodynamics”. In: *International Journal of Heat and Mass Transfer*. DOI: [10.1016/j.ijheatmasstransfer.2017.08.060](https://doi.org/10.1016/j.ijheatmasstransfer.2017.08.060).
- [2018]. “Numerical Simulation of Weld Pool Dynamics Using a SPH Approach”. In: *Welding in the World*. DOI: [10.1007/s40194-018-0615-5](https://doi.org/10.1007/s40194-018-0615-5).
- VARDELLE, ARMELLE, CHRISTIAN MOREAU, NICKOLAS J. THEMELIS, and CHRISTOPHE CHAZELAS [2014]. “A Perspective on Plasma Spray Technology”. In: *Plasma Chemistry and Plasma Processing*. DOI: [10.1007/s11090-014-9600-y](https://doi.org/10.1007/s11090-014-9600-y).
- VARDELLE, M., A. VARDELLE, A. C. LEGER, P. FAUCHAIS, and D. GOBIN [1995]. “Influence of Particle Parameters at Impact on Splat Formation and Solidification in Plasma Spraying Processes”. In: *Journal of Thermal Spray Technology*. DOI: [10.1007/BF02648528](https://doi.org/10.1007/BF02648528).

- VASWANI, ASHISH, NOAM SHAZEER, NIKI PARMAR, JAKOB USZKOREIT, LLION JONES, AIDAN N GOMEZ, ŁUKASZ KAISER, and ILLIA POLOSUKHIN [2017]. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems*.
- VOLLER, V.R., A.D. BRENT, and C. PRAKASH [1990]. “Modelling the Mushy Region in a Binary Alloy”. In: *Applied Mathematical Modelling*. DOI: [10.1016/0307-904x\(90\)90084-i](https://doi.org/10.1016/0307-904x(90)90084-i).
- WANG, HUI, YONGXU JIN, ANQI LUO, XUBO YANG, and BO ZHU [2020]. “Codimensional Surface Tension Flow Using Moving-Least-Squares Particles”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3386569.3392487](https://doi.org/10.1145/3386569.3392487).
- WANG, MENGDI, YITONG DENG, XIANGXIN KONG, ADITYA H. PRASAD, SHIYING XIONG, and BO ZHU [2021]. “Thin-Film Smoothed Particle Hydrodynamics Fluid”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3450626.3459864](https://doi.org/10.1145/3450626.3459864).
- WANG, PENG-SHUAI, YANG LIU, and XIN TONG [2022]. “Dual Octree Graph Networks for Learning Adaptive Volumetric Shape Representations”. In: *ACM Trans. Graph.* DOI: [10.1145/3528223.3530087](https://doi.org/10.1145/3528223.3530087).
- WANG, XIAO-KUN, XIAO-JUAN BAN, YA-LAN ZHANG, SI-NUO LIU, and PENG-FEI YE [2017]. “Surface Tension Model Based on Implicit Incompressible Smoothed Particle Hydrodynamics for Fluid Simulation”. In: *Journal of Computer Science and Technology*. DOI: [10.1007/s11390-017-1793-0](https://doi.org/10.1007/s11390-017-1793-0).
- WANG, YIFAN, LUKAS RAHMANN, and OLGA SORKINE-HORNUNG [2022]. “Geometry-Consistent Neural Shape Representation with Implicit Displacement Fields”. In: *The Tenth International Conference on Learning Representations*.
- WANG, YUE, YONGBIN SUN, ZIWEI LIU, SANJAY E. SARMA, MICHAEL M. BRONSTEIN, and JUSTIN M. SOLOMON [2019]. “Dynamic Graph CNN for Learning on Point Clouds”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3326362](https://doi.org/10.1145/3326362).
- WANG, ZHENGYI, CHENG LU, YIKAI WANG, FAN BAO, CHONGXUAN LI, HANG SU, and JUN ZHU [2023]. “Pro-lificDreamer: High-fidelity and Diverse Text-to-3D Generation with Variational Score Distillation”. In: *Advances in Neural Information Processing Systems*.
- WEI, SI-TONG, RUI-HUAN WANG, CHUAN-ZHI ZHOU, BAOQUAN CHEN, and PENG-SHUAI WANG [2025]. “OctGPT: Octree-based Multiscale Autoregressive Models for 3D Shape Generation”. In: *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*. New York, NY, USA. DOI: [10.1145/3721238.3730601](https://doi.org/10.1145/3721238.3730601).
- WEILER, MARCEL, DAN KOSCHIER, MAGNUS BRAND, and JAN BENDER [2018]. “A Physically Consistent Implicit Viscosity Solver for SPH Fluids”. In: *Computer Graphics Forum*. DOI: [10.1111/cgf.13349](https://doi.org/10.1111/cgf.13349).
- WESTHOFEN, LUKAS, STEFAN RHYS JESKE, and JAN BENDER [2023]. “A Comparison of Linear Consistent Correction Methods for First-Order SPH Derivatives”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.3, 48:1–48:20. DOI: [10.1145/3606933](https://doi.org/10.1145/3606933).
- WILLIAMS, FRANCIS, ZAN GOJCIC, SAMEH KHAMIS, DENIS ZORIN, JOAN BRUNA, SANJA FIDLER, and OR LITANY [2022]. “Neural Fields As Learnable Kernels for 3D Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- WU, GUANJUN, TAORAN YI, JIEMIN FANG, LINGXI XIE, XIAOPENG ZHANG, WEI WEI, WENYU LIU, QI TIAN, and XINGGANG WANG [2024]. “4D Gaussian Splatting for Real-Time Dynamic Scene Rendering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- WU, XIAOYANG, LI JIANG, PENG-SHUAI WANG, ZHIJIAN LIU, XIHUI LIU, YU QIAO, WANLI OUYANG, TONG HE, and HENGSHUANG ZHAO [2024]. *Point Transformer V3: Simpler, Faster, Stronger*. DOI: [10.48550/arXiv.2312.10035](https://doi.org/10.48550/arXiv.2312.10035). arXiv: [2312.10035](https://arxiv.org/abs/2312.10035) [cs]. Pre-published.
- XIE, TIANYI, MINCHEN LI, YIN YANG, and CHENFANFU JIANG [2023]. “A Contact Proxy Splitting Method for Lagrangian Solid-Fluid Coupling”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3592115](https://doi.org/10.1145/3592115).

- XIE, YIHENG, TOWAKI TAKIKAWA, SHUNSUKE SAITO, OR LITANY, SHIQIN YAN, NUMAIR KHAN, FEDERICO TOMBARI, JAMES TOMPKIN, VINCENT SITZMANN, and SRINATH SRIDHAR [2022]. “Neural Fields in Visual Computing and Beyond”. In: *Computer Graphics Forum*. DOI: [10.1111/cgf.14505](https://doi.org/10.1111/cgf.14505).
- XING, JINGRUI, LIANGWANG RUAN, BIN WANG, BO ZHU, and BAOQUAN CHEN [2022]. “Position-Based Surface Tension Flow”. In: *ACM Trans. Graph.* DOI: [10.1145/3550454.3555476](https://doi.org/10.1145/3550454.3555476).
- XU, ZHI-QIN JOHN, YAOYU ZHANG, and YANYANG XIAO [2019]. “Training Behavior of Deep Neural Network in Frequency Domain”. In: *Neural Information Processing*. Cham. DOI: [10.1007/978-3-030-36708-4_22](https://doi.org/10.1007/978-3-030-36708-4_22).
- YANG, K., MIN LIU, KESONG ZHOU, and CHANGGUANG DENG [2012]. “Recent Developments in the Research of Splat Formation Process in Thermal Spraying”. In: *Journal of Materials*. DOI: [10.1155/2013/260758](https://doi.org/10.1155/2013/260758).
- YANG, LIJING, MILAD RAKHSHA, and DAN NEGRUT [2019]. “Comparison of Surface Tension Models in Smoothed Particles Hydrodynamics Method”. In: *Volume 6: 15th International Conference on Multibody Systems, Nonlinear Dynamics, and Control*. DOI: [10.1115/detc2019-98124](https://doi.org/10.1115/detc2019-98124).
- YANG, SHENG, XIAOWEI HE, HUAMIN WANG, SHENG LI, GUOPING WANG, ENHUA WU, and KUN ZHOU [2016]. “Enriching SPH Simulation by Approximate Capillary Waves.” In: *Symposium on Computer Animation*.
- YANG, TAO, MING LIN, RALPH ROBERT MARTIN, JIAN CHANG, and SHIMIN HU [2016]. “Versatile Interactions at Interfaces for SPH-based Simulations”. In: *Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2016)*.
- YANG, TAO, RALPH R. MARTIN, MING C. LIN, JIAN CHANG, and SHI-MIN HU [2017]. “Pairwise Force SPH Model for Real-Time Multi-Interaction Applications”. In: *IEEE Transactions on Visualization and Computer Graphics*. DOI: [10.1109/TVCG.2017.2706289](https://doi.org/10.1109/TVCG.2017.2706289).
- YARIV, LIOR, OMRI PUNY, ORAN GAFNI, and YARON LIPMAN [2024]. “Mosaic-SDF for 3D Generative Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- ZHANG, BIAO, MATTHIAS NIESSNER, and PETER WONKA [2022]. “3DILG: Irregular Latent Grids for 3D Generative Modeling”. In: *Advances in Neural Information Processing Systems*.
- ZHANG, BIAO, JIAPENG TANG, MATTHIAS NIESSNER, and PETER WONKA [2023]. “3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models”. In: *ACM Transactions on Graphics*. DOI: [10.1145/3592442](https://doi.org/10.1145/3592442).
- ZHANG, MINGYU [2010]. “Simulation of Surface Tension in 2D and 3D with Smoothed Particle Hydrodynamics Method”. In: *Journal of Computational Physics*. DOI: [10.1016/j.jcp.2010.06.010](https://doi.org/10.1016/j.jcp.2010.06.010).
- ZHANG, MINGYU, HUI ZHANG, and LILI ZHENG [2008]. “Numerical Investigation of Substrate Melting and Deformation during Thermal Spray Coating by SPH Method”. In: *Plasma Chemistry and Plasma Processing*. DOI: [10.1007/s11090-008-9158-7](https://doi.org/10.1007/s11090-008-9158-7).
- ZHANG, YIZHONG, HUAMIN WANG, SHUAI WANG, YIYING TONG, and KUN ZHOU [2012]. “A Deformable Surface Model for Real-Time Water Drop Animation”. In: *IEEE Transactions on Visualization and Computer Graphics*. DOI: [10.1109/tvcg.2011.141](https://doi.org/10.1109/tvcg.2011.141).
- ZHAO, ZIBO, ZEQIANG LAI, QINGXIANG LIN, YUNFEI ZHAO, HAOLIN LIU, SHUHUI YANG, YIFEI FENG, MINGXIN YANG, SHENG ZHANG, XIANGHUI YANG, HUIWEN SHI, SICONG LIU, JUNTA WU, YIHANG LIAN, FAN YANG, RUINING TANG, ZEBIN HE, XINZHOU WANG, JIAN LIU, XUHUI ZUO, ZHUO CHEN, BIWEN LEI, HAOHAN WENG, JING XU, YILING ZHU, XINHAI LIU, LIXIN XU, CHANGRONG HU, TIANYU HUANG, LIFU WANG, JIHONG ZHANG, MENG CHEN, LIANG DONG, YIWEN JIA, YULIN CAI, JIAAO YU, YIXUAN TANG, HAO ZHANG, ZHENG YE, PENG HE, RUNZHOU WU, CHAO ZHANG, YONGHAO TAN, JIE XIAO, YANGYU TAO, JIANCHEN ZHU, JINBAO XUE, KAI LIU, CHONGQING ZHAO, XINMING WU, ZHICHAO HU, LEI QIN, JIANBING PENG, ZHAN LI, MINGHUI CHEN, XIPENG ZHANG, LIN NIU, PAIGE WANG, YINGKAI WANG, HAOZHAO KUANG, ZHONGYI FAN, XU ZHENG, WEIHAO ZHUANG, YINGPING HE, TIAN

- LIU, YONG YANG, DI WANG, YUHONG LIU, JIE JIANG, JINGWEI HUANG, and CHUNCHAO GUO [2025]. “Hunyuan3D 2.0: Scaling Diffusion Models for High Resolution Textured 3D Assets Generation”. In: *CoRR*.
- ZHENG, WEN, JUN-HAI YONG, and JEAN-CLAUDE PAUL [2009]. “Simulation of Bubbles”. In: *Graphical Models*. DOI: [10.1016/j.gmod.2009.08.001](https://doi.org/10.1016/j.gmod.2009.08.001).
- ZHENG, WEN, BO ZHU, BYUNGMOON KIM, and RONALD FEDKIW [2015]. “A New Incompressibility Discretization for a Hybrid Particle MAC Grid Representation with Surface Tension”. In: *Journal of Computational Physics*. DOI: [10.1016/j.jcp.2014.08.051](https://doi.org/10.1016/j.jcp.2014.08.051).
- ZHENG, Y. Z., Q. LI, Z. H. ZHENG, J. F. ZHU, and P. L. CAO [2014]. “Modeling the Impact, Flattening and Solidification of a Molten Droplet on a Solid Substrate during Plasma Spraying”. In: *Applied Surface Science*. DOI: [10.1016/j.apsusc.2014.08.032](https://doi.org/10.1016/j.apsusc.2014.08.032).
- ZHU, BO, MINJAE LEE, ED QUIGLEY, and RONALD FEDKIW [2015]. “Codimensional Non-Newtonian Fluids”. In: *ACM Transactions on Graphics*. DOI: [10.1145/2766981](https://doi.org/10.1145/2766981).
- ZHU, BO, ED QUIGLEY, MATTHEW CONG, JUSTIN SOLOMON, and RONALD FEDKIW [2014]. “Codimensional Surface Tension Flow on Simplicial Complexes”. In: *ACM Transactions on Graphics*. DOI: [10.1145/2601097.2601201](https://doi.org/10.1145/2601097.2601201).
- ZHU, ZIHANG, SPYROS KAMNIS, and SAI GU [2015]. “Numerical Study of Molten and Semi-Molten Ceramic Impingement by Using Coupled Eulerian and Lagrangian Method”. In: *Acta Materialia*. DOI: [10.1016/j.actamat.2015.02.010](https://doi.org/10.1016/j.actamat.2015.02.010).
- ZIENKIEWICZ, O.C., R.L. TAYLOR, and P NITHIARASU [2014]. *The Finite Element Method for Fluid Dynamics*.
- ZIENKIEWICZ, OLGIERD C and RAMON CODINA [1995]. “A General Algorithm for Compressible and Incompressible Flow—Part I. The Split, Characteristic-Based Scheme”. In: *International Journal for Numerical Methods in Fluids*.
- ZORILLA, FERNANDO, MARCEL RITTER, JOHANNES SAPPL, WOLFGANG RAUCH, and MATTHIAS HARDERS [2020]. “Accelerating Surface Tension Calculation in SPH via Particle Classification and Monte Carlo Integration”. In: *Computers*. DOI: [10.3390/computers9020023](https://doi.org/10.3390/computers9020023).

Appendix for *HYVE: Hybrid Vertex Encoder for Neural Distance Fields*

A.1 Ablation

This section provides information on the various ablation experiments that we conducted in the context of our work.

A.1.1 Design Ablation

To highlight the impact of different design choices within our network architecture, we first conduct a number of ablation studies. The first covers three decisions within our convolution block in Figure 7.2: 1) using the nearest neighbor instead of linear interpolation to map values from latent grid back to the input points, 2) enabling/ disabling the graph convolutions (GNN), and 3) enabling/ disabling the grid convolutions (CNN).

The results for different combinations are reported in Table A.1. The most impactful component is clearly using the grid convolutions, which also contains the majority of trainable weights. The next subimportant decision is enabling the graph convolution. Furthermore, interpolation seems to have a greater impact on network performance when the graph convolution is also enabled. This is because without the graph convolution, the interpolated features are immediately projected back to the grid, eliminating most implicit information gain. For maximal performance, all options should be enabled.

Next, we investigate the impact of varying the number of nearest neighbors (k) for connecting the input point cloud, using point convolutions [Qi et al. 2017a] instead of edge convolutions [Y. WANG et al. 2019], and finally reversing the resolution of latent grids. We show the results in Table A.2. In the depicted parameter sequence, the grid order has the single largest impact. This suggests that coarse features are extracted earlier and fine features later in the network. Using an increasing number of k consistently improves both CD-L1 and NC. For all experiments we use the settings corresponding to the best result.

In addition to this, we investigate the impact of increasing the number of nearest neighbors (k) in combination with changing the point-to-grid interpolation scheme. The results of this ablation are shown in Figure A.1. It becomes clear that increasing the number of nearest neighbors (k) has a positive impact on all metrics regardless of the transfer method used. However, we also observe that our proposed projection method appears to scale better with increasing k , and it outperforms

Table A.1. The results of comparing different design choices within the structure of our network (see Figure 7.2). The table shows the L1 Chamfer distance, normal consistency, intersection-over-union (IoU) and F-score. GNN refers to graph-convolutions and CNN refers to grid-convolutions. The cell color coding is derived from the mean value, different metrics use different color maps. The arrow indicates whether lower or higher values are better. A darker color corresponds to a better value.

(a) Chamfer distance L1 ↓ mean / std ($\cdot 10^{-2}$)				
GNN	Interpolation on		Interpolation off	
	CNN on	CNN off	CNN on	CNN off
on	2.607 / 0.253	4.637 / 1.936	2.771 / 0.355	4.819 / 2.026
off	3.141 / 0.736	12.40 / 1.270	3.176 / 0.774	12.67 / 1.388

(b) Normal Consistency ↑ mean / std ($\cdot 10^{-2}$)				
GNN	Interpolation on		Interpolation off	
	CNN on	CNN off	CNN on	CNN off
on	95.53 / 1.480	85.75 / 7.193	94.88 / 2.055	85.54 / 8.190
off	92.50 / 4.472	85.85 / 5.997	92.30 / 4.683	85.81 / 5.969

(c) IoU _{0.01} ↑ mean / std ($\cdot 10^{-2}$)				
GNN	Interpolation on		Interpolation off	
	CNN on	CNN off	CNN on	CNN off
on	76.69 / 8.015	34.21 / 15.19	70.35 / 12.30	38.89 / 19.61
off	61.29 / 18.40	27.74 / 10.30	59.42 / 17.65	28.92 / 11.07

(d) F _{0.01} ↑ mean / std ($\cdot 10^{-2}$)				
GNN	Interpolation on		Interpolation off	
	CNN on	CNN off	CNN on	CNN off
on	95.87 / 4.836	51.74 / 21.45	91.48 / 9.472	57.37 / 25.49
off	81.76 / 18.26	35.51 / 12.04	80.59 / 18.49	36.00 / 12.20

Table A.2. Ablation study of the influence of the grid order, graph convolution and the number of nearest neighbors on the performance of our method. CD-L1 denotes the L1 Chamfer distance, NC the normal consistency, IoU the intersection-over-union and F the F-score. The best values are highlighted in bold. All values denote the mean.

GNN	Res.	k-NN	CD-L1 ↓	NC ↑	IoU _{0.01} ↑	F _{0.01} ↑
PointConv	Decr.	None	1.045	96.603	82.651	99.165
EdgeConv	Decr.	2	0.994	97.423	85.719	99.613
EdgeConv	Incr.	2	0.953	97.765	88.754	99.684
EdgeConv	Incr.	4	0.947	97.807	89.197	99.724
EdgeConv	Incr.	8	0.942	97.938	89.542	99.771

pooling-based point-to-grid transfer for all values of $k > 2$. For all of our experiments we otherwise use $k = 8$, as increasing k further appears to result in diminishing returns.

A.1.2 Noise Ablation

We investigate our architecture’s sensitivity to noise by training and testing using different noise levels. We add noise by perturbing input vertices by a random amount drawn from $\mathcal{U}(-\sigma, \sigma)$ in random direction (see Table A.3) or in normal direction (see Table A.4). Here, σ is the noise level shown in the

Table A.3. Comparison of the L1 Chamfer distance, normal consistency, intersection-over-union (IoU) and F-score (F) across different noisy perturbations in random directions of the input points. Rows indicate the noise level used during training and columns during testing. The best value for each metric is marked in bold font.

(a) Chamfer Distance L1 ↓ mean / std ($\cdot 10^{-2}$)					
Train \ Test	none	5e-4	5e-3	1e-2	5e-2
none	3.349	3.348	3.357	3.397	4.198
5e-4	3.358	3.358	3.361	3.403	4.262
5e-3	3.381	3.379	3.397	3.448	4.268
1e-2	3.403	3.404	3.415	3.464	4.345

(b) Normal Consistency ↑ mean / std ($\cdot 10^{-2}$)					
Train \ Test	none	5e-4	5e-3	1e-2	5e-2
none	93.200	93.189	93.063	92.702	81.880
5e-4	93.114	93.097	93.016	92.633	81.777
5e-3	93.001	93.017	92.918	92.616	82.344
1e-2	93.130	93.117	93.037	92.809	83.813

(c) IoU _{0.01} ↑ mean / std ($\cdot 10^{-2}$)					
Train \ Test	none	5e-4	5e-3	1e-2	5e-2
none	77.000	76.950	75.272	71.615	36.144
5e-4	76.598	76.603	75.115	71.503	35.673
5e-3	76.356	76.375	75.261	72.393	36.860
1e-2	73.923	73.934	73.509	71.860	37.767

(d) F _{0.01} ↑ mean / std ($\cdot 10^{-2}$)					
Train \ Test	none	5e-4	5e-3	1e-2	5e-2
none	91.725	91.743	91.480	90.537	48.793
5e-4	91.614	91.606	91.425	90.443	47.880
5e-3	91.459	91.469	91.244	90.532	49.494
1e-2	91.254	91.252	91.110	90.451	50.569

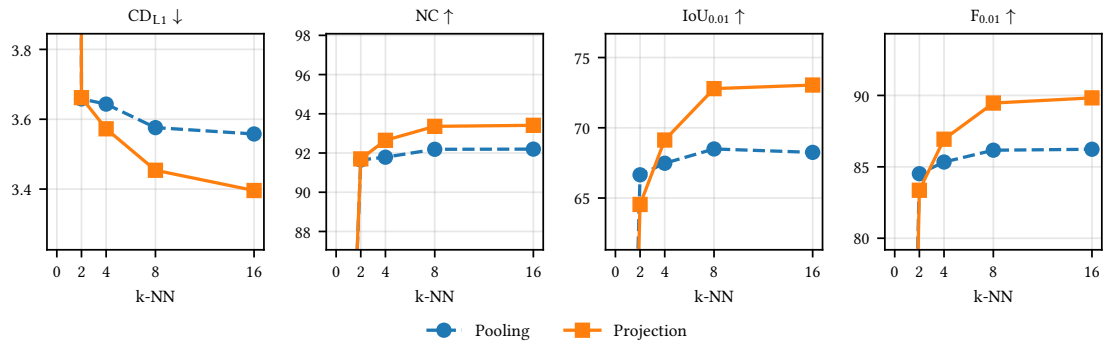


Figure A.1. Ablation study comparing pooling and our projection operator when using different numbers of nearest neighbors (k) to connect input points for the graph convolution. Increasing k consistently results in improved metrics, where our proposed projection operator benefits more and consistently outperforms pooling for $k > 2$. We compare using the L1 Chamfer distance, normal consistency, intersection-over-union (IoU) and F-score.

column labels and row labels of the respective tables. Since the coordinates have been normalized to the unit box, the maximum evaluated noise of 5e-2 corresponds to a shift of 1/40 of the bounding box edge length. Our model remains stable during training up to noise levels of 1e-2 both in normal or random directions and even benefits from data augmentation using noise in normal direction.

Table A.4. Comparison of the L1 Chamfer distance, normal consistency, intersection-over-union (IoU) and F-score (F) across different noisy perturbations in the direction of vertex normals of the input points. Rows indicate the noise level used during training and columns during testing. The best value for each metric is marked in bold font.

(a) Chamfer Distance L1 $\downarrow$ mean / std ($\cdot 10^{-2}$)					
Train \ Test	none	5e-4	5e-3	1e-2	5e-2
none	3.381	3.381	3.429	3.534	4.684
5e-4	3.370	3.371	3.410	3.517	4.642
5e-3	3.365	3.365	3.415	3.531	4.667
1e-2	3.417	3.417	3.452	3.567	4.731

(b) Normal Consistency $\uparrow$ mean / std ($\cdot 10^{-2}$)					
Train \ Test	none	5e-4	5e-3	1e-2	5e-2
none	93.070	93.059	92.770	91.850	77.978
5e-4	93.035	93.033	92.768	91.804	77.722
5e-3	93.254	93.244	93.058	92.503	78.971
1e-2	93.200	93.213	93.165	92.915	80.882

(c) IoU _{0.01} $\uparrow$ mean / std ($\cdot 10^{-2}$)					
Train \ Test	none	5e-4	5e-3	1e-2	5e-2
none	74.459	74.396	71.118	63.869	19.128
5e-4	75.429	75.417	71.846	64.361	19.056
5e-3	75.375	75.372	73.537	68.036	19.557
1e-2	72.430	72.527	72.392	70.457	20.812

(d) F _{0.01} $\uparrow$ mean / std ($\cdot 10^{-2}$)					
Train \ Test	none	5e-4	5e-3	1e-2	5e-2
none	91.198	91.196	90.369	86.334	25.295
5e-4	91.508	91.496	90.706	86.788	25.213
5e-3	91.518	91.512	90.996	88.918	25.550
1e-2	91.138	91.144	90.944	89.934	26.985

A.1.3 Point-to-Grid Feature Projection

In addition to the visual comparison in Figure 7.7, Table A.5 shows the metrics comparing pooling- and projection-based feature transfers across all datasets.

A.1.4 Contours of Volumetric Signed Distance Field

Despite mostly showing only the zero level-set of the computed signed distance field, our network also produces a reasonable volumetric signed distance field. We show this briefly in Figure A.2, where contours are generated at multiple iso-values for the signed distance field.

A.1.5 Non-manifold Handling

Non-manifold meshes, mostly in the form of non-watertightness or self-intersections appear in a number of different datasets (Table 7.1). We show in Figure A.3 using an example from the Objaverse dataset, that our model is able to deal with these self-intersections fairly well.

Table A.5. Comparing our method with pooling-based and projection-based point-to-grid transfers. Optionally, input vertex normals are also used. We evaluate the L1 Chamfer distance (CD-L1), normal consistency (NC), intersection-over-union (IoU) and F-score (F). Bold font marks the best value in each metric and dataset. All values are given scaled to $\cdot 10^{-2}$.

Metric	Method	Dragon	Armadillo	DFAUST	ScanNet	Thing10k	ShapeNet v2	Objaverse
CD _{L1} ↓ mean	Pool w/o n	2.455	0.956	0.627	4.339	3.270	2.011	1.222
	Proj. w/o n	2.406	0.934	0.662	4.721	3.222	1.888	1.363
	Pool w/ n	2.486	0.934	0.700	4.156	3.446		1.118
	Proj. w/ n	2.427	0.906	0.626	4.269	3.075		1.604
NC ↑ mean	Pool w/o n	96.25	97.86	97.65	82.98	93.65	86.11	92.95
	Proj. w/o n	96.33	98.22	97.41	82.27	94.26	86.75	92.00
	Pool w/ n	96.37	98.27	97.61	85.15	95.53		94.87
	Proj. w/ n	96.53	98.47	97.83	85.33	95.89		95.25
IoU _{0.01} ↑ mean	Pool w/o n	84.84	88.63	88.40	54.88	78.85	61.24	65.75
	Proj. w/o n	87.73	89.76	85.82	52.29	80.02	66.96	61.49
	Pool w/ n	80.15	89.01	84.76	57.33	61.74		66.79
	Proj. w/ n	87.30	92.00	88.76	56.50	87.58		86.53
F _{0.01} ↑ mean	Pool w/o n	98.97	99.82	98.17	73.68	92.45	88.53	89.35
	Proj. w/o n	99.47	99.82	97.89	71.01	93.86	93.23	85.28
	Pool w/ n	98.69	99.87	97.53	75.84	81.09		81.02
	Proj. w/ n	99.71	99.89	97.88	74.72	97.47		93.45

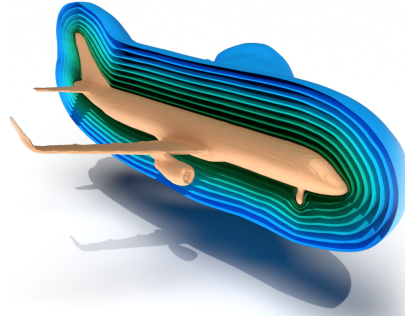


Figure A.2. Visualization of various level-sets using our method on an instance of the planes category from the ShapeNet v2 dataset [CHANG et al. 2015].



Figure A.3. Our model is able to resolve a large number of self-intersections in the Objaverse dataset [DEITKE et al. 2023] without special treatment, as long as surface normals have consistent orientation. The left image shows our reconstruction without self-intersections, while the right image shows the ground-truth shape containing self-intersections. The transparent renders show a close-up of the marked red region to make self-intersections visible.

