



## Detail Synthesis for Fluids and Videos with Deep-Learning Algorithms

Mengyu Chu

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

**Doktors der Naturwissenschaften (Dr. rer. nat.)**

genehmigten Dissertation.

**Vorsitzende:**

Prof. Dr.-Ing. Laura Leal-Taixé

**Prüfende der Dissertation:**

1. Prof. Dr.-Ing. Nils Thuerey
2. Prof. Theodore Kim, Ph.D.  
Yale University

Die Dissertation wurde am 13.05.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 19.08.2020 angenommen.



# Acknowledgement

I would like to express my sincere gratitude to everyone who accompanied me along the pleasant PhD journey. It is a long way with many ups and downs, and with their help, I never felt lonely. Foremost, I would like to express my very great appreciation to Nils Thuerey. As a supervisor, he led me into the world of physics simulations, motivated us to work at the interface between deep learning and physics-based animations, and guided me patiently through each stage of our research. I am particularly grateful for his frank and constructive advice not only on every projects but also on how to approach research in general. Although the path of research is not always smooth, his willingness to generously provide time and support always encourages me to move on.

Furthermore, I highly appreciate the chance to closely collaborated with Rüdiger Westermann and Laura Leal-Taixé. Their experienced suggestions are always extremely helpful. It is my fortune to have Theodore Kim and Laura as my thesis committee. I also want to thank the co-authors of our projects which have contributed to this thesis, You Xie, Erik Franz, Sebastian Weiss, Jonas Mayer, and Maximilian Werhahn, for contributing their efforts, suggestions and ideas. It has always been a pleasure to meet Chris Wojtan, Barbara Solenthaler and their groups through our yearly retreat trips. It is enjoyable to see how everyone conduct interesting approaches in different directions and how we spark exciting discussions. I also want to thank Kun Zhou, Zhong Ren, Qiming Hou and the GAPS Lab of Zhejiang University where I started to learn about computer graphics in the first place.

Moreover, I would like to thank all my colleagues for their company over the years. Special thanks to Kiwon Um for his cheerful personality that enriches the pleasure, to Alexander Kumpf for reliably removing bugs in algorithms as well as in our office, to Marie-Lena Eckert, Kern Michael, and Aljaž Božič for warmly organizing interesting group activities, to You Xie, Steffen Wiewel, Lukas Prantl, Philipp Holl, Liwei Chen, Georg Kohl, Stephan Rasp, Björn List, Wei He, Boris Bonev, Tiffany Inglis and Sebastian Eberhardt for thorough discussions on research topics as well as others. I want to acknowledge the German abstract of the thesis mostly provided by Georg and Erik in addition to a little by Google. My thanks extend to Matthias Niessner, Angela Dai, Justus Thies, and Ji Hou for interesting communications during lunch-break. I am also grateful to have accompany from my co-workers Andreas Rössler, Dejan Azinovic, Zhenyu Chen, Manuel Dahnert, Mathias Kanzler, Henrik Masbruch, Seyedbehdad Ghafari, Kevin Höhle, Christian Reinbold, and Junpeng Wang, as well as generous helps from Susanne Weitz and Sebastian Wohner. I appreciate everyone for making my journey at TUM unforgettable.

Last but not least, I would like to thank my parents, Qinxue Chu and Di Qin, my friends, Haiyan Ma and Mingming He, for their understandings and support without any conditions.



# Abstract

Detail synthesis has been an appealing but challenging research topic in areas of computer vision and graphics for a long time. Despite rapid developments in physically-based simulation, rendering, and image editing, we still lack powerful tools to synthesize animations and videos that match both users' intentions and the realism exhibited by real-world phenomena. In this dissertation, we first focus on detailed flow effects. Then, we also investigate video generation tasks with conditional inputs, e.g. video super-resolution and translation. The central goal of our work is to employ deep-learning algorithms for fluids and videos to understand their complex temporal evolution and to synthesize coherent details that are realistic and appealing.

For fluid simulations, fine details usually require high resolutions and tiny time-steps to reduce numerical viscosity. While these operations are costly and unintuitive to tune, we propose to pre-compute space-time fluid data as a repository and to synthesize new volumes by matching feature descriptors learned by neural networks. In this way, users can quickly tune coarse simulations, and networks can encode local information into descriptors to efficiently retrieve suitable detailed regions from the repository. When training, our goal is not only to encode space-time fluid data but also to establish correspondences between simulations with different numerical viscosity. We advect deformable patches to track flow regions stably. Consequently, the large-scale temporal evolution in the synthesized flow comes from the advection and deformation of patches, while the detailed motion is re-used from the repository. We will demonstrate that these appealing details and small-scale motions can naturally integrate into the underlying coarse flow due to flow-aware descriptors encoding both densities and the curl of flow velocities.

While velocity fields contain temporal information for fluids, videos usually have complex temporal relationships without ground-truth motion. Although deep learning is extremely successful in representing natural images and other data distributions, directly applying them in sequence generation leads to strong temporal artifacts. Hence, we propose temporal self-supervisions for conditional GAN-based video generation tasks. For video super-resolution and unpaired video translation, despite their substantially different challenges, our spatio-temporal adversarial learning achieves coherent solutions without sacrificing spatial detail. We also propose a Ping-Pong loss to improve long-term consistency, which is potentially beneficial for other recurrent networks.

By using neural networks to learn the temporal aspects, our methods can synthesize realistic and coherent fluids and videos. The results presented in this thesis demonstrate that data-driven and deep-learning-based synthesis are promising directions and provide powerful tools. For physically-based animations, shape deformations, video synthesis and many others, efficient and stable algorithms with intuitive user interactions are in high demand. We hope that our methods can provide insights into these sequence generation tasks with convenient user controls.



# Zusammenfassung

Die Synthese von Details ist seit langem ein attraktives aber auch herausforderndes Forschungsthema in den Bereichen Computervision und Computergraphics. Trotz rasanter Entwicklungen von physikalisch-basierten Simulationen, Rendering und Bildbearbeitung fehlt noch immer leistungsstarke Software zur Synthese von Animationen und Videos, die sowohl die Absichten des Nutzers umsetzt als auch realistische Ergebnisse erzeugt. Im ersten Teil dieser Dissertation liegt der Fokus auf detaillierten Strömungseffekten. Anschließend wird die Generierung von Videos aus gegebenen Inputs, wie zum Beispiel Superresolution- und Transferverfahren, untersucht. Das Hauptziel dieser Arbeit ist mithilfe von Deep-Learning Algorithmen die komplexe Entwicklung von Fluiden und Videos zu verstehen und zeitlich zusammenhängende Details zu synthetisieren, die realistisch und ansprechend sind.

Bei Fluidsimulationen sind kleine Details normalerweise mit hohen Auflösungen und kleinen Zeitschritten verbunden, um die numerische Viskosität zu verringern. Da diese Methoden rechenaufwändig und wenig intuitive sind, berechnen wir ein Repository aus Raum-Zeit-Fluiddaten vorab und gleichen Feature-Deskriptoren die von neuronalen Netzen gelernt werden damit ab um neue Volumina zu synthetisieren. Auf diese Weise können Nutzer die Grobjustierung der Simulationen schnell durchführen und das neuronale Netz codiert die lokalen Informationen in Deskriptoren, um geeignete Detailbereiche effizient aus dem Repository abzurufen. Beim Training des Netzwerks ist das Ziel nicht nur Raum-Zeit-Fluiddaten zu codieren, sondern auch Übereinstimmungen zwischen Simulationen mit unterschiedlicher numerischer Viskosität herzustellen. Verformbare Simulationsbereiche werden advektiert, um die Strömungsregionen zeitlich stabil zu verfolgen. Deswegen beruht die hauptsächliche Bewegung der synthetisierten Strömungen auf der Advektion und Verformung der Simulationsbereiche, während die detaillierte Bewegung wird aus dem Repository wiederverwendet wird. Es wird dargelegt, dass sich diese Details und kleinen Bewegungen natürlich in die hauptsächliche Strömung integrieren, da die Flussdeskriptoren sowohl die Dichte als auch die Krümmung der Strömungsgeschwindigkeit codieren.

Während das Geschwindigkeitsfeld einer Fluidsimulation bereits zeitliche Informationen enthält, haben Videos komplexe zeitliche Beziehungen, für die es keine direkte Referenz gibt. Obwohl Deep-Learning bereits erfolgreich für die Darstellung von Bildern und anderen komplexen Datenverteilungen verwendet wird, führt eine direkte Anwendung bei der Sequenzgenerierung zu starken zeitlichen Artefakten. Daher wird in dieser Arbeit zeitliche Eigenüberwachung für die Videogenerierung mit conditional GANs verwendet. Trotz abweichender Anforderungen erzieht unser räumlich-zeitliches Lernen kohärente Ergebnisse für Superresolution und alleinstehende Videotransferverfahren, ohne räumliche Details einzubüßen. Zusätzlich verwenden wir eine Ping-Pong Verlust-

funktion um die langfristige Beständigkeit zu verbessern, welche möglicherweise auch für andere rückgekoppelte Netzwerke von Vorteil ist.

Durch die Anwendung neuronaler Netze zum Erlernen zeitlicher Aspekte kann unsere Methode realistische und kohärente Details für Fluidsimulationen und Videos synthetisieren. Die Ergebnisse dieser Arbeit zeigen, dass die Synthese mit datenbasierten Deep-Learning Methoden eine vielversprechende Forschungsrichtung ist und leistungsstarke Werkzeuge bietet. Es gibt eine starke Nachfrage nach effizienten und stabilen Algorithmen mit intuitiver Nutzerinteraktion für physikalisch-basierten Animation, Formdeformation, Videosynthese und ähnlichen Themen. Wir hoffen, dass unsere Methoden durch praktische Nutzerinteraktion Einblicke in diese Sequenzgenerierungsaufgaben ermöglichen können.



# Contents

<b>Acknowledgement</b>	<b>iii</b>
<b>Abstract</b>	<b>v</b>
<b>Zusammenfassung</b>	<b>vii</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xv</b>
<b>Acronyms</b>	<b>xvii</b>
<b>I Introduction</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Dissertation Overview . . . . .	6
1.2 Publication List . . . . .	7
<b>2 Fundamentals</b>	<b>9</b>
2.1 Fluid Simulations . . . . .	9
2.1.1 Eulerian Methods and Numerical Dissipation . . . . .	10
2.1.2 Lagrangian and Hybrid Methods . . . . .	12
2.1.3 Machine Learning Methods for Fluid Simulations . . . . .	13
2.2 Deep Learning Methods for Images and Videos . . . . .	14
2.2.1 Neural Networks and CNNs . . . . .	15
2.2.2 GANs . . . . .	15
2.2.3 Conditional Generation for Images and Videos . . . . .	17
2.2.4 Image and Video Evaluations . . . . .	18
<b>II Detail Synthesis for Fluids</b>	<b>21</b>
<b>3 Learning Flow Similarity</b>	<b>25</b>
3.1 CNN Architecture . . . . .	26
3.2 Loss Functions . . . . .	28
3.3 Data Generation . . . . .	30
3.4 Evaluation . . . . .	32
3.5 Descriptors for Flow Motions . . . . .	34

<b>4</b>	<b>Fluid Synthesis Based on Similarity</b>	<b>37</b>
4.1	Texture Synthesis for Fluid Simulations . . . . .	38
4.2	Direct Density Synthesis . . . . .	39
4.3	Patch Motion and Synthesis . . . . .	40
4.3.1	Deformation-limiting Motion . . . . .	40
4.3.2	Synthesis and Rendering . . . . .	45
4.4	Results and Discussion . . . . .	47
4.5	Conclusions . . . . .	51
<b>III</b>	<b>Detail Synthesis for Videos</b>	<b>53</b>
<b>5</b>	<b>Temporally Coherent Video Super-Resolution</b>	<b>57</b>
5.1	Network Architectures and Temporal Self-Supervisions . . . . .	58
5.1.1	Generative Network . . . . .	58
5.1.2	Spatio-Temporal Adversarial Supervision . . . . .	59
5.1.3	Long-Term Temporal Supervision . . . . .	60
5.2	Training and Loss Summary . . . . .	62
5.3	Learning Objectives Analysis . . . . .	63
5.3.1	Loss Ablation Study . . . . .	63
5.3.2	Data Augmentation and Temporal Constrains in the PP loss . . .	65
5.4	Results and Performance . . . . .	66
5.5	Limitations and Conclusions . . . . .	66
<b>6</b>	<b>Unpaired Video Translation with Temporal Self-Supervisions</b>	<b>69</b>
6.1	Network Architectures . . . . .	70
6.1.1	Frame-Recurrent Generative Network . . . . .	70
6.1.2	Spatio-Temporal Discriminative Network . . . . .	71
6.2	Losses and Analysis . . . . .	72
6.2.1	Loss Ablation Study . . . . .	73
6.2.2	Spatio-temporal Adversarial Equilibriums . . . . .	75
6.3	Results and Conclusions . . . . .	77
<b>7</b>	<b>Spatio-Temporal Evaluations for Videos</b>	<b>79</b>
7.1	Evaluations Using Spatial Metrics . . . . .	79
7.2	Temporal Metrics and Evaluations . . . . .	81
7.3	User Studies . . . . .	84
7.4	Discussion and Conclusions . . . . .	87
<b>IV</b>	<b>Conclusions of Detail Synthesis for Sequential Data</b>	<b>89</b>
<b>8</b>	<b>Coherent Details for Sequential Data</b>	<b>91</b>

<b>9</b>	<b>Conclusions and Future Work</b>	<b>95</b>
9.1	Method Summary . . . . .	95
9.2	Future Work . . . . .	96
	<b>Bibliography</b>	<b>99</b>
	<b>Appendix</b>	<b>111</b>
<b>A</b>	<b>Technical Details for Video Generation</b>	<b>113</b>
A.1	Network Architecture . . . . .	113
A.2	Training Details . . . . .	114
A.2.1	Technical Details of the Spatio-Temporal Discriminator in VSR . .	114
A.2.2	Training Details for VSR and UVT . . . . .	115



# List of Figures

1.1	Examples of detail synthesis methods . . . . .	4
1.2	Remote usage of visual content is desired. . . . .	5
1.3	Developments on detail synthesis for images, fluids, and videos . . . . .	5
1.4	An overview of methods discussed in the dissertation. . . . .	6
2.1	Computational cost of CG solvers . . . . .	11
2.2	Numerical dissipation . . . . .	12
2.3	Neural Networks and CNNs . . . . .	14
2.4	GANs . . . . .	16
2.5	The perception-distortion trade off. . . . .	19
3.1	Similarity problems caused by viscosity . . . . .	25
3.2	CNN architecture . . . . .	27
3.3	Positive and negative data pairs . . . . .	28
3.4	Concept figures of the descriptor space when training with $l_n$ and $l_e$ . . . . .	29
3.5	Training data generation . . . . .	30
3.6	Recall over rank for HOG, CNN using $l_h$ and $l_e$ . . . . .	33
3.7	Top-ranking density pairs matched by our CNN. . . . .	33
3.8	Recall over rank for HOG, CNN with and without motion . . . . .	34
3.9	Curl comparison. . . . .	35
3.10	Recall over rank for 3D . . . . .	35
4.1	Method overview . . . . .	37
4.2	Texture synthesis methods for fluid simulations . . . . .	38
4.3	Comparison to the direct synthesis method . . . . .	39
4.4	Patch advection with and without deformation control. . . . .	40
4.5	An example of patch cage with $n = 3$ . . . . .	41
4.6	Patch anticipation comparisons . . . . .	43
4.7	A horizontal plume simulation. . . . .	48
4.8	Descriptor comparisons. . . . .	48
4.9	More comparisons. . . . .	48
4.10	A simulation with a obstacle. . . . .	49
4.11	Two colliding jets of smoke simulated with our approach. . . . .	49
4.12	Comparison to a full simulation. . . . .	50
5.1	Our method generates sharp and coherent VSR results. . . . .	57
5.2	The frame-recurrent VSR generator based on motion compensation. . . . .	58
5.3	Conditional VSR $D_{s,t}$ . . . . .	59

5.4	The Ping-Pang loss. . . . .	61
5.5	VSR results of the foliage scene. . . . .	63
5.6	VSR results of the calendar scene. . . . .	64
5.7	Comparing the augmentation and the temporal constraint in PP Loss. . .	65
5.8	Detail views of the VSR results . . . . .	67
5.9	VSR results on captured images comparing to previous work [131, 94]. . .	67
5.10	Additional VSR Comparisons . . . . .	68
6.1	Our method generates sharp and coherent UVT results. . . . .	69
6.2	Frame-recurrent UVT generator with cycle consistency . . . . .	70
6.3	Conditional UVT $D_{s,t}$ . . . . .	71
6.4	Loss ablation study on the smoke rendering translation. . . . .	74
6.5	Loss ablation study on the translation between Trump and Obama. . . . .	75
6.6	Analysis on discriminator inputs . . . . .	76
6.7	Results of UVT tasks on different data-sets. . . . .	77
7.1	Visual summary of VSR models. . . . .	79
7.2	Bar graphs of temporal metrics for Vid4. . . . .	82
7.3	Spatial metrics for Vid4. . . . .	82
7.4	Metrics for ToS. . . . .	82
7.5	Visual summary using PieAPP, LPIPS and tOF for the VSR of Vid4. . .	84
7.6	Optical flow and tOF visualization of the armor scene. . . . .	84
7.7	A sample setup of user study. . . . .	85
7.8	User study scores for Vid4 VSR . . . . .	86
7.9	User study scores for Obama&Trump UVT. . . . .	87
7.10	Temporal metric evaluation correlates well to the user-studies. . . . .	87
8.1	A multi-pass GAN. . . . .	92
8.2	SR for rendering sequences . . . . .	93
9.1	The connections of generative methods for fluids, videos and renderings. .	96
A.1	Boundaries are cropped since flow estimation is less accurate. . . . .	115

# List of Tables

4.1	Details of our animation setups and repository data generation. . . . .	47
7.1	Metrics evaluated for the VSR Vid4 scenes. . . . .	80
7.2	Metrics evaluated for VSR of ToS scenes. . . . .	81
7.3	Temporal evaluation for the Obama&Trump translation task. . . . .	83
7.4	A summary of the evaluations on the Vid4 data-set . . . . .	88
A.1	Training parameters . . . . .	116





# Acronyms

$L^2$ or $\ \mathbf{x}\ _2$	The $l_2$ -norm.
1D, 2D, 3D, ...	n spatial dimentionions.
2AFC	two-alternative forced choice.
CNNs	Convolutional Neural Networks.
FLIP	fluid implicit particle.
GANs	Generative Adversarial Networks.
HR	high resolution.
KL divergence	Kullback-Leibler Divergence.
LPIPS	learned perceptual image patch similarity.
LR	low resolution.
MPM	material point method.
OF	optical flow.
PCG	preconditioned conjugate gradient.
PIC	particle-in-cell.
PSNR	peak signal-to-noise ratio.
SPH	smoothed-particle hydrodynamics.
UVT	unpaired video translation.
VSR	video super-resolution.



## **Part I**

# **Introduction**





*“The difference between something good and something great is attention to detail.”*

*— Charles R. Swindoll*

## 1 Introduction

The ultimate goal of computer graphics is to allow people to express themselves with natural images that are indistinguishable from real-world scenes. Despite the rapid developments of physics-based algorithms, producing photo-realistic results with regard to user requirements remains highly challenging due to the intrinsic complexity of real physics. Approaching this goal, detail synthesis offers a different perspective. Instead of constructing virtual scenes entirely in a physics-based manner, detail synthesis provides a convenient mechanism for users to perform rough edits and enhances the realism of the results by adding fine details. Examples are given in Fig. 1.1. Through intuitive user interactions, in-depth information about the properties of the objects, such as the material of the cloth and the physical state of the fluid, is delivered by the illuminated textures [1] and turbulent vortices [2].

Besides effectively improving visual experiences, detail synthesis is also a convenient tool for remote usage. With advancements of high-speed networks, live streaming and cloud gaming have expanded to ordinary Internet users since 2010s [3, 4]. While a few



(a) Cartoon animations generated from sketches and textures using TexToons [1] (b) High-resolution fluid synthesis using wavelet turbulence [2]

**Figure 1.1:** With detail synthesis methods, users can design with interactive controls, e.g. strokes and coarse fluid simulations shown on the left of a) and b), and achieve high-quality results with rich details shown on the right.

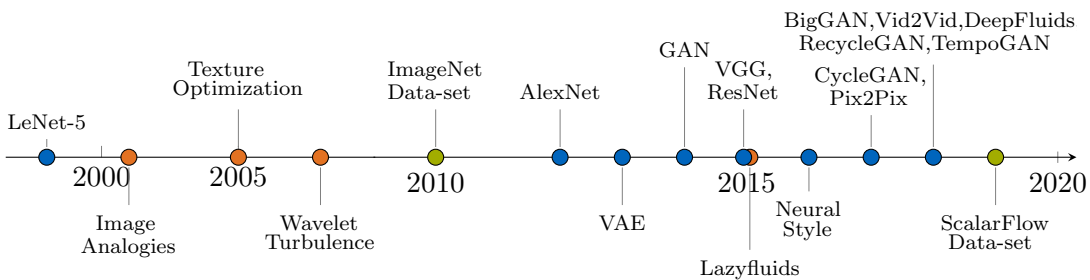
games are available on Google Stadia, as shown on the left of Fig. 1.2, the network latency due to large image transmission is still a major complain from consumers. In addition to entertainment such as games and videos, remote visualization services have become an essential part of business for solving increasingly complex computational tasks. Although it is possible to view simple information such as near real-time global temperatures on NASA’s Eyes, as shown on the right of Fig. 1.2, there is still a long way to go for complex remote tasks such as monitoring natural disasters. In such a circumstance, there is an urgent demand for technologies to reduce bandwidth across all areas of remote applications. From this point of view, detail synthesis methods have attracted more and more attention. With a decent visual quality maintained by synthesizing details on client computers, data can be transmitted using compressed representations and more operations are allowed interactively.

Being a very important direction of research for computer graphics and vision, detail synthesis technologies develop rapidly. Traditionally, detail for images usually comes from exemplars describing Markov random fields [7, 8], while in physics-based problems, detail can be inferred from physical models such as fine-scale simulations [9]. In recent years, the increasing computational power and data storage capacity have brought us into the Big Data era. With a large amount of data available from virtual simulations to real captures, data-driven methods and machine-learning algorithms show their advantages in a variety of tasks. Methods including Generative Adversarial Networks (GANs) [10] achieve state-of-the-art performances in understanding and generating complex distributions such as natural images. While new opportunities are offered by these techniques, applying them to detail synthesis is a challenging task for a number of reasons. Since visual continuity plays a vital role in a good user experience, realistic temporal evolu-



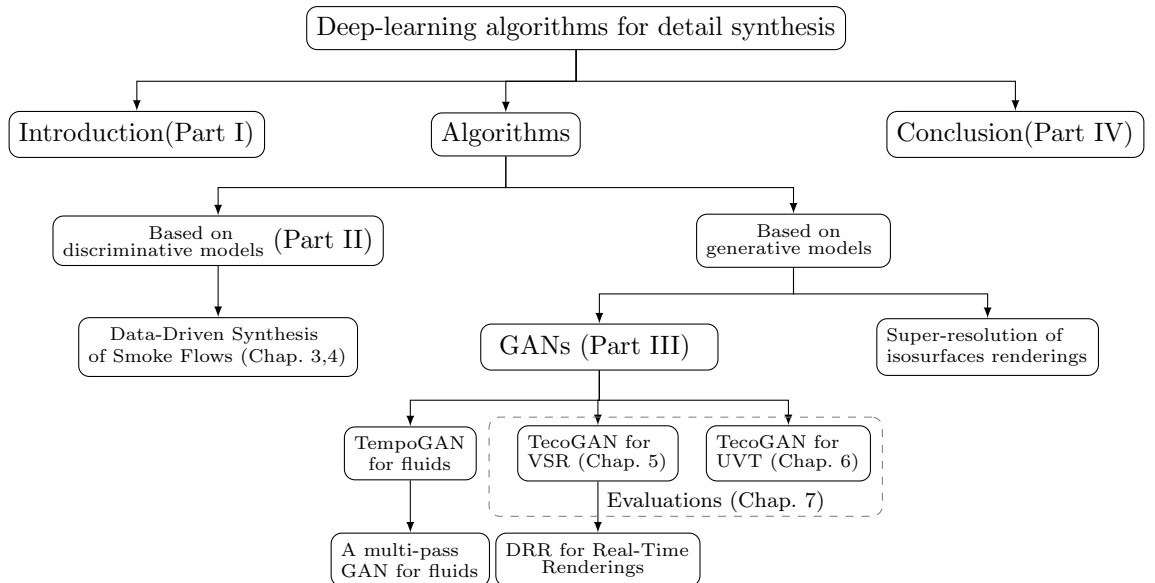
**Figure 1.2:** Remote usage of visual content has expanded to ordinary users, e.g. the game streaming service from Google Stadia [5] and the weather and climate information showing on NASA’s Eyes [6]. Detail synthesis methods help to preserve perceptual quality with reduced bandwidth for these remote applications.

tion becomes one of the key issues. Compared to the success of modeling static data distributions, people have just begun to explore the temporal relationships in sequential data-sets. As listed in Fig. 1.3, well-established data-sets [11] initiated rapid developments of deep-learning algorithms, from classification tasks to generation tasks, from static distributions to spatio-temporal distributions. Since 2018, an increasing number of work start to explore on sequential data-sets. Among them, this dissertation focuses on temporally coherent detail synthesis and employs deep-learning algorithms for efficient synthesis of various visual sequences including flow effects, natural videos and rendering sequences. The explorations on data distributions from different domains help to reveal the ability and the generality of the proposed learning methods.



**Figure 1.3:** Deep-learning-based methods [12, 13, 14, 10, 15, 16, 17, 18, 19, 20, 21, 22, 23], shown in blue points, and traditional detail synthesis methods [7, 8, 2, 24], shown in orange points. Based on established data-sets [11, 25] (shown in green), technologies develop from classification tasks to generation tasks, from specialized images (e.g. MINST) to natural images and sequential data-sets (e.g. fluids and videos).

## 1.1 Dissertation Overview



**Figure 1.4:** An overview of methods discussed in the dissertation.

In this dissertation, we introduce the motivation for detail synthesis in Part I. After that, we explore discriminative and generative modelings for fluids simulations (Part II) and natural videos (Part III) in detail. A discussion in Part IV is then provided on the connections and differences among various data domains, i.e, fluids, natural videos, and renderings. The teaser image at the beginning of this chapter shows a preview of the quality that can be achieved using methods proposed in the following parts of this dissertation.

Flow effects in computer graphics vary from basic phenomena, e.g. smoke in the air and water in a cup, to eye-catching scenes with impressive detail like tornado and tsunami in games and movies. While traditional technologies require tedious tuning on simulation and rendering with long turn-around time, we present a data-driven algorithm for fluid synthesis in Chap. 3 and 4. Using a large collection of pre-computed space-time fluid regions as a fluid repository, our method can synthesize new high-resolution volumes based on coarse input flows efficiently. Given that Convolutional Neural Networks (CNNs) are particularly popular in learning image similarity, our work bridges the gap between machine-learning and numerical simulations and learns physical descriptors for density and velocity functions of fluid simulations. Besides learning similarity for fluid in different resolutions, the descriptors are also trained to establish correspondences where the amount of numerical viscosity changes. While the spatial and temporal continuity of the synthesized flow is partially guaranteed by the physical descriptor, a deformation limiting patch advection method is beneficial for tracking deformable fluid regions robustly. In the later part of Chap. 4, we use several examples to demonstrate that our



method yields non-dissipating small scale details that are naturally integrated into the motions of the underlying flow.

In Chap. 5 and 6, we focus on video generation tasks, e.g. video super-resolution (VSR) and unpaired video translation (UVT). For natural images, GAN-based methods surpass other generative models with results following data distributions of the target domain even for multi-modal training data-sets. However, state-of-the-art VSR methods still favor simpler norm losses such as  $L^2$  over adversarial training. Since the averaging nature of these losses can easily leads to temporally smooth results with an undesirable lack of spatial detail, we propose a temporally self-supervised algorithm for GAN-based video generation tasks. With detailed analysis on the results of VSR and UVT shown in Chap. 5 and 6 respectively, we demonstrate that temporal adversarial learning is key to achieving temporally coherent solutions without sacrificing spatial detail. While spatio-temporal discriminators supervise short-term temporal coherence, a bi-directional Ping-Pong loss is used to improve the long-term temporal consistency. It effectively prevents recurrent networks from accumulating artifacts temporally without depressing detailed features. Together, the proposed temporal self-supervision leads to models that outperform previous work in terms of temporally-coherent detail.

While the visual results are important quality indicators for algorithms in computer graphics and vision, quantitative evaluations across larger numbers of samples are indispensable for identifying advantages and disadvantages of the proposed methods. With a focus on the temporal aspects of sequential data, we present temporal metrics for video evaluations in Chap. 7. When evaluating images, both pixel-wise differences and perceptual similarity are widely used. Although the proposed temporal metrics are not perfect, we believe it is the right time to consider pixel-wise differences and perceptual changes together for temporal evaluations as well. With a series of user studies conducted for VSR and UVT tasks, we confirm that these metrics closely correspond to the rankings achieved in user studies.

Besides spatial-temporal adversarial learning for videos, we train GAN models for fluid simulations and rendered sequences as well. A discussion is given in Chap. 8 on their connections and differences in order to gain an in-depth understanding of the spatial-temporal learning for visual content in general. Furthermore, we summarize contributions, open questions and future directions in the last part of this dissertation.

## 1.2 Publication List

This dissertation explains and concludes researches from the following manuscripts:

### Publications:

1. **M. Chu** and N. Thuerey, “Data-driven synthesis of smoke flows with CNN-based feature descriptors,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 69, 2017

2. Y. Xie\*, E. Franz\*, **M. Chu\***, and N. Thuerey, “tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 95, 2018
3. M. Werhahn, Y. Xie, **M. Chu**, and N. Thuerey, “A multi-pass GAN for fluid flow super-resolution,” *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 2, no. 2, Jul. 2019
4. **M. Chu\***, Y. Xie\*, J. Mayer, L. Leal-Taixé, and N. Thuerey, “Learning temporal coherence via self-supervision for GAN-based video generation,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, p. 75, 2020
5. S. Weiss, **M. Chu**, N. Thuerey, and R. Westermann, “Volumetric Isosurface Rendering with Deep Learning-Based Super-Resolution,” *IEEE Transactions on Visualization and Computer Graphics*, 2019

**Pre-prints:**

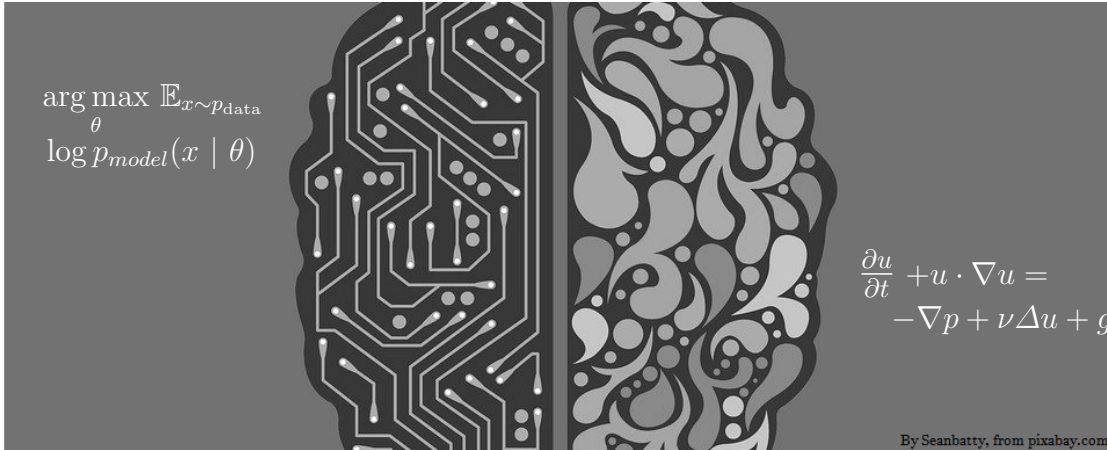
1. N. Thuerey, Y. Xie, **M. Chu**, S. Wiewel, and L. Prantl, “Physics-Based Deep Learning for Fluid Flow,” *NeurIPS Workshop, Modeling the Physical World*, 2018

**Patents:**

1. N. Kang, **M. Chu**, N. Thuerey, H. E. Lee, and D. Sagong, *Method and apparatus for modeling smoke turbulence based on patch*, 2017

---

\*: These authors contributed equally to the paper



“What I cannot create, I do not understand.”

— Richard Feynman

## 2 Fundamentals

Presenting complex and believable details relies on an understanding of the missing content in the first place. Compared to the real-world data with infinite space and continuous time, synthetic and captured data have different limitations. As an example of the former case, fluid simulations usually suffer from numerical dissipation. Videos, on the other hand, are projections in image-space that is limited by the sensitivity of capture devices. In this chapter, we first outline the fundamentals and related work for fluid simulations. Given that machine learning algorithms are widely used in computer vision tasks, we then introduce the development of these methods with a focus on image and video generation tasks. In the end, we discuss the latest findings in image and video evaluation.

### 2.1 Fluid Simulations

Fluid simulations for animation purposes are typically governed by the *incompressible Navier-Stokes equations*:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla \mathbf{u} + \mathbf{g} && \text{(the momentum equation) ,} \\ \nabla \cdot \mathbf{u} &= 0 && \text{(the incompressibility condition) ,} \end{aligned} \tag{2.1}$$

where  $\mathbf{u}$ ,  $p$  and  $\mathbf{g}$  denote velocity, pressure, and acceleration due to external forces respectively. Focusing on the single-phase inviscid flow, we drop out the viscosity term and deal with the *Euler* equations in this dissertation:

$$\begin{aligned} D\mathbf{u}/Dt &= -\frac{1}{\rho}\nabla p + \mathbf{g} , \\ \nabla \cdot \mathbf{u} &= 0 . \end{aligned} \tag{2.2}$$

In Eq. 2.1 and 2.2, the first row is derived from Newton's second law while the second row preserves the volume of the flow.

Solving these partial differential equations, fluid simulations have a long history in computer graphics [26]. In general, these numerical fluids solvers can be divided into *Lagrangian* methods with the observers following an individual fluid parcel as it moves, *Eulerian* methods that observe flow field on specific locations such as a uniform grid, and *Hybrid* methods involving both.

### 2.1.1 Eulerian Methods and Numerical Dissipation

For every time step, a typical Eulerian fluid solver splits the mathematical problem into three parts, i.e. the *advection* part, the *body forces* part, and *pressure projection* part.

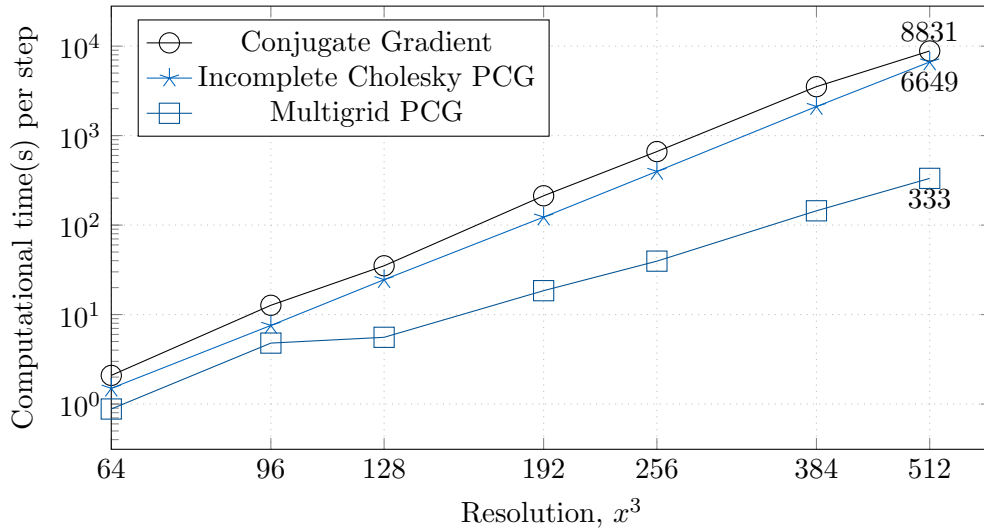
$$\begin{aligned} \frac{D\mathbf{u}}{Dt} &= 0 && \text{(advection) ,} \\ \frac{\partial \mathbf{u}}{\partial t} &= \mathbf{g} && \text{(body forces) ,} \\ \frac{\partial \mathbf{u}}{\partial t} &= -\frac{1}{\rho}\nabla p, \text{ s.t. } \nabla \cdot \mathbf{u} = 0 && \text{(pressure projection) .} \end{aligned} \tag{2.3}$$

The advection step computes the changes in the grid due to the transportation. Velocity changes caused by extra forces including gravity is considered in the second step. The pressure projection step makes the velocity field divergence-free under appropriate boundary conditions. With a divergence-free velocity field, the fluid stays mass-preserving and incompressible. Among these three typical steps, the pressure step is usually the most time-consuming one, while most of the numerical dissipation is introduced in the advection step. In the following, we outline related work that accelerate the pressure projection step and methods that reduce numerical dissipation. Our discussion is restricted to the case of single-phase flows. For an overview of fluid simulations in computer graphics, we recommend the book written by R. Bridson [27].

**Accelerating the Pressure Solve** Assuming that the intermediate velocity field before pressure solving is  $\mathbf{u}^*$ , the pressure solving step in Eq. 2.3 transforms into the Poisson equation:

$$\nabla \cdot \nabla p = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^* . \tag{2.4}$$

The problem is then equivalent to solving  $Ap = b$ , where  $A$ , sometimes referred to as the Laplacian matrix, is large, sparse and symmetric. Linear systems with this type of matrix

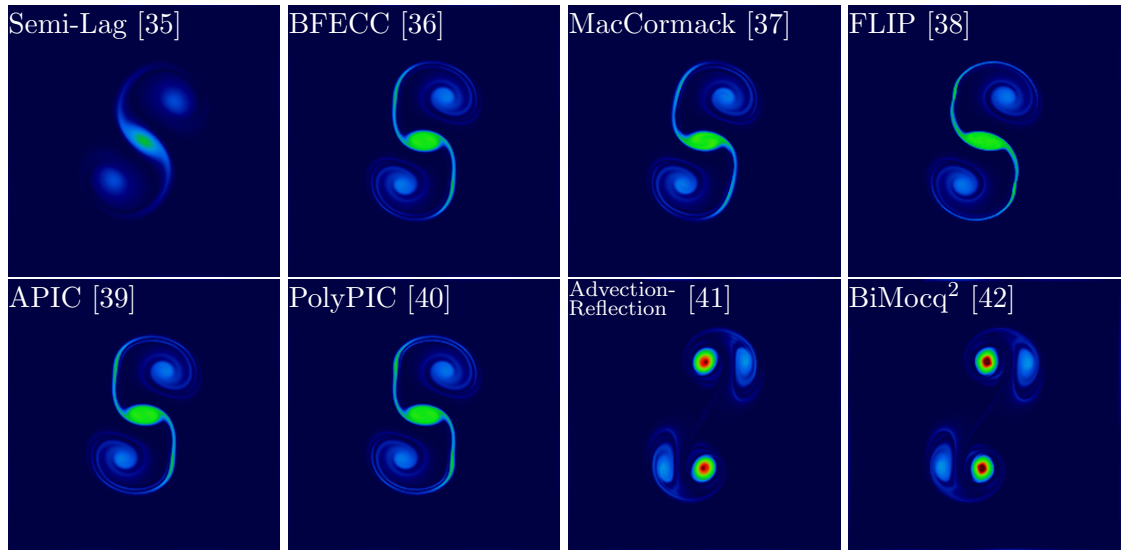


**Figure 2.1:** The computational cost of conjugated gradient solvers increases with the grid resolution. Numbers are from McAdams et al. [32] and are shown in logarithmic axes.

are usually solved using preconditioned conjugate gradient (PCG) algorithms. It is an iterative method providing approximations that converge to the exact solution. During the iteration, the approximation error is monotonically decreasing until the required tolerance is reached. However, the PCG solver scales poorly with grid resolution, as shown in Fig. 2.1. When the spatial resolution grows, the time-step length should be reduced for accuracy and the pressure solving step becomes the bottleneck for the whole simulation step.

Different methods have been proposed to speed up the necessary calculations there, e.g., coarse projections [28], lower-dimensional approximations [29], or most recently, deep-learning based approximations [30, 31, 21]. While algorithms for reducing algorithmic complexity are vital for fast solvers, the choice of data-structures is likewise important. Among others, recent work have proposed ways to solve the Poisson equation on multigrid schemes [32], highly efficient tree structures [33], and power diagram typologies [34].

**Numerical Dissipation Reduction** Solving the advection step, semi-Lagrangian integration approaches has been very popular in the atmospheric sciences community and is introduced to graphics in 1999 [35]. The name of “semi-Lagrangian” comes from the Lagrangian viewpoint that is used in a regular grid, i.e., new values at grid locations  $\mathbf{x} = i, j, k$  are taken from their backward trajectories calculated using the velocity field. This scheme is preferred over the forward Euler method because it is unconditionally stable for any choice of the time step and thus, allows for large time steps when modeling large scale flows. However, numerical dissipation arises from re-sampling with spatial interpolations and accumulates for every time step. So far, it is impossible to quantify this viscosity-like error with closed form expressions. While the viscosity of water and



**Figure 2.2:** Computer graphics has a long-history fighting against the numerical dissipation of the advection step. The comparison is presented by Qu et al. [42].

air is close to zero on human scales, the numerical dissipation always easily smooth out turbulent details and lead to unnaturally viscous motions for practical resolutions. As complex flow motion is an important component for movies and games, reducing the numerical dissipation remains a challenging problem for computer graphics.

In order to achieve natural fluid motion with non-dissipative vortices, many algorithms have been proposed to improve the advection step including the back-and-forth error compensation and correction (BFECC) method [36], the MacCormack advection method [37] which is second-order accurate and unconditionally stable, the advection-reflection scheme [41] involving a energy-preserving reflection operator, and most recently the BiMocq<sup>2</sup> method [42] that reduces re-sampling operations by saving mapping functions. Fig. 2.2 shows a visual comparison of these methods in together with some hybrid methods that will be explained below.

There are also methods that directly improving vortices to combat dissipation, e.g. algorithms that amplify existing vorticity [43, 44], research on modeling boundary layer effects [45] anisotropic vortices [46], or buoyant turbulence effects [47], and turbulence synthesis models based on numerical procedures [2, 48] or exemplar velocity fields [49].

### 2.1.2 Lagrangian and Hybrid Methods

Smoothed-particle hydrodynamics (SPH) is one of the most popular Lagrangian fluid simulation techniques. Compared to Eulerian methods, the standard SPH method [50] has the advantage to conserve linear and angular momentum but its incompressibility and stability are limited by the local computation of pressure values. Many variants are then proposed to improve these properties, e.g. the weakly-compressible SPH [51] and the implicit incompressible SPH [52] methods.

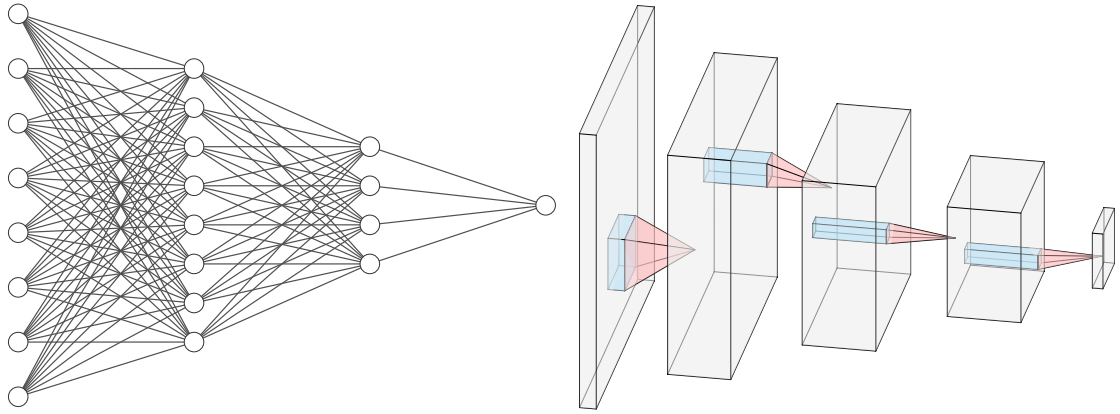
Hybrid methods have been proposed to couple particles with grid-based techniques. Usually, kinematic steps are done on particles, dynamic steps are done on the grid, and necessary information is transferred between them. These methods include Particle-In-Cell (PIC) [53], Fluid Implicit Particle (FLIP) [38], Affine PIC (APIC) [39] and material point method (MPM) [54] algorithms. With whole particle information obtained from grids, PIC is found to be stable but dissipative. FLIP preserves some information with particles and offers more turbulent motions with less stability. Transferring affine velocity information, APIC is less dissipative and stable. PolyPIC [40] further generalized APIC using polynomial representations. By saving and calculating material information on particles, e.g. deformation gradients and volumes, MPM methods [54, 55, 56] become popular for continuum material simulations including solids and fluids.

For Lagrangian and hybrid methods, previous work has likewise proposed algorithms to generate appealing results and detailed features at a moderate computational cost. Ihmsen et al. apply secondary particles to simulate spray, foam and air bubbles [57]. A surface wave simulation using surface points is proposed by Mercier et al. [58]. Ferstl et al. present a method that only uses FLIP particles within a narrow band of the liquid surface [59]. More recently, machine learning techniques are applied on fluid simulations for similar purpose, which will be discussed below.

### 2.1.3 Machine Learning Methods for Fluid Simulations

While machine learning achieves remarkable success in computer vision, there is less work so far that combines machine learning algorithms with animating fluids. As first steps in this direction, some machine-learning methods are proposed to approximate physical solutions with reduced computational cost. E.g., a regression-forest-based approach is proposed for SPH [60] and convolutional networks are trained to solve Poisson equations for Eulerian fluid simulation [30, 61]. Learning the temporal evolution is another research direction with growing interest. Among them, an LSTM-based method learns to predict pressure fields for multiple subsequent time-steps [31] and a hybrid deep learning framework with multilevel spectral decomposition is proposed for turbulent flow [62]. Generative models for fluids are likewise studied. While proper deformation can be learned by networks to transform signed distance functions in a parameter space [63], divergence-free velocity fields can also be directly generated from a set of reduced parameters [21]. Most recently, reinforcement learning [64] and differentiable fluid solvers [65, 66] are being explored for fluid control.

For detail synthesis, machine learning algorithms are proposed as well. Um et al. propose to learn liquid splashes from SPH simulation to FLIP simulation [67]. Inspired by image style transfer [16], Kim et al. transfer features from natural images to volumetric smoke densities through differentiable rendering [68]. In the dissertation, we will introduce both discriminative and generative learning methods to synthesize details for smoke simulations in Part II and Part IV. While in this section, we briefly introduced machine-learning-based methods for fluid simulations, the fundamentals about machine learning will be introduced in the next section, in together with related work on image and video generation tasks.



**Figure 2.3:** A 1D fully connected network and a 2D convolutional network.

## 2.2 Deep Learning Methods for Images and Videos

With more and more data available every day, deep learning algorithms are developed to learn representations through large data-sets using artificial neural networks. Mapping data samples from the source domain  $A$  to the target domain  $B$ , the parametric functions being learned can be summarized as:

$$f_{\theta} : A \rightarrow B \quad (2.5)$$

where  $\theta$  is the learned function parameters. We can further classify these learning tasks as discriminative and generative learning problems. E.g., image classification is usually solved with discriminative learning, where the target domain  $B$  contains  $k$  possible categories and networks learn the conditional probability of each category for a given image sample in the source domain  $R^{w \times h \times c}$ . On the other hand, generative learning is usually considered for tasks such as image generation. From a latent code in domain  $A$ , networks learn to generate an image sample so that these generated results follow the data distribution of target domain  $R^{w \times h \times c}$ . To train a discriminative model, it is usually possible to get supervised training data with input and output pairs as  $\{(a_i, b_i) | a \in A, b \in B\}$  though automatic or human labeling. Generative models on the other hand are usually trained to reproduce the training data by explicitly or implicitly maximizing the likelihood of the data-set. A common learning objective is to maximize the log-likelihood of a training data-set, which is equivalent to minimizing the KL divergence between the model and the data distributions. In this section, we first introduce the fundamentals for both discriminative and generative learning methods. We then introduce an important approach to generative model, the Generative Adversarial Network (GAN). After that, we give a review on generation and evaluation methods for images and videos.



### 2.2.1 Neural Networks and CNNs

Neural networks are constructed by layers of neurons. While the first layer contains the input and the last layer forms the output, inner layers can be described as:

$$y = \sigma(Wx + b) , \quad (2.6)$$

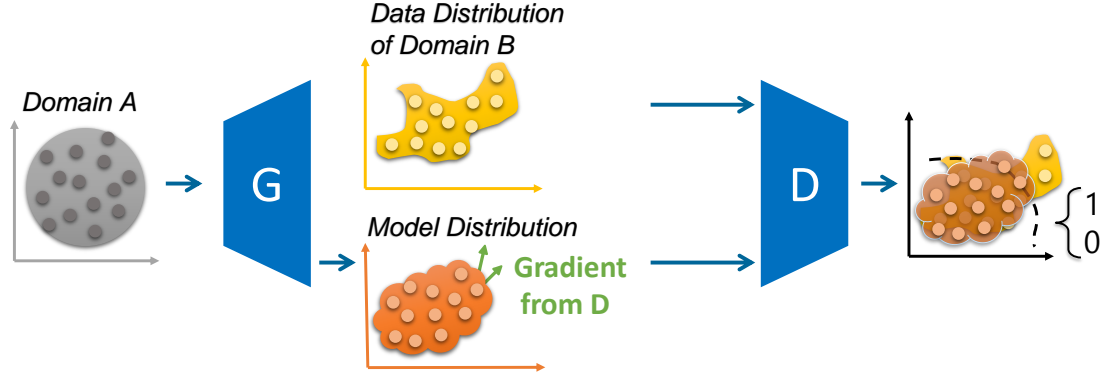
where  $x$  comes from the previous layer,  $W$  and  $b$  are trainable weights and biases, and  $y$  is the output of this layer. While  $Wx + b$  is a linear function, the non-linearity is introduced by activation functions  $\sigma$ , e.g. the sigmoid function  $\sigma_S(z) = \frac{1}{1+e^{-z}}$  and the ReLU function  $\sigma_{ReLU}(z) = \max(0, z)$ . In this way, networks are capable to compute all logical functions.

While the feed-forward pass stands for the modeling of a neural network, the weights and biases are trained by loss functions using gradient backpropagation. Loss functions measure differences between ground-truth solutions and network outputs. Backpropagation algorithms compute gradients of loss functions with respect to weights using the chain rule. While the gradient decent method directly reduces weights with their gradients weighted by a learning rate, momentum is considered and learning rates are updated dynamically in Adam and other optimization methods. Thus, they are more reliable for cases with noisy or sparse gradients. Through training, the total loss can be reduced and a local minimum can be achieved. However, since the underlying problem is usually non-convex, a global optimal solution is not guaranteed. While we present a brief introduction for the fundamentals, we refer readers to two books for details from Goodfellow et al. [69] and Nielsen [70].

For Eulerian representations including images and videos, convolutional operations are found to be useful. Important information can be extracted by their local-wise kernels, e.g. the approximation of derivatives by finite differences and a low-pass filter using Gaussian blur. By using convolutions in place of the general matrix multiplication in Eq. 2.6, Convolutional Neural Networks (CNNs) can be considered as networks with specialized weight sharing. These convolutional layers are commonly combined with pooling layers which reduce the size of a layer by applying a function, such as the maximum, over a small spatial region. Effectively, this down-samples the spatially arranged output of a layer, in order to decrease the dimensionality of the problem for the next layer [13]. Fig. 2.3 shows examples for a 1D fully conneted network and a 2D CNN. With the weight sharing, CNNs benefit from translation invariance and have a reduced risk of over-fitting, resulting in a powerful architecture that is widely used for image and video applications.

### 2.2.2 GANs

Based on the fundamentals of neural networks, we now explain one of the most important generative models, GANs [10]. As shown in Fig. 2.4, in order to reproduce the training data, GAN pose the training process as a game between two separate networks. There is a generator using input from a latent space to synthesize data samples. A discriminator is trained to classify them as either following the true data distribution or a different



**Figure 2.4:** GANs learn to reproduce the distribution of the training data-set using a game between generators and discriminators. While discriminators are trained to classify true data samples (in yellow) of Domain B from generated ones (in orange), generators are trained to map the data from Domain A to a model distribution that can fool discriminators. The gradients from discriminators (in green) can help generators to reduce the difference in distributions.

model distribution. During training, the discriminator discovers the differences between the two distributions and the generator adjusts its weights accordingly to reduce the difference. Theoretically, the training converges when two networks reach an equilibrium and the discriminator couldn't separate the two distributions.

Several loss functions have been proposed for the GAN training. In its most basic form, a GAN uses a cross entropy loss to train the classification task of the discriminator and the generator uses an opposite one as the loss function, i.e.:

$$\begin{aligned}\mathcal{L}_D &= -\mathbb{E}_{b \sim p_b(b)}[\log D(b)] - \mathbb{E}_{a \sim p_a(a)}[\log(1 - D(G(a)))] , \\ \mathcal{L}_G &= \mathbb{E}_{a \sim p_a(a)}[\log(1 - D(G(a)))] , \\ \theta_G &= \arg \max_{\theta_G} \min_{\theta_D} \mathcal{L}_D .\end{aligned}\tag{2.7}$$

This loss is interesting theoretically as it is equivalent to minimizing the Jensen-Shannon divergence between two distributions and an equilibrium can be reached when optimizing in function space. In practice, there is a saturation problem, i.e. the generator's gradient will vanish when  $\mathcal{L}_D$  is small, which makes it hard for the generator to improve. Many variants are proposed to solve this problem, e.g. a non-saturating loss

$$\begin{cases} \mathcal{L}_D &= -\mathbb{E}_{b \sim p_b(b)}[\log D(b)] - \mathbb{E}_{a \sim p_a(a)}[\log(1 - D(G(a)))] \\ \mathcal{L}_G &= -\mathbb{E}_{a \sim p_a(a)}[\log(D(G(a)))] \end{cases},\tag{2.8}$$

a least-square GAN loss [71]

$$\begin{cases} \mathcal{L}_D &= \mathbb{E}_{b \sim p_b(b)}[D(b) - 1]^2 + \mathbb{E}_{a \sim p_a(a)}[D(G(a))]^2 \\ \mathcal{L}_G &= \mathbb{E}_{a \sim p_a(a)}[D(G(a)) - 1]^2 \end{cases},\tag{2.9}$$

a Wasserstein GAN loss with gradient-penalty [72]

$$\begin{cases} \mathcal{L}_D &= -\mathbb{E}_{b \sim p_b(b)}[D(b)] + \mathbb{E}_{a \sim p_a(a)}[D(G(a))] + \lambda \mathbb{E}_{\hat{b} \sim p_{\hat{b}}(\hat{b})}[(\nabla D(\hat{b}) - 1)^2] \\ \mathcal{L}_G &= -\mathbb{E}_{a \sim p_a(a)}[D(G(a))] \\ \hat{b} &= \epsilon b + (1 - \epsilon)G(a) \end{cases}, \quad (2.10)$$

and a relativistic average GAN loss [73]

$$\begin{cases} \mathcal{L}_D &= -\mathbb{E}_{b \sim p_b(b)}[\log R(b)] - \mathbb{E}_{a \sim p_a(a)}[\log(1 - R(G(a)))] \\ \mathcal{L}_G &= -\mathbb{E}_{a \sim p_a(a)}[\log(R(G(a)))] - \mathbb{E}_{b \sim p_b(b)}[\log(1 - R(b))] \\ R(b) &= \sigma(D(b) - \mathbb{E}_{a \sim p_a(a)}D(G(a))) \\ R(a) &= \sigma(D(G(a)) - \mathbb{E}_{b \sim p_b(b)}) \end{cases}. \quad (2.11)$$

Note that the vanilla and non-saturating GANs apply the Sigmoid activation on output neuron of the discriminator, while there should be no activation for the least-square, Wasserstein, and relativistic GANs. So far, the GAN training still depends heavily on the training data-sets. Thus, there is no clear conclusion if there is a best loss across all data domains.

Despite all the different variants, GAN shows state-of-the-art performance on generating data samples in multi-modal domains. The problem of multi-modality is very common but difficult for generative models. When using normal losses, data samples in different modalities may offer conflicted gradients and results in an “averaged” distribution that is largely different to the training data. GAN achieves better performance because discriminators are trained to identify the distribution difference. For detailed explanations, we recommend the tutorial of GANs [74].

At the same time, new difficulties arise with the adversarial training of two networks. First of all, it is very hard to get a balanced training that converges with simultaneous gradient descent. As an adversarial game, the reproducibility is not as good as supervised training. Finally, mode collapse occurs sometimes. Facing these difficulties, advanced training strategies are developed along with the aforementioned improvements on loss functions. ProGAN [75] achieves stable training by increasing the spatial resolution progressively. styleGAN [76] benefits from the separation of the latent input into multiple spatial scales. SAGAN [77] uses self-attention layers in together with spectral normalization, while BigGAN [19] suggest using orthogonal regularization.

While generative model is a wide and interesting topic, conditional generation is particularly important for computer graphics, because it allows for user controls and interactions by using a conditional input. In the following, we introduce the development on conditional generation for images and videos.

### 2.2.3 Conditional Generation for Images and Videos

Natural images and videos represent very complex data distributions. They contain large amount of details with a wide span of diversity. Being able to understand and

recreate complex data distributions, deep learning show advantages with the help of large data-sets.

For conditional image generation tasks, deep learning has made great progress. While regular losses such as  $L^2$  [78, 79] offer good performance for image super-resolution (SR) tasks in terms of mean square errors, adversarial training can significantly improve the perceptual quality in multi-modal settings such as image colorization [80], super-resolution [81], and translation [17, 18] tasks.

Sequential generation tasks additionally require the generation of realistic content to change naturally over time [23, 68]. It is especially important for conditional video generation tasks [82, 83, 84, 85], where specific correlations between the input and the generated spatio-temporal evolution are required when ground-truth motions are not provided. Hence, motion estimation [86, 87] and compensation become crucial. The compensation can take various forms, e.g. explicitly using variants of optical flow networks [88, 89, 90] and implicitly using deformable convolution layers [91, 92] or dynamic up-sampling [93].

Based on the motion compensation, different architectures are used for conditional video generation. Recent work on VSR either uses multiple low-resolution (LR) frames as inputs [93, 94, 95] or use previously estimated outputs recurrently [90] to improve the spatial detail and temporal coherence in the results. In general, adversarial learning is less explored for VSR due to the temporal coherence requirement, even though it is a multi-modal problem. In video translation tasks, GANs are more commonly used and discriminators are used to supervise the spatial content. E.g., Zhu et al. [17] focuses on images without temporal constrains and generators can fail to learn the temporal cycle-consistency for videos. In order to learn temporal dynamics, RecycleGAN [22] proposes to use a prediction network in addition to a generator, while MocycleGAN [96] chose to learn motion translation in addition to the spatial content translation.

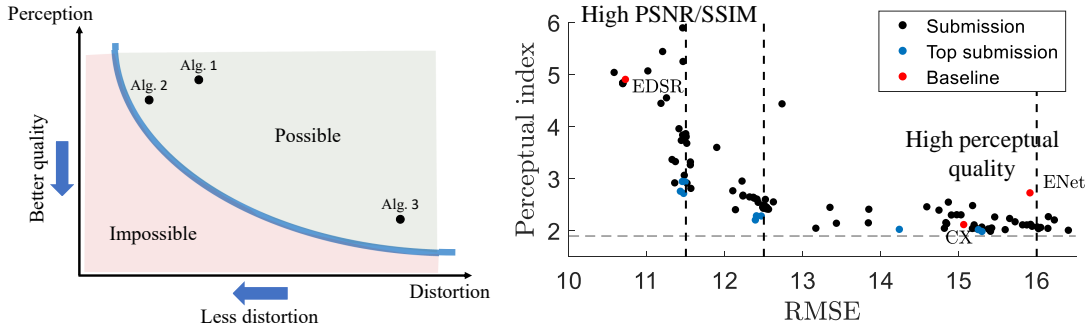
Regarding temporal constrains for videos,  $L^2$  temporal losses based on warping are generally used in video style transfer [97, 98] and UVT [99] work. However, it leads to an undesirable smooth over spatial detail and temporal changes in outputs. The vid2vid [20] method proposes adversarial temporal losses to achieve time consistency for paired video translation tasks.

The fast development of image and video generation methods leads to a high demand on advanced evaluation methods. Below, we summarize evaluation methods for them.

#### 2.2.4 Image and Video Evaluations

Quality assessment is an important part of research. Regarding images, classic pixel-wise measurements, e.g. Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) have been used for decades. In practice, these shallow functions are insufficient for assessing structured data such as images and their assessment usually differs from human judgments. A well-known case is the blurring operation which is considered as a small distortion by these metrics, but can cause large perceptual errors for humans. Thus, lots of metrics have been proposed with a perceptual motivation e.g. SSIM [100] and MSSIM [101]. Besides metrics calculated with regard to references, no-reference

metrics are also proposed, e.g. NIQE [102]. Recently, machine learning techniques are applied to learn metrics from user studies, e.g. a non-reference metric from Ma et al. [103] and the LPIPS metric using references [104]. With the help of large user study data-sets, machine-learning-based metrics managed to match human perceptions closely.



**Figure 2.5:** For image restoration, there is an inherent trade-off between the perception quality and the pixel-wise distortion [105]. The theory (conceptually illustrated on the left) agrees with the performance measured from all submissions of the 2018 PIRM Challenge [106] (shown on the right).

Studying across classic shallow metrics and perceptual metrics, researchers find that it is impossible for algorithms to reduce both pixel-wise differences and the perceptual dissimilarity. As shown in Fig. 2.5, this observation is named as the perceptual-distortion trade-off [105]. Methods with high perceptual quality usually have larger pixel-wise errors and stay at the bottom right of the figure. The top left part of the figure then contains methods with low pixel-wise errors and high perceptual errors. Unfortunately, the bottom left region is empty due to the impossibility of achieving both goals simultaneously. Thus, it is important to evaluate image generation methods using the perception-distortion plane and models stay close to the perception-distortion bound are considered as dominating others. This theory agrees with the result of the 2018 PIRM Image SR Challenge [106].

While per-frame quality of videos can be measured using image metrics above, natural temporal evolution should be considered for video assessments. Similar to the spatial shallow metrics such as PSNR, the  $L^2$  temporal metric based on warping represents a sub-optimal way to quantify temporal coherence. The situation is worse when the ground-truth motion is not available. Similar to the Fréchet Inception Distance (FID) measuring the perceptual difference of two distributions [107], Fréchet Video Distance (FVD) is proposed to measure the model distribution of a video generation method [108]. However, this metric considers the whole model distribution and perceptual metrics that evaluate natural temporal changes with regard to specific ground-truth sequences are unavailable up to now. It is worth mentioning that in addition to images and videos, researchers have explored user studies and perceptual metrics to evaluate physical problems including fluid simulations as well [67, 109, 110].

In this chapter, we have introduced the fundamentals for fluids and neural networks. Related work is included for fluid simulations and conditional generations of images and

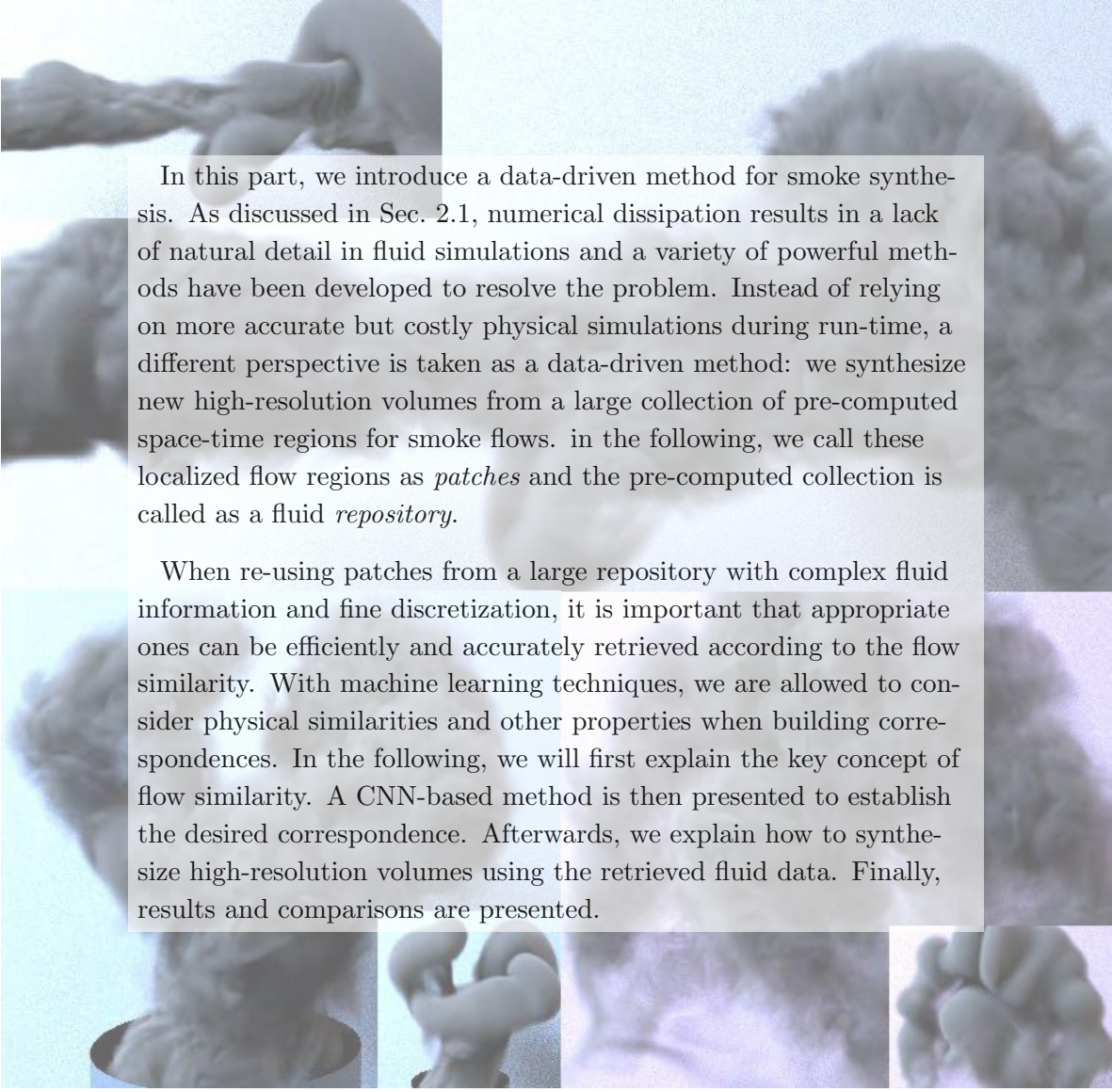
videos. We also looked into evaluation methods for images and videos. Based on the background knowledge above, we propose methods to synthesize coherent details for sequential data-sets including fluids and videos, in the rest of this dissertation.

## **Part II**

# **Detail Synthesis for Fluids**





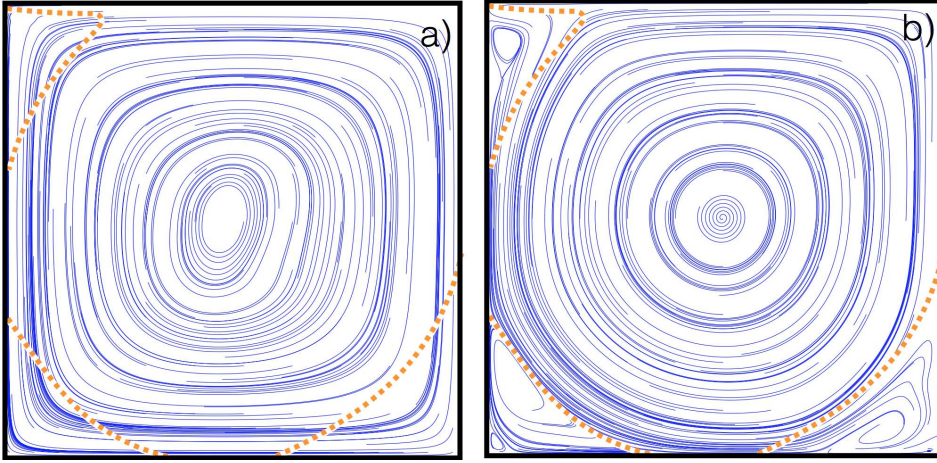


In this part, we introduce a data-driven method for smoke synthesis. As discussed in Sec. 2.1, numerical dissipation results in a lack of natural detail in fluid simulations and a variety of powerful methods have been developed to resolve the problem. Instead of relying on more accurate but costly physical simulations during run-time, a different perspective is taken as a data-driven method: we synthesize new high-resolution volumes from a large collection of pre-computed space-time regions for smoke flows. In the following, we call these localized flow regions as *patches* and the pre-computed collection is called as a fluid *repository*.

When re-using patches from a large repository with complex fluid information and fine discretization, it is important that appropriate ones can be efficiently and accurately retrieved according to the flow similarity. With machine learning techniques, we are allowed to consider physical similarities and other properties when building correspondences. In the following, we will first explain the key concept of flow similarity. A CNN-based method is then presented to establish the desired correspondence. Afterwards, we explain how to synthesize high-resolution volumes using the retrieved fluid data. Finally, results and comparisons are presented.



### 3 Learning Flow Similarity



**Figure 3.1:** Stream lines are shown for a lid driven cavity simulation without (left) and with viscosity (right). The orange lines indicate the correct position of the center vortex [111]. In this case, the graphics approach commonly used to leave out viscosity leads to a very different vortex shape.

In general, we assume that flow simulations of the same physical problem should be similar to each other in regardless of their simulation methods and discretization levels. Vice versa, flow simulations for different phenomena are then considered as dissimilar. Specifically, given a spatial region  $\Omega$  and two numerical representations of flow effects in it, our goal is to compute a dissimilarity score  $s$  for the two representations. Considering two simulations, one being a coarse approximation and the other one being a more accurate version, e.g. based on a finer spatial discretization, the score  $s$  tells us how likely that the phenomena they describe are different. We use functions  $F_c$  and  $F_f$  for the coarse and fine flow, respectively, where  $F$  could be a scalar value such as smoke density, or alternatively could also include the velocity, i.e.,  $F \in \mathbb{R}^3 \rightarrow \mathbb{R}^4$ . We will revisit which properties to include in  $F$  in Sec. 3.5, but for now we can assume without loss of generality that  $F$  is a scalar function.

In order to compute similarity, we need to extract enough information from a region of the flow such that  $s$  can infer similarity from it. We will sample the flow functions in a regular grid within  $\Omega$ , assuming that  $F$  is sufficiently smooth to be represented by point samples. All point samples from this grid are then combined into an input vector  $\mathbf{x}_c$  and  $\mathbf{x}_f$  for coarse and fine simulations, respectively. Given these inputs, we aim for computing  $s(\mathbf{x}_c, \mathbf{x}_f)$  for  $\Omega$  such that  $s$  approaches zero if the pair is actually one that

corresponds to the same phenomenon being represented on the coarse and fine scales. For increasing dissimilarity of the flows,  $s$  should increase. This dissimilarity can, e.g., result from considering different regions  $\Omega$  in the fine and coarse simulations, or when the two are offset in time.

A first guess would be to use an  $L^2$  distance to compute  $s$  as  $\int_{\Omega} \|\mathbf{x}_f - \mathbf{x}_c\|^2 d\mathbf{x}$ . This turns out to be a sub-optimal choice, as even small translations can quickly lead to undesirably large distance values. The presence of numerical viscosity makes the situation worse that different resolutions for  $F_c$  and  $F_f$  can quickly lead to significantly different velocity and density values even when they should represent the same fluid flow and thus should be considered as similar. Fig. 3.1 illustrates how strongly viscosity can influence the outcome of a simulation. Instead of manually trying to find heuristics or approximations of how these numerical errors might propagate and influence solutions, we transfer this task to a machine learning algorithm.

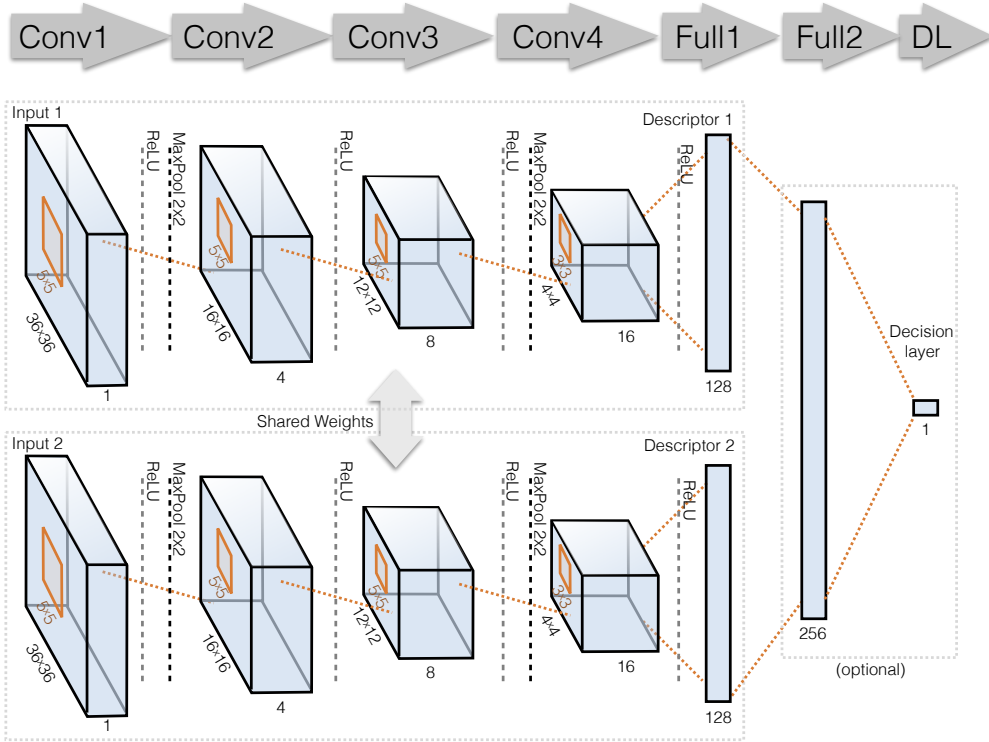
Our goal is not only to measure the distance between two inputs, but rather, given a new coarse input, we want to find the best match from a large collection of pre-computed data sets. Within the repository, the best match should give us a fine representation of the phenomena that is most similar to the coarse input. Thus, we propose to use neural networks to map the correspondence problem into a feature space. In such a space, distances computed with a simple distance metric, e.g. Euclidean distance, should correspond to the desired dissimilarity score  $s$ .

More specifically, we use CNNs to encode non-linear fluid information. They were shown to be powerful tools to predict the similarity of image pairs [112, 113]. We leverage the regressive capabilities of these CNNs to train networks that learn to encode very small, yet expressive flow descriptors,  $\mathbf{d}(\mathbf{x}) \in \mathbb{R}^m$ , with  $m$  as small as possible. These descriptors will encode correspondences in the face of numerical approximation errors and at the same time allow for very efficient retrievals of suitable space-time data-sets from a repository. Given a coarse flow region  $F_c$  we can then retrieve the best match from a set of fine regions  $F_f$  by minimizing  $\|\mathbf{d}(\mathbf{x}_f) - \mathbf{d}(\mathbf{x}_c)\|^2$ .

In the following, we will first describe the CNN architecture used for learning and explain the choice of loss functions. Details for data generation is presented after that. Finally, we provide an evaluation of fluid descriptors based on CNNs and traditional object detection methods.

### 3.1 CNN Architecture

Our neural networks consist of a typical stack of convolution layers which translate spatially arranged data into an increasing number of feature signals with lower spatial resolution. A visual summary of the network is given in Fig. 3.2. As our network compares two inputs, it initially has two branches which contain a duplicated stack of convolutional layers with shared weights. Networks with this kind of architecture are usually called as siamese neural networks [114]. Taking two different inputs at the same time, each branch acts separately on one of them and reduces its dimensionality. The outputs of the last convolutional layer of each branch (*Conv4* in our case) are fully

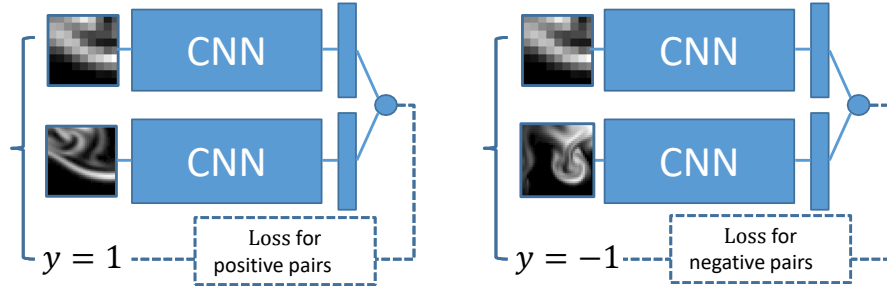


**Figure 3.2:** Our CNN architecture with two convolution stacks and shared weights, followed by a feature and an optional decision layer. Names for each layer can be found on the top of the figure.

connected into a serial vector (*Full1*) which later forms the desired feature descriptor. While descriptors are the learning targets in our project, more layers can be added if networks are required to give a single dissimilarity score as its final output. For such a case, we first concatenate the feature descriptors from the two branch together, then add another fully connected layer (*Full2*), and finally use an output layer with a single node to compute the final dissimilarity score. However, these two layers are optional for our case. When training with the hinge embedding loss, Eq. 3.3 explained below, we omit these two layers.

We will use the following abbreviations to specify the network structure: convolutional layer (CL), max pooling layer (MP), and fully connected layer (FC). We start with inputs of size  $36 \times 36$  in 2D. The input from a low-resolution simulation is linearly up-scaled to this resolution. Typically, an original resolution of  $9 \times 9$  is used for the coarse simulation. One convolutional branch of our network yields a serial layer with 128 values (*Full1*). These 128 outputs are vectorized as the final feature descriptor  $\mathbf{d}_w$  with normalization. For three dimensional inputs, we extend the spatial dimension in each layer correspondingly. Hence, the first layer has a resolution of  $36^3$  in 3D, and the spatial resolution for *Conv4* is  $2^3$  samples with 32 features. Accordingly, our feature descriptor in 3D,  $D_w$ , has dimension of 256.

### 3.2 Loss Functions



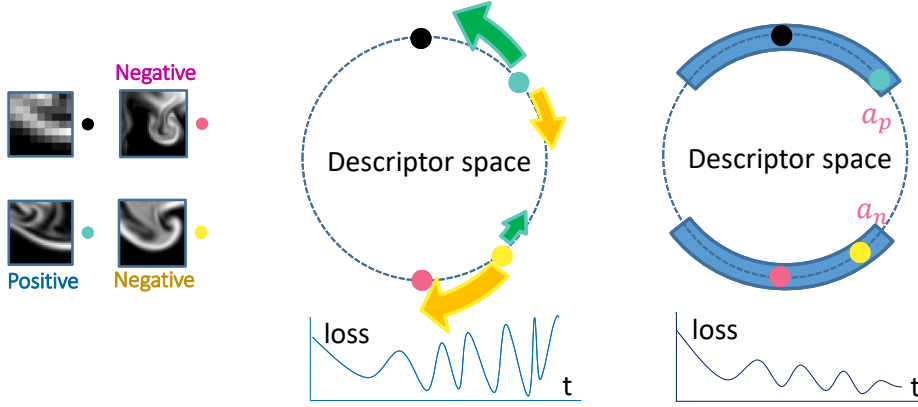
**Figure 3.3:** Both positive and negative data pairs are important to train a network for similarity.

When computing our flow similarity metric, a first learning approach could be formulated as  $s = f_s(\mathbf{x}_1, \mathbf{x}_2, \mathbf{w})$ , where  $f_s$  is the function learned by neural networks with weights  $\mathbf{w}$ ,  $\mathbf{x}_1$  and  $\mathbf{x}_2$  represent a pair of input data extracted from two simulations, and  $s$  is the output indicating the dissimilarity of the input pair. In a supervised learning, we should label our data to train the value of  $s$ . As the inputs for our regression problems stem from a chaotic process, i.e. turbulent flow, the inputs  $(\mathbf{x}_1, \mathbf{x}_2)$  look “noisy” from a regression standpoint and the training data-set  $\{(\mathbf{x}_1, \mathbf{x}_2)\}_n$  is usually not linearly separable. When labeling such a non-linear data-set, only showing similar data pairs are not enough and it is impossible to offer exact values as desired dissimilarity scores for negative data pairs. It is also crucial that the learning process is not supervised to encode some notions such as Euclidean distance of the two inputs, but learns to non-linearly map the data into a continuous latent space where physically similar data entries are close to each other and different ones are far apart. In such training runs, a proper learning objective based on positively and negatively labeled data pairs, i.e. similar and dissimilar data pairs, is crucial for establishing robust similarity between inputs.

Many loss functions have been explored for the problem of learning similarity, which usually takes an input pair with a generated label  $y$  identifying whether the pair is similar ( $y = 1$ ) or not ( $y = -1$ ), as illustrated in Fig. 3.3. While a naive  $L^2$  loss to learn exactly these labels is clearly insufficient, a slightly improved loss function could be formulated as

$$l_n(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} -f_s(\mathbf{x}_1, \mathbf{x}_2, \mathbf{w}) & , \quad y = 1 \\ +f_s(\mathbf{x}_1, \mathbf{x}_2, \mathbf{w}) & , \quad y = -1 \end{cases} \quad (3.1)$$

In this case, the network would be rewarded to give a very small dissimilarity score for positive pairs and a very large one for negative ones, but due to the lack of any limit, the learned values would diverge wildly and it is hard to get a stable training [115]. Instead, it is crucial to have a loss function that does not unnecessarily constrain the regressor, and at the same time gives it the freedom to push correctly classified pairs apart as much as necessary. The established loss function in this setting is the so called



**Figure 3.4:** Regarding a coarse input (the black point), there is a positive sample (the blue point) and several negative samples (pink and yellow points). Since some of latter (e.g. the yellow one) are similar to the positive one, the  $l_n$  loss, shown in the middle, will not be able to converge due to their conflict. The  $l_e$  loss with margins, shown on the right, is able to handle these problems.

*hinge* loss, which can be computed with:

$$l_h(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \max(0, 1 - f_s(\mathbf{x}_1, \mathbf{x}_2, \mathbf{w})) , & y = 1 \\ \max(0, 1 + f_s(\mathbf{x}_1, \mathbf{x}_2, \mathbf{w})) , & y = -1 \end{cases} \quad (3.2)$$

This loss function typically leads to significant improvements over the naive loss functions outlined above. When using a network in conjunction with the loss function of Eq. 3.2, a feature descriptor can be extracted by using the outputs of the last fully connected layer with normalization [113].

While this approach works, we will demonstrate that it is even better to embed the  $L^2$  distance of the descriptors directly into the hinge loss [112]. As we later build search data structures for our repository according to the  $L^2$  distances of their descriptors, it is important to guide the network towards encoding discriminative distances based on the feature descriptors themselves, instead of only optimizing for a final dissimilarity score  $s$ . In order to do this, we can re-formulate the learning problem to generate the descriptor itself, directly using the descriptor distance as the dissimilarity. In the following we will denote the outputs of a specific *descriptor* layer of our network with  $\mathbf{d}_w(\mathbf{x})$ , where  $\mathbf{x}$  is the input for which to compute the descriptor. Based on these descriptors, we change the regression problem to  $f_e(\mathbf{x}_1, \mathbf{x}_2, \mathbf{w}) = \beta - \alpha \|\mathbf{d}_w(\mathbf{x}_1) - \mathbf{d}_w(\mathbf{x}_2)\|$ ,  $\alpha > 0$ . Here we have introduced the parameters  $\alpha$  and  $\beta$  to fine tune the onset and steepness of the function. Using  $f_e$  to replace  $f_s$  in Eq. 3.2 yields

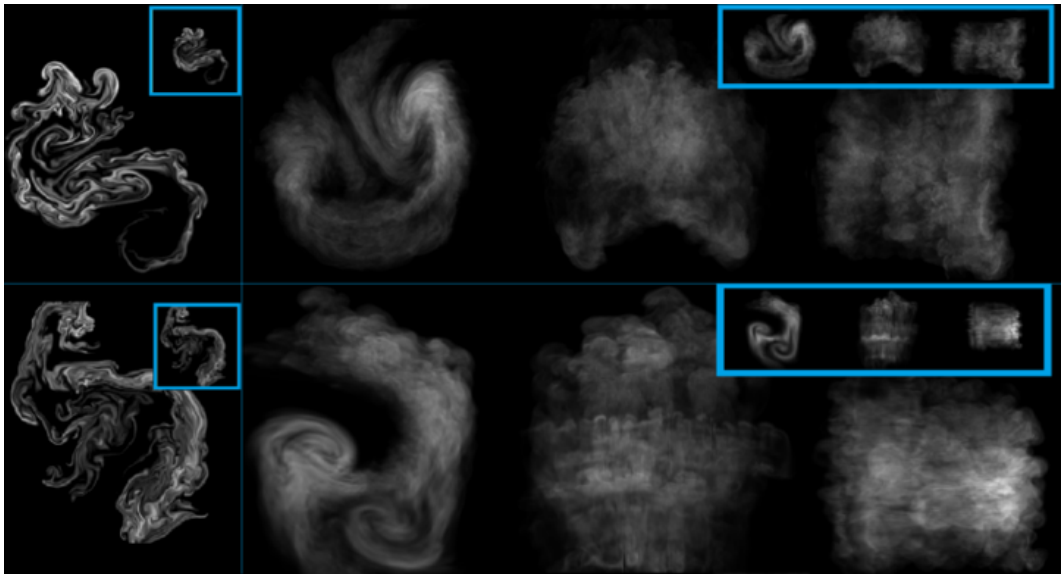
$$l_e(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \max(0, \alpha_p + \|\mathbf{d}_w(\mathbf{x}_1) - \mathbf{d}_w(\mathbf{x}_2)\|) , & y = 1 \\ \max(0, \alpha_n - \|\mathbf{d}_w(\mathbf{x}_1) - \mathbf{d}_w(\mathbf{x}_2)\|) , & y = -1 , \end{cases} \quad (3.3)$$

where  $\alpha, \beta$  parameters are replaced by  $\alpha_{p,n}$  which can be used to fine tune the margins individually for positive and negative pairs, as we will discuss below. Note that the

descriptors  $\mathbf{d}_w$  are normalized before the loss calculation. This significantly improves convergence, and supports learning distributions of components, rather than absolute values in the descriptor.

In order to offer more insight of the difference between  $l_n$  and  $l_e$ , we illustrate concept figures for the normalized descriptor space in Fig. 3.4 with some examples. Given a coarse input  $x_1$ , a positive pair can be formed with a similar input  $x_2$ , but there can be many negative pairs and some of them may have certain similarities regarding the one labeled as positive. Using the  $l_n$  loss, these similar but negative samples will conflict with the positive sample. When using the  $l_e$  loss, the margin  $\alpha_p$  and  $\alpha_n$  offer more freedom to the network and stable training runs can be achieved. We will demonstrate that the loss function  $l_e$  outperforms the other alternatives, after describing the details of the training data.

### 3.3 Data Generation



**Figure 3.5:** Examples of our data generation for training, in both 2D (left) and 3D (front, left and top views). The coarse simulation (with blue outline) is synchronized with the high resolution data in intervals  $t_r$ .

For machine learning approaches, it is crucial to have good training data sets. Specifically, the challenge in our settings is to create a controlled environment for simulations with differing discretizations and therefore different numerical viscosity. Without special care, the coarse and fine versions will deviate over time, and due to their non-linearity, initial small differences will quickly lead to completely different flows. In the following, we consider flow similarity within a chosen time horizon  $t_r$ . Note that this parameter depends on the setup under consideration, e.g., for very smooth and slow motions, larger



values are suitable, while violent and fast motions mean flows diverge faster. We will discuss the implications of choosing  $t_r$  in more detail below.

We use randomized initial conditions to create our training data. Given an initial condition, we set up two parallel simulations, one with a coarse resolution of  $r_c$  cells per axis, and we typically use a four times higher resolution  $r_f = 4r_c$  for the fine version. While it would be possible to simply run a large number of simulations for a time  $t_r$ , we found that it is preferable to instead synchronize the simulation in intervals of length  $t_r$ . Here, we give priority to the high resolution, assuming that with lower numerical viscosity, it is closer to the true solution. We thus re-initialize the coarse simulation in intervals of length  $t_r$  with a low-pass filtered version of the fine simulation.

This synchronization leads to a variety of interesting and diverse flow configurations over time, which we would otherwise have to recreate manually with different initial conditions. For our data generation, we found buoyant flows to be problematic in rectangular domains due to their rising motion. Using tall domains would of course work, but typically wastes a significant amount of space. Instead, we compute a center of mass for the smoke densities during each time step. We then add a correction vector during the semi-Lagrange advection for all quantities to relocate the center of mass to the grid center. Along with the data generation, we seed patches throughout the volume, and track them with the same algorithm we use for synthesis later on. Thus, distortion is reduced, which we describe in detail in Sec. 4.3.1. For each patch region, we record the full coarse and fine velocity and density functions within each deforming patch region for each time step. Currently, we advect the patches with the fine simulation, and use the same spatial region in the coarse simulation. An interesting improvement would be tracking another set of patches in the coarse one, and synchronize them in intervals likewise.

The recorded pairs of spatial regions for the same time step give us the set of positive pairs for training. Note that coarse and fine data in these regions may have diverged up to the duration  $t_r$ . To create negative pairs, we assign a random fine data sample to each coarse input. Two samples from a negative pair are either recorded by different patches, or recorded by the same patch, but in different time steps. Therefore, for any coarse feature example  $\mathbf{x}_1$  in our training and evaluation data-sets, there is only one fine feature example  $\mathbf{x}_2$  marked as relevant.

In this way, we have created several combined simulations in both 2D and 3D, with  $t_r = 20$  and  $t_r = 40$ , to generate training data-sets as well as evaluation data-sets. These  $t_r$  are selected so that the resulting training data have a maximum discriminative capabilities. For smaller intervals, the network presumably only sees very similar inputs, and hence cannot generate expressive descriptors. When the interval becomes too large, inputs can become too dissimilar for the network. In general,  $t_r$  is negatively correlated to the time step, kinetic energy and resolution difference. We currently select the  $t_r$  manually through comparisons.

Several images of our data generation in 2D and 3D can be found in Fig. 3.5. The detailed simulations have a 4 times higher resolution. For training, we generated 18,449 positive pairs for 2D and 16,033 pairs in total for 3D. In 2D, every training batch contains 1:1 positive and negative pairs. The latter ones are randomly generated from all positive

ones while training. While a ratio of 1:1 was sufficient in 2D, training with this ratio turned out to be slow in 3D. When the number of negative pairs is increased, we found that networks converge faster and the influence on the converged state is negligible. Thus, we use a ratio of 1:7 for 3D training runs.

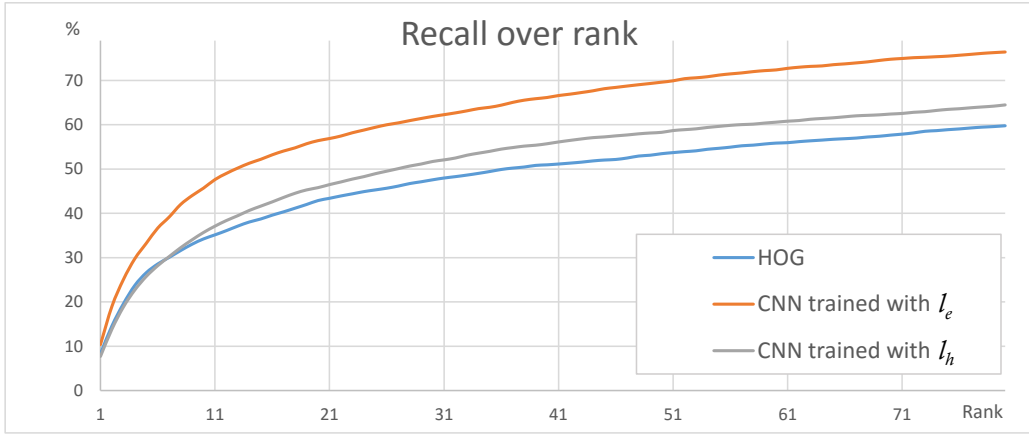
For our evaluations below, data-sets with  $t_r = 20$  and  $t_r = 40$  give consistent results. Since the latter one shows a clearer differences between the methods, in Sec. 3.4, we will focus our evaluations on the dataset with  $t_r = 40$ , which has 5440 and 5449 positive pairs in 2D and 3D respectively.

### 3.4 Evaluation

In order to evaluate and compare the performance of different approaches, it would be straight forward to compute descriptors with a chosen method for a coarse input  $i$ , and then find the best match from a large collection of fine pairs. If the best match is the one originally corresponding to  $i$ , we can count this as a success, and a failure otherwise. In this way, we can easily compute a percentage of successfully retrieved pairs. However, this metric would not represent our application setting well. Our goal is to employ the descriptors for patches in new simulations, that don't have a perfectly corresponding one in the repository. For these we want to robustly retrieve the closest available match. Thus, rather than counting the perfect matches, we want to evaluate how reliably our networks have learned to encode the similarity of the flow in the descriptor space. To quantify this reliability, we will in the following measure the true positive rate, which is typically called *recall*, over the cut-off rank  $k$ .

The recall over a cut-off rank is commonly employed in the information retrieval field to evaluate ranked retrieval results [116]. Recall stands for the percentage of correctly retrieved data sets over all given related ones. The rank in this case indicates the number of nearest neighbors that are retrieved from the repository for a given input. In particular, for our evaluation dataset with  $N$  pairs, with a given cut-off  $k$ , we evaluate the recall for all  $N$  coarse features, and thus  $kN$  pairs are retrieved in total per evaluation. In these retrieved pairs, if  $r$  pairs are correctly labeled as related, the recall at cut-off  $k$  would be  $r/N$ . In such a case, a perfect method, would yield 100% for the recall at  $k = 1$ , and then be constant for larger  $k$ . In practice, methods will slowly approach 100% for increasing  $k$ , and even the worst methods will achieve a recall of 100% for  $k = N$ . Thus, especially the first range of small  $k$  values is interesting to evaluate how reliably a method has managed to group similar data in the Euclidean space of descriptors.

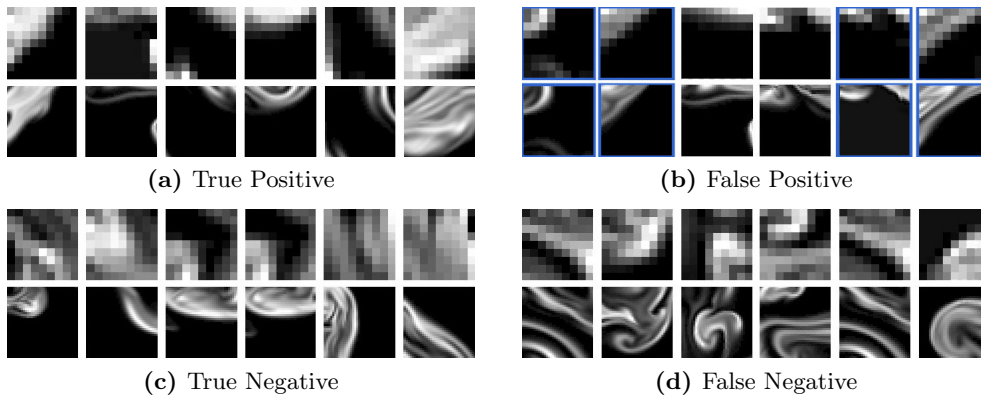
We first compare two CNN-based descriptors created with the two loss functions explained above, and the popular hand-crafted HOG descriptor in 2D. The latter is a commonly employed, and very successful feature descriptor based on histograms of gradients of a local region. As can be seen in Fig. 3.6, the HOG descriptor fares worst in this setting. Beyond a rank of 6, it's recall is clearly below that of the regular hinge loss  $l_h$  CNN descriptor. The hinge embedded loss function  $l_e$  yields the best descriptor, which in this case is consistently more than 10% better from rank 10 on. The high recall



**Figure 3.6:** Recall over rank for HOG (blue), CNN trained with  $l_h$  on similarity output node (gray), and CNN trained with  $l_e$  on descriptors directly (orange).

rates show that our CNN successfully learns to extract discriminative features, and its descriptors have a higher accuracy than conventional descriptors.

We also investigate the influence of the threshold  $\alpha_p$  and  $\alpha_n$  in the loss function  $l_e$  in Eq. 3.3. As the parametrization  $[0.0, 0.7]$  has a slightly higher accuracy among others, we will use these parameters in the following. We found that it is not necessary to have any margin on the positive side of the loss function  $l_e$ , but on the negative side, a relatively large margin gives our CNN the ability to better learn the dissimilarities of pairs.

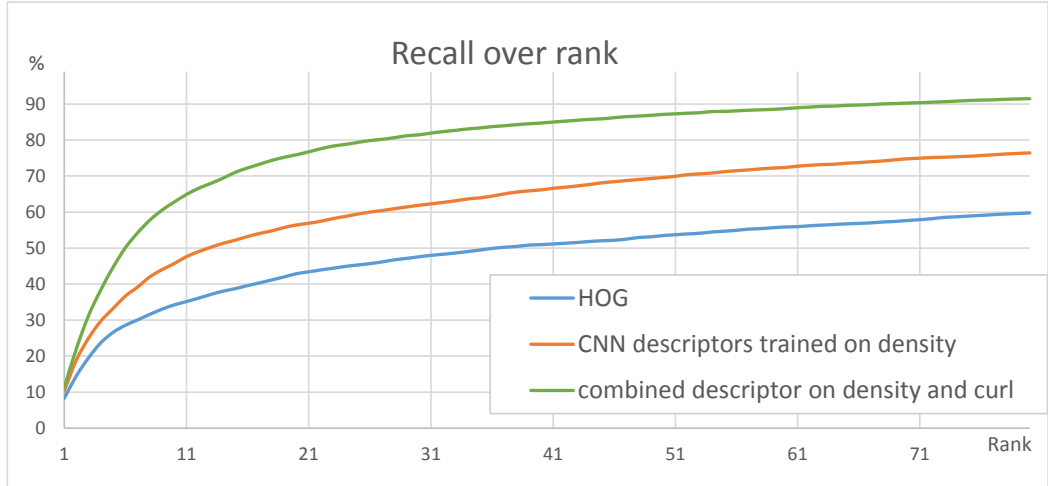


**Figure 3.7:** Top-ranking density pairs matched by our CNN.

Fig. 3.7 shows some of the top ranking true and false correspondences made by our CNN for the smoke density pairs. Correct positive and negative pairs are shown on the left. False negative pairs are related ones, for which the CNN descriptors still have a large difference, while the false positive ones are mistakenly matched pairs which were not related. In practice, the false negative pair have no effects on the synthesized results, in contrast to the false positives. However, we notice that these false positives typically

contain visually very similar data. As such, these data sets will be unlikely to introduce visual artifacts in the final volume. Some of these false positives actually stem from the same tracked patch region during data generation, and were only classified wrongly in terms of their matched time distance. These pairs are marked with blue borders in Fig. 3.7(b).

### 3.5 Descriptors for Flow Motions



**Figure 3.8:** Using curl as well as density descriptors improves matching performance.

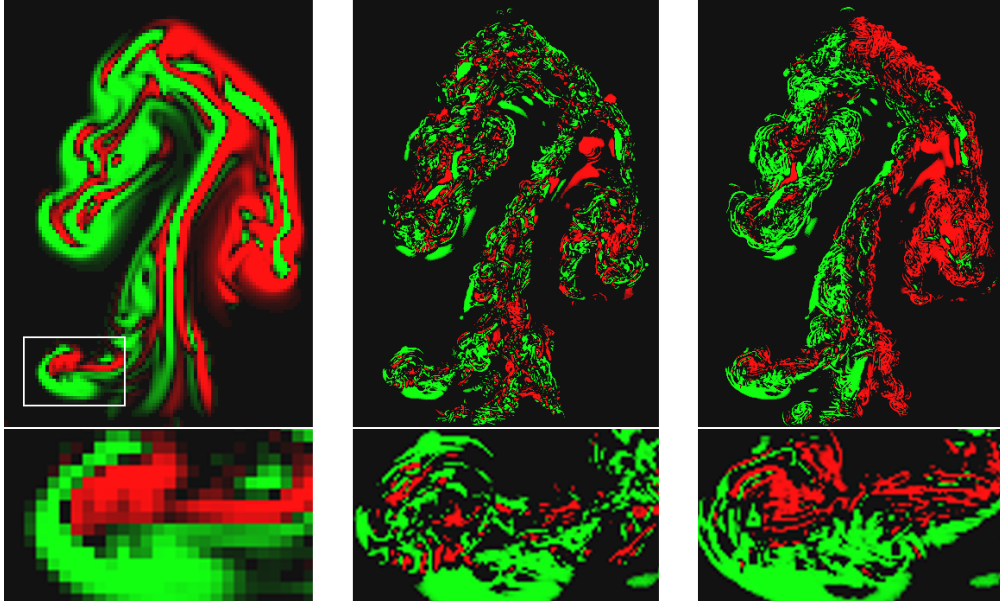
Up to now we have only considered smoke densities for establishing correspondences, however, in the fluid simulation context we also have velocity information. The velocities strongly determine the smoke motion over time and as such, they are likewise important for making correspondences between the data of a new simulation and the data-sets in the repository.

To arrive at a method that also takes the flow motion into account, we use two networks: one is trained specifically for density descriptors, while we train the second one specifically for the motion. This yields two descriptors,  $\mathbf{d}_{den}$  and  $\mathbf{d}_{mot}$ , which we concatenate into a single descriptor vector for our repository lookups with

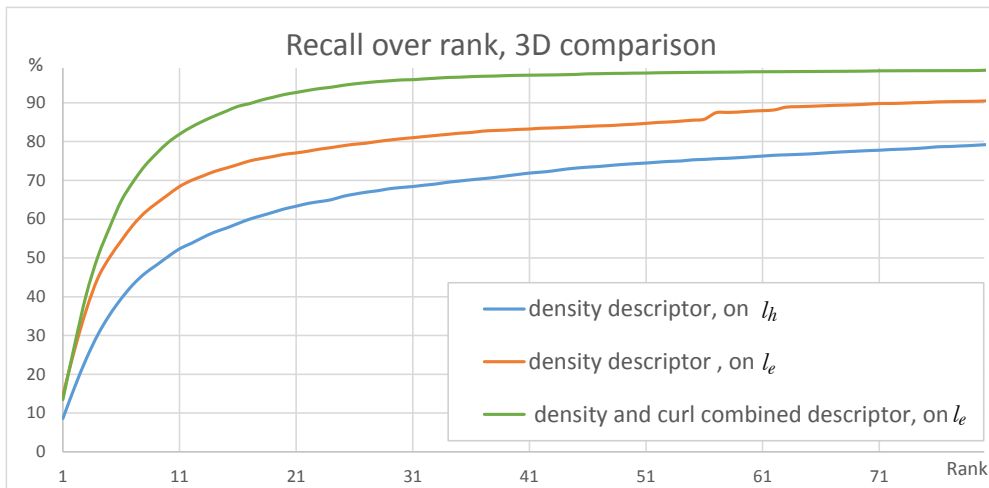
$$\mathbf{d}(\mathbf{x}) = \frac{1}{\sqrt{1 + w_m^2}} \begin{bmatrix} \mathbf{d}_{den}(\mathbf{x}) \\ w_m \mathbf{d}_{mot}(\mathbf{x}) \end{bmatrix}. \quad (3.4)$$

Note that the separate calculation of density and motion descriptor mean that we can easily re-scale the two halves to put more emphasis on one or the other. Especially when synthesizing new simulations results, we put more emphasis on the density content with  $w_m = 0.6$ .

For the motion descriptor, we use the vorticity as input, i.e.,  $\omega = \nabla \times \mathbf{u}$ . During the synthesis step, motion descriptors generated from vorticity offer significantly better



**Figure 3.9:** A visual comparison on the curl of velocity fields. From left to right, we show the input simulation, the synthesized simulation using density descriptors and the one using both density and motion descriptors.



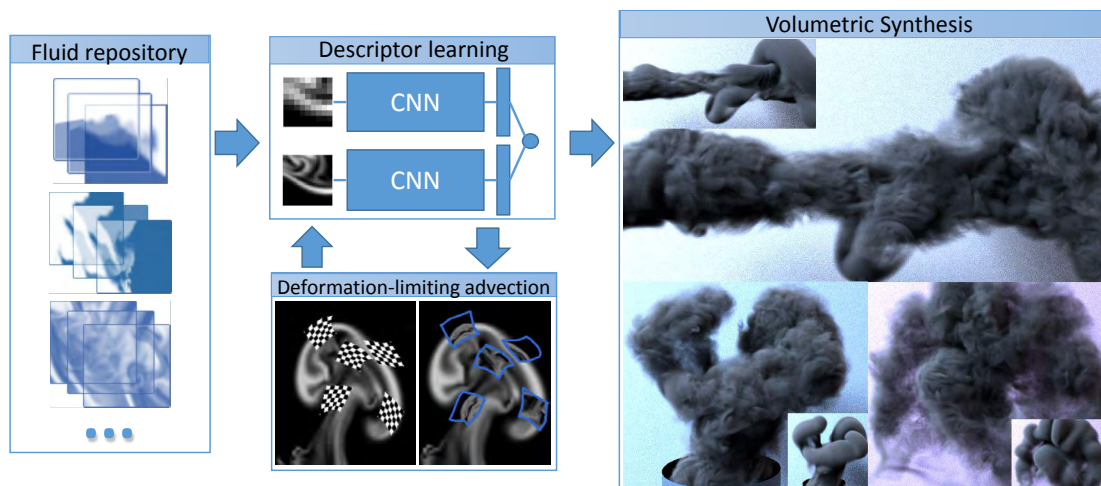
**Figure 3.10:** In 3D, using curl as well as density descriptors improves matching performance.

look-ups than the ones trained with  $\mathbf{u}$ , as the vorticity better reflects local changes in the flow field. Due to our scale separation with patch motion and content, our goal is to represent local, relative motions with our descriptors, instead of, e.g., large scale translations or stretching effects. Using a combined density and curl descriptor improves the recall rate even further. A comparison with our 2D data set is shown in Fig. 3.8, e.g., at rank 11, the recall improves by ca. 35%, and we see similar gains in three dimensions (shown in Fig. 3.10). Note that these two figures use a weight of  $w_m = 1.0$  for the curl descriptor. Based on the descriptors, a fluid synthesis method is introduced in the next chapter. Fig. 3.9 shows a visual comparison on synthesized curl fields using such

a method with and without motion descriptors. With motion descriptors, the method successfully retrieves patches that has similar motion to the input.

Due to the aforementioned improvements in matching accuracy, this approach represents our final method. In the following we will use a double network, one trained for densities and a second one for the curl to compute our descriptors. Till now, we have described our approach for learning flow descriptors with CNNs. In the next chapter, we will use these descriptors to retrieve patches and synthesize high-resolution fluid volumes.

## 4 Fluid Synthesis Based on Similarity



**Figure 4.1:** An overview of the data-driven method for smoke synthesis. The method employs a pre-computed fluid flow repository, uses CNNs to encode fluid descriptors, and synthesizes high-resolution smoke volume by matching descriptors and applying patches from the repository.

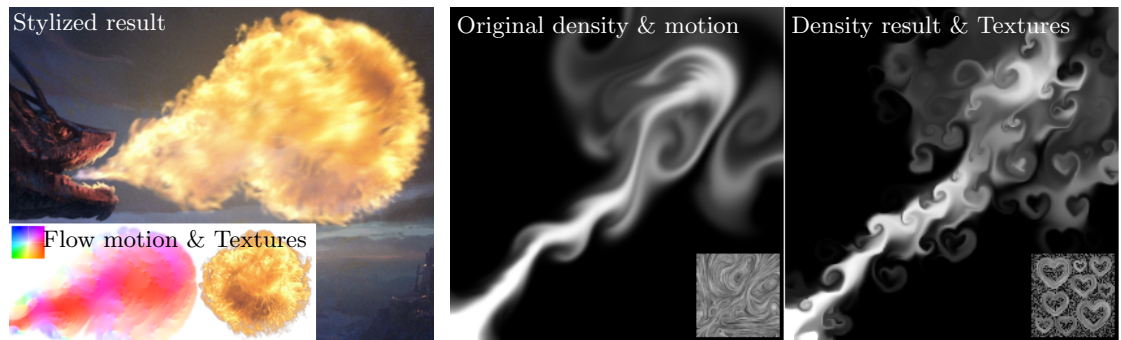
An overview of the method is given in Fig. 4.1. In the pre-computational stage, a fluid repository is prepared using high-resolution fluid simulations. It contains a large number of spatio-temporal patches with fine discretization and little numerical viscosity. When tracking and recording the raw density and velocity information of patches, a deformation-limiting advection method is used, which can be considered as a deforming Lagrangian frame. Taking the raw information as inputs, the CNN-based algorithm explained in the previous chapter is used to encode flow-aware feature descriptors. In this way, we can efficiently find the best match from this repository. In Sec. 3.5, we show that in the learned feature space, the  $L^2$  distances for patches correspond to their differences in terms of fluid density as well as flow motion. Although our method does not explicitly reduce the numerical viscosity for low-resolution simulations, the influence of discretization on the patch correspondence is automatically considered during the supervised training. In the run-time, coarse patches are sampled and advected for simulations with arbitrary resolutions. An algorithm is proposed for volumetric synthesis using matched patches from the repository.

Compared to static or rigid regions, tracking information with deforming patches has the advantage that small features stored in the repository do not inadvertently dissipate. By reusing repository patches, we side-step the strict time step restrictions

that fine spatial discretization usually impose. On the other hand, we have to make sure the regions do not become too ill-shaped over time. For this, an anticipation step is used in the patch advection scheme when synthesizing high-resolution flow volumes. Motivated by the fractal nature of turbulence and the pre-dominantly uni-directional energy transfer towards small scales in Richardson’s energy cascade [117], we match and track each patch independently. This results in a very efficient method, as it allows us to perform all patch-based computations in parallel.

In combination, the whole method make it possible to very efficiently synthesize highly detailed flow volumes with the help of a re-usable space-time flow repository. Starting from existing technologies on fluid texture synthesis, the following part will explain our methods for patch advection and volume synthesis. Then results and comparisons are presented.

## 4.1 Texture Synthesis for Fluid Simulations



**Figure 4.2:** Most of texture synthesis methods for fluid simulations focus on stylization in image space, e.g. Lazyfluids [24] on the left. The density result of a motion stylization method [118] is shown on the right.

Applying textures to fluid simulations has been an active area of research, e.g. examples in Fig. 4.2. There are methods focusing on the problem of applying textures on fluid surfaces for liquid simulations [119, 120, 121] and algorithms that synthesize two-dimensional fluid textures [8, 24]. Ma et al. [118] extend these texture synthesis techniques to flow velocities. Notably, this was the only work performing the synthesis in three dimensions back then, as the cost for synthesizing 3D volumes is typically significant even for moderate sizes. In movie productions, the *stamping* approach [122] is widely used. Instead of textures, small regions from existing simulations are reused, which usually move rigidly and therefore, do not keep aligned with new simulations.

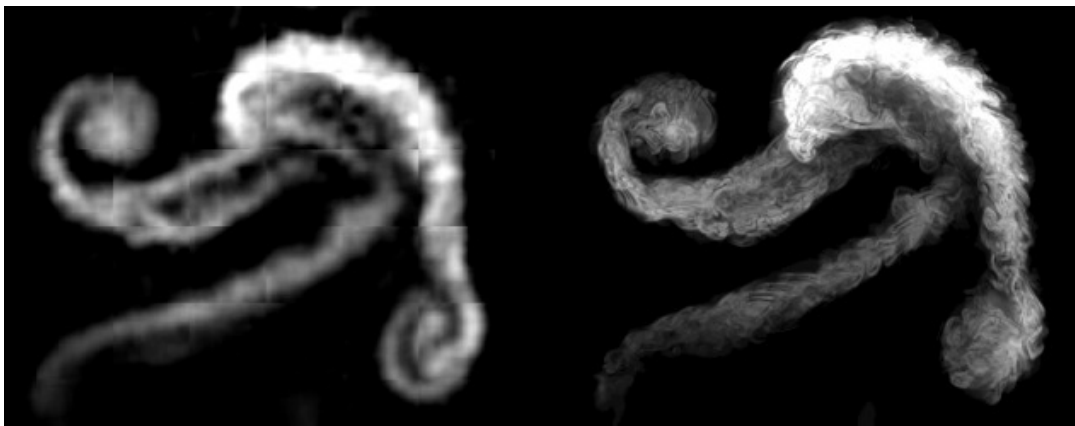
In our project, we focus on re-using high-resolution simulation data with complex fluid information. In such a case, it is important that appropriate data can be efficiently and accurately retrieved from a large repository according to the flow similarity. Meanwhile, a controllable deformation is crucial for such a Lagrangian representation with com-



plex neighborhoods. A similar idea on texture deformation was explored for animating two-dimensional flows with frequency controlled textures [123]. While their algorithm measures the amount of deformation and blends in un-deformed content, a controllable deformed state is explicitly computed for our patches to follow the flow transportation without undesirable distortion. This leads to increased life time of patches and reduces blending operations.

Before going into details of deformation-aware patch tracking, we want to show the result of a neural-network-based direct density synthesis method first. Trained on  $L^2$  losses, it has blurry results and motivated us to use the data-driven approach based on deformable patches.

## 4.2 Direct Density Synthesis



**Figure 4.3:** Directly trying to synthesize densities with CNNs yields blurred results that lack structures (left). Our algorithm computes highly detailed flows for the same input (right).

Seeing the generative capabilities of modern neural networks, we found it interesting to explore how far these networks could be pushed in the context of high-resolution flows. Instead of aiming for the calculation of a low-dimensional descriptor, the network can also be given the task to directly regress a high resolution patch of smoke densities. An established network structure for this goal is a stack of convolutional layers to reduce the input region to a smaller set of feature response functions, which then drive the generation of the output with stack of convolution-transpose layers [13].

Using  $L^2$  losses, we ran an extensive series of tests, and the best results we could achieve for a direct density synthesis are shown in Fig. 4.3. In this case, the network receives a region of  $16 \times 16$  density values, and produces outputs of four times higher resolution ( $64 \times 64$ ) with the help of two convolutional layers, a fully connected layer, and four deconvolutional layers. While we could ensure convergence of the networks without over-fitting, and relatively good temporal stability, the synthesized densities lack any detailed structures. This lack of detail arises despite the fact that this network

has a significantly larger number of weights than our descriptor network, and had more training data at its disposal.

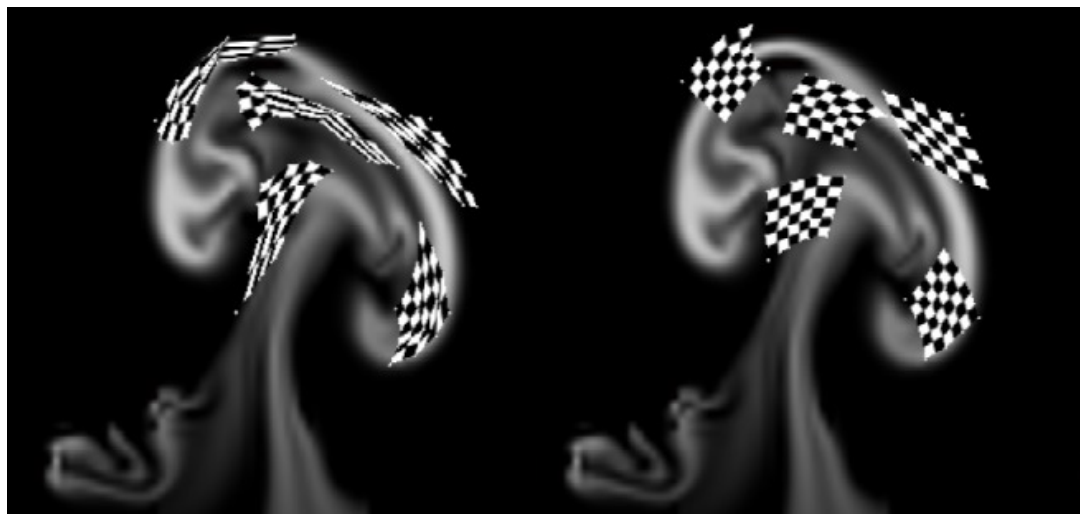
This test illustrates that the chaotic details of turbulent smoke flows represent an extremely challenging task. When trying to avoid over-fitting with a sufficiently large number of inputs, the turbulent motions seem like noise to the networks. As a consequence, the network learns an *averaged* behavior that smooths out detailed structures.

In Part III and IV, methods based on GANs are explored to solve similar problems. While the spatial adversarial learning managed to improve visual detail in the generated results, it brings temporal problems. Part III will discuss and tackle those difficult problems. In this project, we propose to combine advantages from texture synthesis and neural networks. By synthesizing detail from a pre-computed repository, we side-step the difficulties of learning and generating details directly. Compared to a generative learning task, encoding the distance between flow regions is a discriminative learning that can be well fulfilled more easily with smaller amount of weights and training data. Searching with the encoded descriptors, the best matched details in our flow repository can be supplied at render time efficiently.

### 4.3 Patch Motion and Synthesis

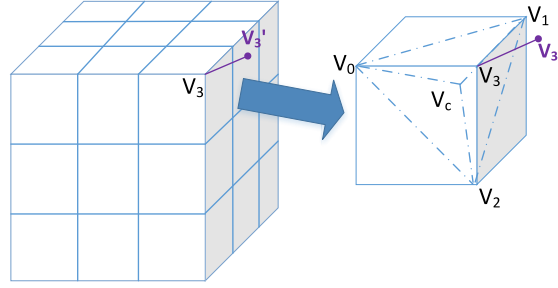
In this section, we explain methods for deformation-limiting advection of patches and volumetric synthesis of smoke volumes.

#### 4.3.1 Deformation-limiting Motion



**Figure 4.4:** Naive patch advection (left) quickly leads to distorted regions. Our deformation limiting advection (right) can keep stable regions while following the fluid motion.

Even simple flows exhibit large amounts of rotational motion that quickly lead to very undesirable stretching and inversion artifacts. An example can be seen on the left side



**Figure 4.5:** An example of patch cage with  $n = 3$ . The deformation error is accumulated from every cell, as shown on the right. The distance between a target position  $v_3$  and the advected position  $v_3'$  yields the deformation error.  $v_3$  is expressed in terms of  $v_{0,1,2}$  and their center point  $v_c$ .

of Fig. 4.4. Thus, when advecting the Lagrangian patch regions through a velocity field, we are facing the challenge to avoid overly strong distortions while making the patch follow the pre-scribed motion.

The result on the right side of Fig. 4.4 is achieved by using a local grid to track and control the motion for each patch. These grids are called as *cages* in the following, to distinguish them from the Cartesian grids of the fluid simulations, and we will denote the number of subdivision per spatial axis with  $n_{\text{cage}}$ , with a resulting cell size  $\Delta x_{\text{cage}}$ . Below, we describe our approach to limit excessive deformation of these cages.

Inspired by previous work on *as-rigid-as-possible* deformations [124, 125], we express our Lagrangian cages in terms of differential coordinates, and minimize an energy functional in the least-squares sense to retrieve a configuration that limits deformation while adhering to the flow motion.

For the differential coordinates, we span an imaginary tetrahedron between an arbitrary vertex  $\mathbf{v}_3$ , and its three neighboring vertices  $\mathbf{v}_{0,1,2}$ , as shown in Fig. 4.5. For the un-deformed state of a cell, the position of  $\mathbf{v}_3$  can be expressed with rotations of the tetrahedron edges as

$$\mathbf{v}_3 = \mathbf{v}_c + (R_{\mathbf{e}_{1,0}}\mathbf{e}_{p,2} + R_{\mathbf{e}_{2,1}}\mathbf{e}_{p,0} + R_{\mathbf{e}_{0,2}}\mathbf{e}_{p,1})/3\sqrt{2}. \quad (4.1)$$

Here  $\mathbf{e}_{i,j}$  denotes the edge between points  $i$  and  $j$ , and  $R_{\mathbf{v}}$  is the the 3x3 rotation matrix that rotates by 90 degrees around axis  $\mathbf{v}$ .  $\mathbf{v}_c$  is the geometric center of the triangle spanned the three neighbors, i.e.,  $\mathbf{v}_c = (\mathbf{v}_0 + \mathbf{v}_1 + \mathbf{v}_2)/3$ .

We can rewrite Eq. 4.1 as  $\mathbf{v}_3 = A\mathbf{v}_0 + B\mathbf{v}_1 + C\mathbf{v}_2$ , where

$$\begin{aligned} A &= I/3 + (R_{\mathbf{e}_{0,2}} + R_{\mathbf{e}_{1,0}} - 2R_{\mathbf{e}_{2,1}})/9\sqrt{2} \\ B &= I/3 + (R_{\mathbf{e}_{1,0}} + R_{\mathbf{e}_{2,1}} - 2R_{\mathbf{e}_{0,2}})/9\sqrt{2} \\ C &= I/3 + (R_{\mathbf{e}_{2,1}} + R_{\mathbf{e}_{0,2}} - 2R_{\mathbf{e}_{1,0}})/9\sqrt{2}. \end{aligned}$$

For a new position of  $\mathbf{v}'_3$ , e.g. later during a simulation run, we can measure the squared error with

$$E_{\{\mathbf{v}'_3\}} = \|\mathbf{A}\mathbf{v}'_0 + \mathbf{B}\mathbf{v}'_1 + \mathbf{C}\mathbf{v}'_2 - \mathbf{v}'_3\|^2. \quad (4.2)$$

Correspondingly, we can compute an overall deformation error for the whole cage with  $m = (n + 1)^3$  new positions  $\mathbf{v}'$  using the equation:

$$E_{defo}(\mathbf{v}') = \frac{1}{n^3} \sum_{i=0}^{n^3} \sum_{j=0}^8 E_{\{\mathbf{v}'_{ij}\}} / \left(\frac{1}{n}\right)^2 = \frac{\mathbf{v}'^T \mathbf{G} \mathbf{v}'}{n}. \quad (4.3)$$

For a whole patch cage with  $n^3$  cells, we accumulate the deformation energy for the eight corners in each cell. The energy for a single vertex is given by  $E_{\{\mathbf{v}'_{ij}\}}$  above, where  $i$  and  $j$  are the index of the cell and its corner respectively. The right side of Eq. 4.3 is a shortened notation, where  $G$  is a  $3m \times 3m$  matrix containing the accumulated contributions for all points of a cage. Since every vertex has at most 6 connected neighbors, every row vector in  $G$  has at most 19 entries, corresponding to the x, y and z positions of its neighbors, and a diagonal entry for itself. Minimizing this quadratic form directly will lead to a trivial solution of zero, so it is necessary to solve this problem with suitable constrains. In practice we want the solution to respect the advected positions. For this we add an additional advection error  $\|\mathbf{v} - \mathbf{v}'\|^2$  that pulls the vertices towards the advected positions, i.e.,  $\mathbf{v}'$ . Thus, the total energy we minimize is:

$$E(\mathbf{v}) = \lambda_0 \frac{\mathbf{v}^T \mathbf{G} \mathbf{v}}{n} + \frac{1}{m} \sum \|\mathbf{v} - \mathbf{v}'\|^2, \quad (4.4)$$

where  $\mathbf{v}'$  is the advected position,  $m = (n + 1)^3$  is the number of vertices in the grid, and  $\lambda_0$  controls the balance between advection and deformation. Note that in the original formulation,  $R_{\mathbf{v}}$ , and thus also  $G$ , are expressed in terms of  $\mathbf{v}$ , making the whole problem non-linear. Under the assumption that the advected coordinates do not deform too strongly within a single step, which we found to be a valid assumption even for the large CFL numbers used in graphics, we linearize the optimization problem by computing  $G'$  using  $\mathbf{v}'$  in Eq. 4.1. The full minimization problem is now given by

$$\frac{\partial E(\mathbf{v})}{\partial \mathbf{v}} = 0 \approx 2[(\lambda G' + I)\mathbf{v} - \mathbf{v}'], \quad \lambda = \frac{m}{n} \lambda_0, \quad (4.5)$$

where we have introduced a scaling factor  $\frac{m}{n}$ , that makes the system independent of the chosen cage resolution.

Solving the linear system of  $\mathbf{v}^* = (\lambda G' + I)^{-1} \mathbf{v}'$ , we get a good set of positions considering the deformation and fluid motion. However, we haven't take the scale of cages into account yet. Similar to Eq. 4.3, an error for the scale can be likewise calculated to keep the original length,  $\Delta x_{cage}$ , for every edge, i.e.

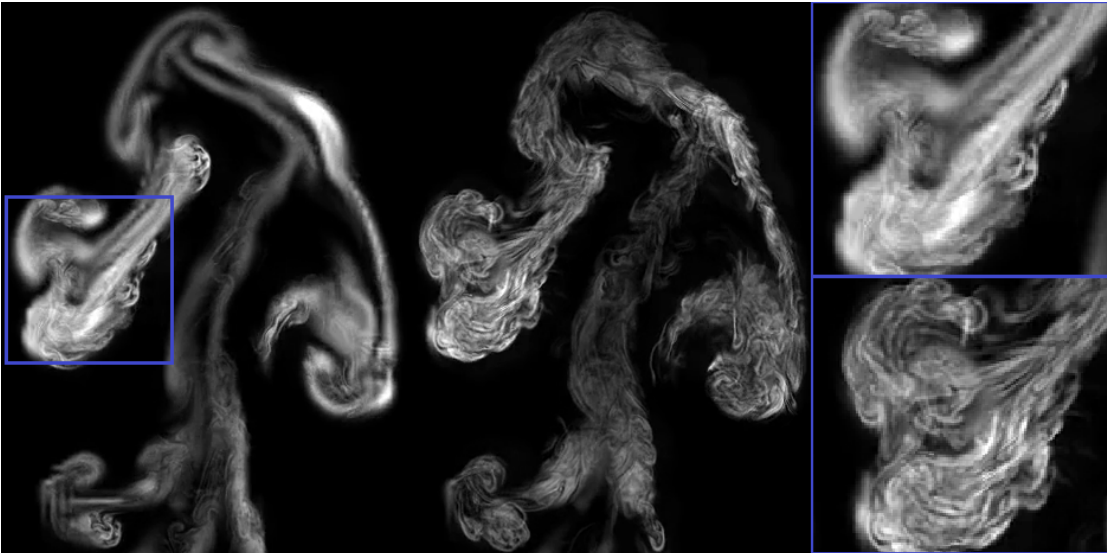
$$E_{scale}(\mathbf{v}) = \frac{1}{n^3} \sum \sum_{e_{i,j}} \left\| e_{i,j} - \frac{\Delta x_{cage} \mathbf{e}^*}{\|\mathbf{e}^*\|} \right\|^2 = \frac{1}{n^3} \sum \sum_{e_{i,j}} \left\| \mathbf{v}_j - \mathbf{v}_i - \frac{\Delta x_{cage} (\mathbf{v}_j^* - \mathbf{v}_i^*)}{\|\mathbf{v}_j^* - \mathbf{v}_i^*\|} \right\|^2. \quad (4.6)$$

In this equation,  $\mathbf{v}_j$  and  $\mathbf{v}_i$  are the unknowns and  $\mathbf{v}_j^*$  and  $\mathbf{v}_i^*$  are the solution of the last step. Minimizing Eq. 4.6, we get

$$\frac{\partial E_{scale}(\mathbf{v})}{\partial \mathbf{v}} = 0 \Rightarrow \frac{\partial \{\mathbf{v}^T H \mathbf{v} + \mathbf{f}(\mathbf{v}^*) \mathbf{v} + c\}}{\partial \mathbf{v}} = 0 \Rightarrow \mathbf{v} = -H^{-1} \mathbf{f}(\mathbf{v}^*) . \quad (4.7)$$

This time,  $H$  is a constant matrix and  $\mathbf{f}(\mathbf{v}^*)$  is calculated from  $\mathbf{v}^*$ , resulting in a very efficient scaling step. Combining Eq. 4.5 and 4.7, the final set of positions on a patch cage is achieved considering the flow advection, deformation and scaling. Note that all these calculations are relatively small. The most expensive step is the inverse of  $(\lambda G' + I)$ , which is a symmetry and sparse matrix in size of  $3m \times 3m$ . Hence, it can be solved very efficiently with a few steps of a conjugate gradient solver, independently for all patches.

As our goal is to track large scale motions with our patches, we have to respect the different spatial scales merged in the flow field. To reduce the effects of small scale perturbations in the flow, we advect the patches with a low-pass filtered version of the velocity, where the filter is chosen according to the cage cell size  $\Delta x_{cage}$ .



**Figure 4.6:** While patches are faded in normally on the left, the patch anticipation strategy is applied for the middle. For the region in blue, the right part shows a zoom-in of the normally version on top and the anticipation version on the bottom.

**Anticipation** To prevent abrupt changes of densities, we fade patches in and out over a time interval  $t_f$  for rendering (see below). For our examples, we use a  $t_f$  of 40 time steps. Unfortunately, this temporal fading means that when patches are fully visible, they are typically already strongly deformed due to the swirling flow motions. In Fig. 4.6, when patches fade in normally, the result has a lot less details because ill-shaped patches are removed. We can circumvent this problem by letting the patches *anticipate* the flow motion. I.e., for a new patch at time  $t$ , we back-track its motion and deformation to the

previous time  $t - t_f$ . This leads to completely un-deformed patch configurations exactly at the point in time when they are fully visible. The result therefore contains much more details as shown in the middle of Fig. 4.6. On the right part, the difference for the blue region is clear.

**Initialization** When seeding a new, un-deformed patch at a given position, we found that having axis-aligned cages is not the best option. Inspired by classic image feature descriptors, we initialize the orientations of our cages according to the gradient of the density field. Specifically, we calculate gradient histograms in the cage region  $\Omega$ , and use the top ranked directions as main directions. The solid angle bins of 3D gradient histograms can be defined using meridians and parallels [126]. We evenly divide azimuth  $\theta_i$  and polar angle  $\phi_j$  with step sizes  $\Delta\theta = \Delta\phi = \pi/n_b$ , resulting in  $2n_b$  and  $n_b$  subdivisions for the azimuth and the polar angle, respectively. 3D vectors are then specified as  $(r, \theta, \phi)$ , where  $r$  denotes the radius. The unit sphere is divided into a set of bins  $\{\mathbf{b}_{ij}\}, 0 \leq i < 2n_b, 0 \leq j < n_b$ , where  $\mathbf{b}_{ij} = (1, \theta_i - \Delta\theta/2, \phi_j - \Delta\phi/2)$  denotes the normalized central direction of the bin in the spherical coordinate system. Based on these bins, histograms are calculated as:

$$h_{ij} = \frac{1}{A_{ij}} \sum_{\mathbf{o} \in \Omega} w \left( \left| \frac{\phi_{\mathbf{o}} - \phi_j}{\Delta\phi} + \frac{1}{2} \right| \right) w \left( \left| \frac{\theta_{\mathbf{o}} - \theta_i}{\Delta\theta} + \frac{1}{2} \right| \right) G((\mathbf{x} - \mathbf{o})^2) r_{\mathbf{o}}, \quad (4.8)$$

where  $\mathbf{o}$  is the position of a sample point in region  $\Omega$  with density gradient  $(r_{\mathbf{o}}, \theta_{\mathbf{o}}, \phi_{\mathbf{o}})$ ,  $w(d) = \max(0, 1 - d)$ ,  $A_{ij}$  represents the solid angle of  $\mathbf{b}_{ij}$ , and  $G$  denotes a Gaussian kernel.

At the position  $\mathbf{x}$  of a new patch, we compute the gradient histogram for the smoke density  $d_s$  as outlined above. The histogram has a subdivision of  $n_b = 16$ , and the region  $\Omega$  is defined as a  $9^3$  grid around  $\mathbf{x}$ . We choose the main direction of the patch as  $\mathbf{b}_k$ , where  $k$  denotes the histogram bin with  $k = \arg \max_{i,j} (h_{ij})$ . For the secondary direction, we recompute the histogram with gradient vectors in the tangent plane. Thus, instead of  $\nabla d_s$  we use  $(\nabla d_s - (\nabla d_s \cdot \mathbf{b}_x) \mathbf{b}_x)$ . The initial orientation of the patch is then defined in terms of  $(\mathbf{b}_x, \mathbf{b}_y, \mathbf{b}_x \times \mathbf{b}_y)$ . This "gradient-aware" initialization narrows down the potential descriptor space, which simplifies the learning problem, and leads to more robust descriptors.

**Discussion** While as-rigid-as-possible deformations have widely been used for geometric processing task, our results show that they are also highly useful to track fluid regions while limiting deformation. In contrast to typical mesh deformation tasks, we do not have handles as constraints, but instead an additional penalty term that keeps the deformed configuration close to the one pre-scribed by the flow motion. The direct comparison between a regular advection, and our cage deformations is shown in Fig. 4.4. It is obvious that a direct advection is unusable in practice, while our deformation limiting successfully lets the cages follow the flow, while preventing undesirable deformations. The anticipation step above induces a certain storage overhead, but we found that it greatly reduces the overall deformation, and hence increases the quality of

synthesized densities. In practice, we found the induced memory and storage overheads to be negligible.

### 4.3.2 Synthesis and Rendering

In the following, we will outline our two-pass synthesis algorithm, as well as the steps necessary for generating the final volumes for rendering. A pseudo-code summary of the synthesis step is given in Alg. 1.

In the forward pass for flow synthesis, new patches are seeded and advected step by step, and are finally faded out. To seed new patches, a random seeding grid with spacing  $s_p/2$  is used,  $s_p = n\Delta x_{cage}$  being the size of a patch. In addition, we make use of a patch weighting grid  $w_s$ . It uses the native resolution of the simulation, and acts as a threshold to avoid new patches being seeded too closely to existed ones.  $w_s$  accumulates the spatial weights of patches, i.e., spherical kernels centered at the centroids of each patch cage with a radius of  $s_p/2$ . A linear falloff is applied for the last two-thirds of its radius, ramping from one to zero. At simulation time, we typically accumulate the patch weights in  $w_s$  without applying the patch deformations. This is in contrast to render time, where we deform the patch kernels. The high-resolution weight grid with deformed patch weights for rendering will be denoted  $w_r$  to distinguish it from the low-resolution version  $w_s$ . As  $w_s$  is only used for thresholding the creation of new particles, we found that using un-deformed kernels gives very good results with reduced runtimes.

New patches will not be introduced at a sampling position  $\mathbf{x}_n$  unless  $w_s(\mathbf{x}_n) < 0.5$ . In practice, this means the distance to the closest patch is larger than  $s_p/3$ . For each newly assigned patch, we compute its initial gradient-aligned frame of reference with Eq. 4.8, calculate the CNN inputs at this location, and let both CNNs generate the feature descriptors. Based on the descriptors, we look-up the closest matches from our repository with a pre-computed kd-tree. For successfully matched patches, our deformation limiting advection is performed over their lifetime. The maximal lifetimes are determined by the data-set lengths of matched repository patches, which are typically around 100 frames. We remove ill-suited patches, whose re-evaluated descriptor distance is too large for the current flow configuration, or whose deformation energy in Eq. 4.3 exceeds a threshold. In practice, we found a threshold of  $0.15s_p^2$  to work well for all our examples. After the forward pass, matched patches anticipate the motion of the target simulation in a backward pass. Here we move backwards through time, and advect all newly created patches backward over the course of the fade in interval. Finally, for each frame we store the coarse simulation densities, as well as patch vertex positions  $\mathbf{v}$ , together with their repository IDs and temporal fading weights. This data is all that is necessary for the rendering stage.

During synthesis the deformation limiting patch advection effectively yields the large motions which conform to the input flow, while small scale motion is automatically retrieved from the repository frame by frame when we render an image. Note that we only work with the low-dimensional feature descriptors when synthesizing a new simulation result, none of the high-resolution data is required at this stage.

---

**Flow quantities:**density grid  $d$ , velocity grid  $\mathbf{u}$ , weight grid  $w_s$ , patch cages  $\mathbf{c}_j$ , scalar weights  $w_j$ .

---

**Pre-computation:**

```

for  $t = 0$  to  $t_{end,e}$  do
  Run exemplar simulation, update  $d_e, \mathbf{u}_e$ 
  PatchAdvection( $\mathbf{u}_e$ )
  SeedNewPatches( $w_s$ )
  Compute CNN descriptors  $\mathbf{d}$  from  $d_e, \mathbf{u}_e$ 
  Save  $\mathbf{d}$ , and high-resolution  $d_e$  in patch regions to disk
  Accumulate patch spacial weights in  $w_s$ 
end

```

---

**Runtime synthesis:**

```

 $G =$  load descriptors from repository
// forward pass:
for  $t = 0$  to  $t_{end,t}$  do
  Run source simulation, update  $d_s, \mathbf{u}_s$ 
  PatchAdvection( $\mathbf{u}_s, \mathbf{c}$ )
  SeedNewPatches( $w_s$ )
  Compute CNN descriptors  $\mathbf{d}$  from  $d_s, \mathbf{u}_s$ 
  for  $j = 1$  to  $\nu_{patches}$  do
    if Patch  $j$  unassigned then
      Find closest descriptor to  $\mathbf{d}_j$  in  $G$ 
    end
    else
      Update and check quality of  $\mathbf{d}_j$ 
      LimitDeformation( $\mathbf{c}_j$ )
      Update patch fading weights  $w_j$  (fade out)
      Store  $j, \mathbf{c}_j, w_j$  for frame  $t$ 
    end
  end
  Store patches at time  $t, d_s, \mathbf{u}_s$ 
  Accumulate patch spacial weights in  $w_s$ 
end
// backward pass:
for  $t = t_{end,t}$  to  $0$  do
  Load  $d_s, \mathbf{u}_s, patches$  at time  $t$ 
  for  $j = 1$  to  $\nu_{patches}$  do
    if Anticipation active for Patch  $j$  then
      PatchAdvection( $-\mathbf{u}_s, \mathbf{c}$ )
      Store  $j, \mathbf{c}_j, w_j$  for frame  $t$ 
      Update  $w_j$  (fade in)
    end
  end
end

```

---

**Algorithm 1:** Pseudo-code for our algorithm.



**Table 4.1:** Details of our animation setups and repository data generation.

Scenes, Fig.	Base res.	Base patch res.	Avg. patches	Time	
PlumeX, Fig. 4.7	$108 \times 60 \times 60$	$15^3$	388	5.3s	
Obstacle, Fig. 4.10	$76 \times 64 \times 64$	$12^3$	362	3.9s	
Jets, Fig. 4.11	$90 \times 60 \times 60$	$12^3$	486	4.0s	
Comp. $L^2$ , Fig. 4.9	$50 \times 80 \times 50$	$9^3$	682	2.23s	
Comp. Wlt, Fig. 4.12	$50 \times 80 \times 50$	$9^3$	647	2.42s	
Repo.	Res. $440^3$	Patch res. $72^3$	Patch no. 14894	Patch storage 5.1GB densities, 30 MB descriptors	

At render time we synthesize the final high-resolution volume. To prepare this volume, we also need to consider spatial transitions between the patches amongst each other, and transitions from the patch data to the original simulation. For this, we again accumulate the deformed spherical patch kernels into a new grid  $w_r$  with the final rendering resolution, to spatially weight the density data.

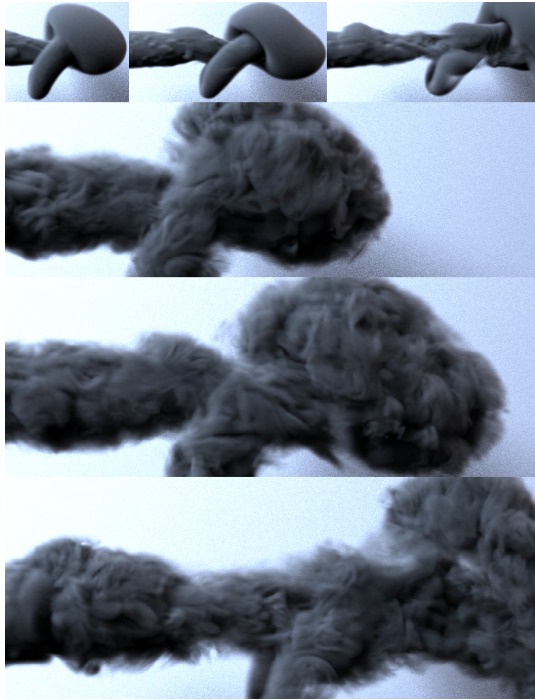
Then, the high-resolution density contained in the matched patches is weighted with the spherical kernel and accumulated together to render the high-resolution volume. As our repository contains densities that are normalized to  $[0..1]$  and our descriptor is invariant to scaling and offset transformations, we map the repository content to the min-max range of the densities in the target region. To blend the contributions of overlapping patches, we normalize the accumulated high resolution density by  $w_r$ . Additionally, we use a blurred version of the original coarse densities as a mask. We noticed that patches can sometimes drift outside of the main volume while moving with the flow. The density mask effectively prevents patches from contributing densities away from the original volume.

## 4.4 Results and Discussion

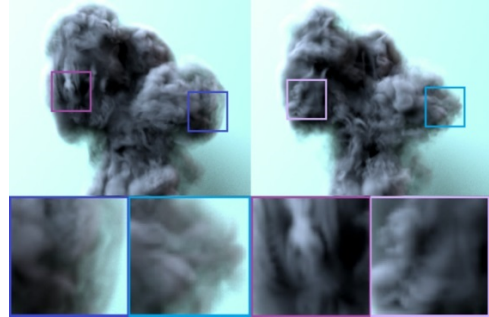
We will now demonstrate that our approach can efficiently synthesize high-resolution volumes for a variety of different flows. Typically, we run a single time step per frame. All run-times in the following are given as the average per frame.

As a first example, we have simulated a simple rising plume example, shown in Fig. 4.7. Note that gravity acts along the x-axis in our setup. In this case, the resolution of the original simulation was  $108 \times 60 \times 60$ , and on average, 388 patches were active over the course of the simulation. Our approach is able to synthesize a large amount of clearly-defined detail to the input flow that does not dissipate over time. Details of this simulation setup, as well as for the other examples, can be found in Table 4.1.

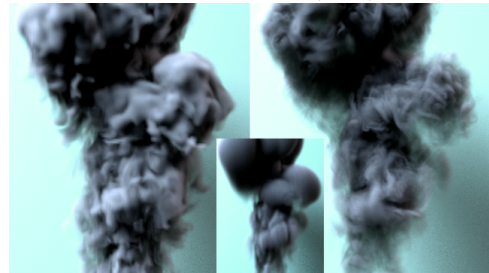
A second example is shown in Fig. 4.10. Here we add an additional obstacle, which diverts the flow. For the patch motion, we simply extend flow velocities into the obstacles, and add a slight correction along the gradient of the obstacle’s signed distance function if a patch vertex inadvertently ends up inside the obstacle. Our deformation limiting advection smoothly guides the patches around the obstacle region.



**Figure 4.7:** A horizontal plume simulation. The top three images show the input, the bottom three the high-resolution densities generated with our method. On average, 388 patches were active per timestep.



**Figure 4.8:** Descriptor comparisons. Our algorithm with simple descriptors (left) results in overly regular structures (blue) and sub-optimal matches (purple). Chances for such cases are reduced with the CNN-based descriptors (right).



**Figure 4.9:** More comparisons. On an input with  $50 \times 80 \times 50$  cells (center), the wavelet turbulence method (left) takes 2.75s/frame for a volume with  $150 \times 240 \times 150$  cells. Our method (right) yields effective resolutions of  $400 \times 640 \times 400$  with 2.23s/frame.

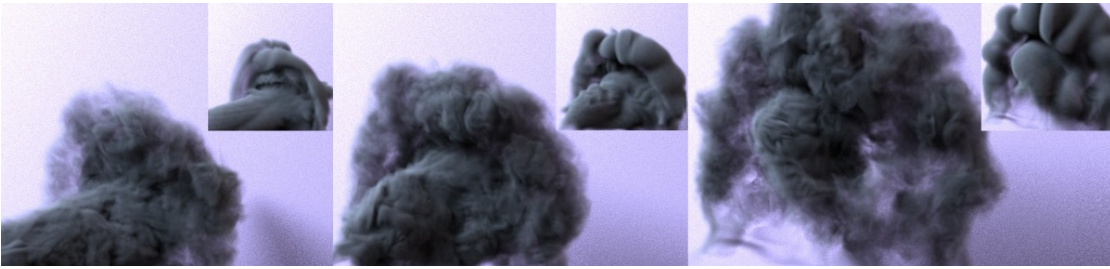
A different flow configuration with colliding jets of smoke is shown in Fig. 4.11. For this setup, on average 486 patches were active. Note that our patches contain  $72^3$  cells in this case. Thus, the effective resolution for this simulation was ca  $560 \times 360 \times 360$  cells. The whole simulation took only 4.0s/frame, which is very efficient given the high effective resolution.

For the three examples above, we use  $w_s$  accumulated from deformed patch kernels. By considering the deformation,  $w_s$  better represents the coverage of the patches. However, since we limit deformations, we found that using un-deformed kernels generates equivalent visual results with reduced calculation times. Besides, there is still significant room left for accelerating our implementation further. E.g., we run the fluid solve on the CPU, and we only use GPUs for the CNN descriptor calculations.

**Evaluations:** The CNN descriptors not only increase the recall rate, but also improve the visual quality by retrieving patches that better adhere to the input flow. This can be seen in Fig. 4.8, where our result is shown on the right, while the left hand side uses our full pipeline with a simplified distance calculation, i.e., without the use of a CNN. Instead, we use an  $L^2$  distance of down-sampled versions of  $d_s$  and the curl of  $\mathbf{u}_s$ . These values are normalized and used as descriptors directly. Specifically, we down-



**Figure 4.10:** Our method applied to the flow around a cylindrical obstacle. The resolution of the underlying simulation is only  $76 \times 64 \times 64$ .

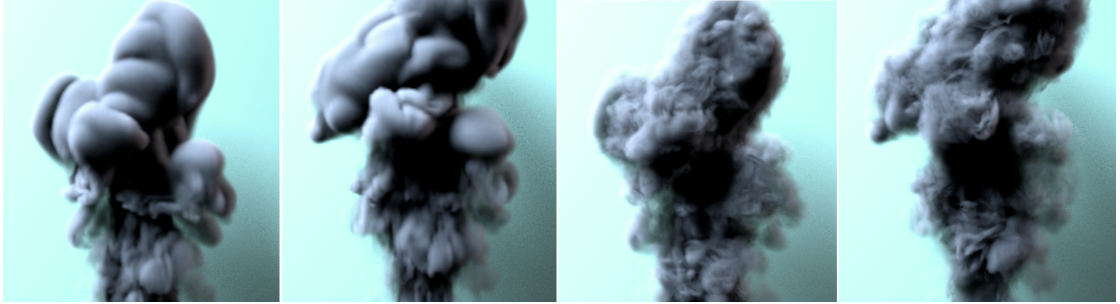


**Figure 4.11:** Two colliding jets of smoke simulated with our approach. The whole simulation including descriptor calculations, look-up, and patch advection took only 4.0 seconds per frame on average.

sampled  $d_s$  to a resolution of  $7^3$ , and the curl to  $5^3$ , resulting in a combined descriptor with  $(343 + 375)$  entries. This is similar to the size of our CNN-based descriptors. Since CNN descriptors have a good understanding of correlation between different fluid resolutions, they offer results with small-scale vortices and vivid structures that fit the target well, while the simple descriptors sometimes offer plain and noisy structures (the blue regions in Fig. 4.8). Additionally, the simple descriptors can introduce un-plausible motions, which becomes apparent in the regions marked in purple in Fig. 4.8. There, we know from theory that the baroclinic contribution to vorticity should be along the cross product of density and pressure gradient. Thus, the vertical structures caused by the simple descriptors are not plausible for the buoyancy driven input simulation.

Based on the setup from Fig. 4.8, we further compare our approach to the wavelet turbulence method [2] as a representative from the class of up-res methods. In order to make the approaches comparable, we consider their performance given a limited and equivalent computational budget. For our simulation, this setup used the  $w_s$  field with un-deformed kernel evaluations. Apart from the difference in detail, the wavelet turbulence version exhibits a noticeable deviation from the input flow in the lower part of the volume. Here numerical diffusion accumulates to cause significant drift over time, while our method continues to closely conform to the input.

Finally, we compare our method to a regular simulation with doubled resolution. As expected, this version results in a different large scale motion, and in order to compare the outputs, we applied our CNN based synthesis method to a down-sampled version of the



**Figure 4.12:** In order to compare a full simulation of  $100 \times 160 \times 100$  (the left two) with our approach, we downsample the full simulation to  $50 \times 80 \times 50$ , and then apply our algorithm (the right two). The latter spends 2.42s/frame, while the full simulation requires 2.51s/frame.

high resolution simulation. While the regular high resolution scene spends 2.5s/frame, our method takes only 2.42s, but offers fine details, as shown in Fig. 4.12.

**Limitations and Discussion:** One limitation of our approach is that we cannot guarantee fully divergence-free motions on small scales. For larger scales, our outputs conform to the original, divergence-free motion. The small scale motions contained in repository patches are likewise recorded from fully divergence-free flows, but as our patches deform slightly, the resulting motions are not guaranteed to be divergence-free. Additionally, spatial blending can introduce regions with divergent motions. Our algorithm shares this behavior with other synthesis approaches, e.g., texture synthesis. However, as we do not need to compute an advection step based on these motions, our method avoids accumulating mass losses (or gains) over time.

There are many avenues for smaller improvements of our neural network approach, e.g., applying techniques such as batch normalization, or specialized techniques for constructing the training set. However, we believe that our current approach already demonstrates that deforming Lagrangian patch regions are an excellent basis for CNNs in conjunction with fluid flows. It effectively makes our learning approach invariant to large scale motions. Removing these invariants for machine learning problems is an important topic, as mentioned e.g. by Ladicky et al. [60]. Apart from motion invariance, we arrive at an algorithm that can easily be applied to any source resolution. For other CNN-based approaches this is typically very difficult to achieve, as networks are specifically trained for a fixed input and output size [30].

Due to its data-driven nature, our method requires more hard-disk space than procedural methods. As shown in Table 4.1, the density data of the patches in our 3D repository takes up ca. 5.1GB of disk space. Fortunately, we only load the descriptors at simulation time: ca. 15MB for density descriptors, and another 15MB for curl descriptors. At render time, we have to load ca. 400 data sets per frame, i.e. ca. 137MB in total.

Another consequence of our machine-learning approach is that our network is specifically adapted to a set of algorithmic choices. Thus, for a different solver, it will be advantageous to re-train the network with a suitable data set and adapted interval  $t_r$ . It will be an interesting area of future work whether a sufficiently deep CNN can learn to compute descriptors for larger classes of solvers.

## 4.5 Conclusions

In Chap. 3, we have presented a novel CNN-based method to learn the flow similarity. Based on that, Chap. 4 introduced a practical data-driven workflow for smoke synthesis. Re-using a flow repository of space-time data-sets to synthesize high-resolution results, the approach only takes a few seconds of run-time per frame. The main contribution of this work is to demonstrate the usefulness of descriptor learning in the context of fluids flows, and we have shown that it lets us establish correspondences between different simulations in the presence of numerical viscosity. As our approach is a data-driven one, it can be used for any choice of Navier-Stokes solver, as long as enough input data is available. Additionally, the localized descriptors make our approach independent of the simulation resolution.

While this method successfully achieved spatial details and temporal coherence for smoke simulations with the help of velocity fields, the motion information is usually not available for other sequential data-set, e.g. natural videos. Applying existing image generation methods to videos often results in temporal artifacts. In the next part, we will focus on the temporally coherent detail synthesis for videos using generative learning methods.



## **Part III**

# **Detail Synthesis for Videos**









## 5 Temporally Coherent Video Super-Resolution



**Figure 5.1:** Our temporally coherent VSR GAN generates realistic results. The low-resolution inputs, our results and the HR ground truth images for the Tears of Steel [127] room scene are shown from top to bottom.

VSR is a conditional video generation task where algorithms take the low resolution (LR) image sequence as inputs and generate high resolution (HR) sequence as outputs. This kind of learning problems are very challenging because networks should not only learn to represent the static data distribution of the target domain but also learn to correlate the output distribution over time with conditional inputs. Without ground-truth motion information, their central difficulty is to faithfully reproduce the temporal dynamics of the target domain and not resort to trivial solutions such as features that arbitrarily appear and disappear over time.

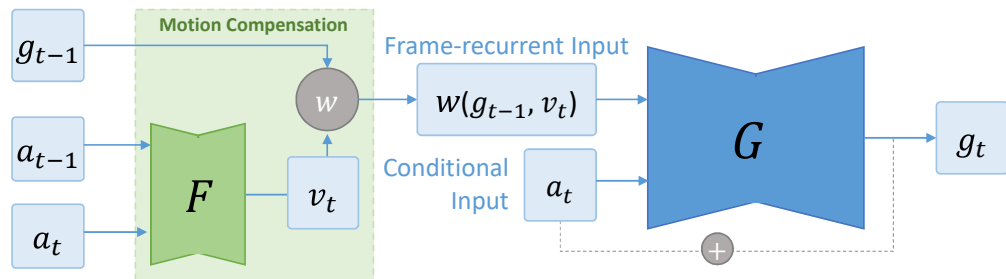
As introduced in Sec. 2.2.3, state-of-the-art VSR methods often favor simpler norm losses such as  $L^2$  over adversarial training. For a multi-modal training problem, these simple losses result in averaging different modalities, thus easily lead to temporally smooth results with an undesirable lack of spatial detail. In this chapter, our goal is to properly apply GANs to improve the learning objective with a spatio-temporal adversarial learning in together with a long-term temporal constraint. These temporal self-supervision techniques allow us to generate high quality contents regarding spatial details as well as temporal coherence. One example is shown in Fig. 5.1. In the following, we first introduce the network architecture and learning objectives step by step. Then

we devote ablation studies, comparisons and results for an in-depth understanding of the approach.

## 5.1 Network Architectures and Temporal Self-Supervisions

Based on GANs introduced in Sec. 2.2.2, our VSR network architecture consists of three components: a recurrent generator, a flow estimation network, and a spatio-temporal discriminator. The generator  $G$  is used to recurrently generate HR video frames from LR inputs. The flow estimation network  $F$  learns the motion compensation between frames to aid both generator as well as the spatio-temporal discriminator  $D_{s,t}$ . During the training, the generator and the flow estimator are trained together to fool the spatio-temporal discriminator  $D_{s,t}$ . Taking spatial as well as temporal aspects into account, this discriminator is the central component of our temporal self-supervision. It has the ability to penalize unrealistic temporal discontinuities in the results without excessively smoothing the image content. In this way,  $G$  is required to generate coherent and high-frequency details according to previous frames. Once trained, the additional complexity of  $D_{s,t}$  does not play a role, as only the trained models of  $G$  and  $F$  are required to infer new super-resolution video outputs. In the following subsections, we describe the generative network and temporal supervisions one by one.

### 5.1.1 Generative Network



**Figure 5.2:** The frame-recurrent VSR generator based on motion compensation.

Our generator networks produce image sequences in a frame-recurrent manner with the help of a recurrent generator  $G$  and a flow estimator  $F$ . We follow previous work [90], where  $G$  produces output  $g_t$  in the target HR domain from a conditional input frame  $a_t$  in the input LR domain, and recursively uses the previous generated output  $g_{t-1}$ . This frame-recurrent structure has the advantage to re-use high-frequency details over time.

Specifically,  $F$  is trained to estimate the motion  $v_t$  between  $a_{t-1}$  and  $a_t$ . Although estimated from low-resolution data,  $v_t$  can be re-sized and is then used as a motion compensation that aligns high-resolution  $g_{t-1}$  to the current frame. Based on the input of  $a_t$  and warped  $g_{t-1}$ , we propose to train our generator to learn the residual content, which we then add to the bi-cubic interpolated low-resolution input. This procedure,

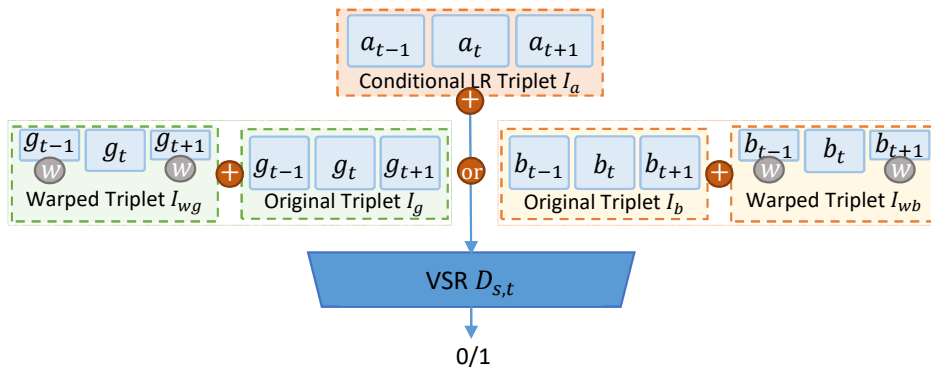
also shown in Fig. 5.2, can be summarized as:

$$\begin{aligned} g_t &= G(a_t, W(g_{t-1}, v_t)) + \text{BicubicResize}(a_t), \text{ where} \\ v_t &= F(a_{t-1}, a_t) \end{aligned} \quad (5.1)$$

and  $W$  is the warping operation. In line with methods for single image processing [78], we find that learning the residual content only makes the training more stable. A ResNet architecture is used for the generator  $G$ , while an encoder-decoder structure is applied to the  $F$ . We intentionally keep generators simple and in line with previous work, in order to demonstrate advantages of the temporal self-supervision that we will explain in the following paragraphs.

### 5.1.2 Spatio-Temporal Adversarial Supervision

The central building block of our approach is a novel *spatio-temporal* discriminator  $D_{s,t}$  that receives triplets of frames. This contrasts with typically used *spatial* discriminators which supervise only a single image. By concatenating multiple adjacent frames along the channel dimension, the frame triplets form an important building block for learning because they can provide networks with gradient information regarding the realism of spatial structures as well as short-term temporal information, such as first- and second-order time derivatives.



**Figure 5.3:** Conditional VSR  $D_{s,t}$ . Taking LR triplet  $I_a$  as a conditional input, the spatio-temporal discriminator supervise the generated triplets ( $I_g$  and  $I_{wg}$ ) according to the ground-truth triplets ( $I_b$  and  $I_{wb}$ ).

We propose a  $D_{s,t}$  architecture, illustrated in Fig. 5.3, that supervises two types of triplets, three adjacent frames and the corresponding warped ones, on condition of the three low-resolution frames from the input domain. In this way,  $D_{s,t}$  can guide the generator to learn the correlation between LR inputs and HR targets. In the warped triplets, we warp later frames backward and previous ones forward. While original frames contain the full spatio-temporal information, warped frames more easily yield temporal information with their aligned content. For the input variants we use the following

notations:

$$\begin{aligned} \mathbf{I}_g &= \{g_{t-1}, g_t, g_{t+1}\}, & \mathbf{I}_b &= \{b_{t-1}, b_t, b_{t+1}\}; \\ \mathbf{I}_{wg} &= \{W(g_{t-1}, v_t), g_t, W(g_{t+1}, v'_t)\}, & \mathbf{I}_{wb} &= \{W(b_{t-1}, v_t), b_t, W(b_{t+1}, v'_t)\}; \\ \mathbf{I}_a &= \{a_{t-1}, a_t, a_{t+1}\}. \end{aligned} \quad (5.2)$$

Here, a subscript of  $a$  denotes the input low-resolution domain, while a subscript of  $b$  denotes the target high-resolution domain.

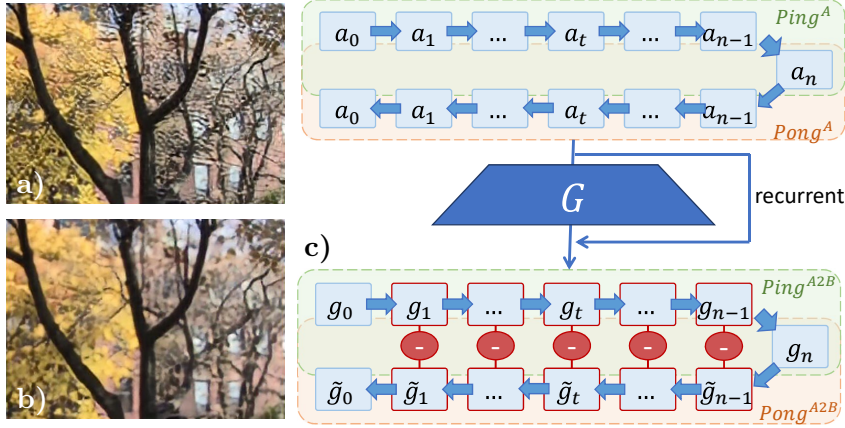
The input of  $D_{s,t}$  can be summarized as  $\mathbf{I}_{s,t}^b = \{\mathbf{I}_b, \mathbf{I}_{wb}, \mathbf{I}_a\}$  labeled as *real* and the generated inputs  $\mathbf{I}_{s,t}^g = \{\mathbf{I}_g, \mathbf{I}_{wg}, \mathbf{I}_a\}$  labeled as *fake*. In this way, the conditional  $D_{s,t}$  will penalize  $G$  if  $\mathbf{I}_g$  contains less spatial details or unrealistic artifacts in comparison to  $\mathbf{I}_a, \mathbf{I}_b$ . At the same time, temporal relationships between the generated images  $\mathbf{I}_{wg}$  and those of the ground truth  $\mathbf{I}_{wb}$  should match. With our setup, the discriminator profits from the warped frames to classify realistic and unnatural temporal changes, and for situations where the motion estimation is less accurate, the discriminator can fall back to the original, i.e. not warped, images.

Compared to non-adversarial training which typically employs loss formulations with static goals, the GAN training yields dynamic goals due to discriminative networks discovering the learning objectives over the course of the training run. Therefore, their inputs have strong influence on the training process and the final results. Modifying their inputs in a controlled manner can lead to different results and substantial improvements with correct settings, as will be shown in Sec. 5.3. Although the concatenation of several frames is a simple operation that has been used in a variety of projects, it is important and allows discriminators to understand spatio-temporal data distributions. Below, we will show that it can effectively reduce temporal problems encountered by spatial GANs.

While  $L^2$ -based temporal losses are widely used in the field of video generation, the spatio-temporal adversarial loss is crucial for preventing the inference of blurred structures in multi-modal data-sets. Compared to GANs using multiple discriminators, the single  $D_{s,t}$  network can learn to balance the spatial and temporal aspects from the reference data and avoid inconsistent sharpness as well as overly smooth results. Additionally, by extracting shared spatio-temporal features, it allows for smaller network sizes.

### 5.1.3 Long-Term Temporal Supervision

When relying on a previous output as input, i.e., for frame-recurrent architectures, generated structures easily accumulate frame by frame. In an adversarial training, generators learn to heavily rely on previously generated frames and can easily converge towards strongly reinforcing spatial features over longer periods of time. For videos, this especially occurs along directions of motion, and these solutions can be seen as a special form of temporal mode collapse. We have noticed this issue in a variety of recurrent architectures, examples are shown in Fig. 5.4(a). While this issue could be alleviated by training with longer sequences, we generally want generators to be able to work with



**Figure 5.4:** a) Result without PP loss. The VSR network is trained with a recurrent frame-length of 10. When inference on long sequences, frame 15 and latter frames of the foliage scene show the drifting artifacts. b) Result trained with PP loss. These artifacts are removed successfully for the latter. c) With our PP loss, the  $L^2$  distance between  $g_t$  and  $\tilde{g}_t$  is minimized to remove drifting artifacts and improve temporal coherence.

sequences of arbitrary length for inference. To address this inherent problem of recurrent generators, we propose a new bi-directional “Ping-Pong(PP)” loss.

For natural videos, a sequence with forward order as well as its reversed counterpart offer valid information. Thus, from any input of length  $n$ , we can construct a symmetric PP sequence in form of  $a_0, \dots, a_{n-1}, a_n, a_{n-1}, \dots, a_0$  as shown in Fig. 5.4. When inferring this in a frame-recurrent manner, the generated result should not strengthen any invalid features from frame to frame. Rather, the result should stay close to valid information and be symmetric, i.e., the forward result  $g_t = G(a_t, g_{t-1})$  and the one generated from the reversed part,  $\tilde{g}_t = G(a_t, \tilde{g}_{t+1})$ , should be identical. Based on this observation, we train our networks with extended PP sequences and constrain the generated outputs from both passes to be the same using the loss:

$$\mathcal{L}_{pp} = \sum_{t=1}^{n-1} \|g_t - \tilde{g}_t\|_2. \quad (5.3)$$

Note that in contrast to the generator loss, the  $L^2$  norm is a correct choice here: We are not faced with multi-modal data where an  $L^2$  norm would lead to undesirable averaging, but rather aim to constrain the recurrent generator to its own, unique version over time. The PP terms provide constraints for short-term consistency via  $\|g_{n-1} - \tilde{g}_{n-1}\|_2$ , while terms such as  $\|g_1 - \tilde{g}_1\|_2$  prevent long-term drifts of the results. The first frame  $g_0$  is ignored because a black image is used as its recurrent input and it usually contains less details comparing to other frames. As shown in Fig. 5.4(b), this PP loss successfully removes drifting artifacts while appropriate high-frequency details are preserved. In addition, it effectively extends the training data set, and as such represents a useful form of data augmentation.

A test in Sec. 5.3.2 disentangles effects of this PP-sequence augmentation and the PP temporal constrains. The results show that the temporal constraint is the key to reliably suppressing the temporal accumulation of artifacts, achieving consistency, and allowing models to infer much longer sequences than seen during training.

## 5.2 Training and Loss Summary

We now explain the setups for the VSR training and losses for all networks. For the training data-set, we download 250 short videos with 120 frames each from [Vimeo.com](https://www.vimeo.com). In line with other VSR projects, we down-sample these frames by a factor of 2 to get the ground-truth HR frames. Corresponding LR frames are achieved by applying a Gaussian blur and sampling every fourth pixel. A Gaussian blur step is important to mimic the information loss due to the camera sensibility in a real-life capturing scenario. Although the information loss is complex and not unified, a Gaussian kernel with a standard deviation of 1.5 is commonly used for a super-resolution factor of 4.

As an adversarial learning, we use a standard cross-entropy loss for the  $D_{s,t}$ :

$$\mathcal{L}_{D_{s,t}} = -\mathbb{E}_{b \sim p_b(b)}[\log D(\mathbf{I}_{s,t}^b)] - \mathbb{E}_{a \sim p_a(a)}[\log(1 - D(\mathbf{I}_{s,t}^a))] . \quad (5.4)$$

Correspondingly, a non-saturated  $\mathcal{L}_{adv}$ , listed in Eq. 5.6, is used for the  $G$  and  $F$  of VSR. Compared to the original saturated adversarial loss [10], the non-saturated one can prevent the gradient vanishing problem.

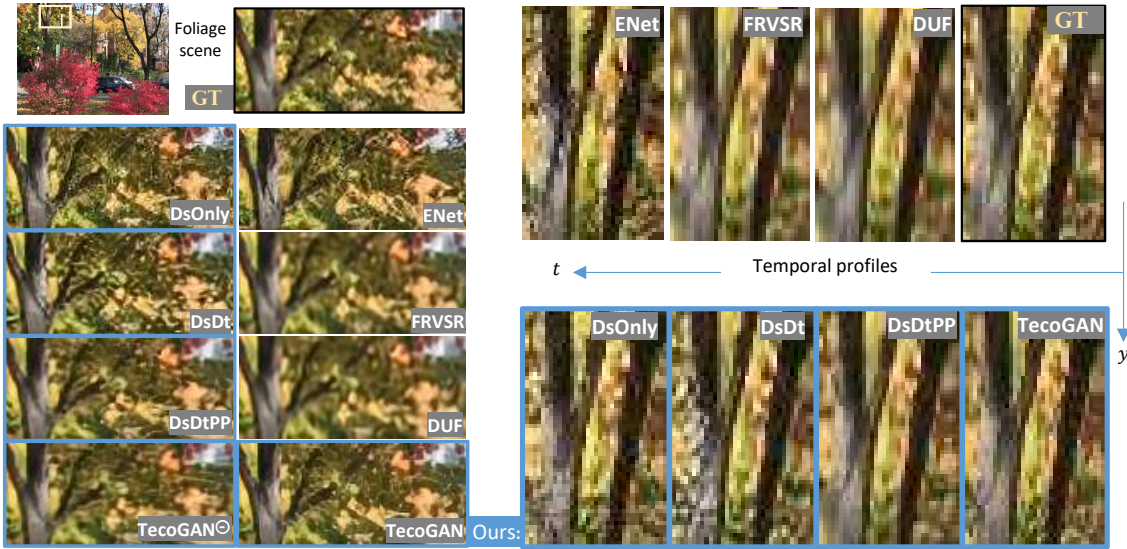
To improve the perceptual quality for image generation tasks, both pre-trained NNs [128, 129] and GANs' discriminators [23] were successfully used as perceptual losses in previous work. Here, we choose to use feature maps from a pre-trained VGG-19 network [15] as well as  $D_{s,t}$  itself. We can encourage the generator to produce features similar to the ground truth ones by increasing the cosine similarity between their feature maps. Note that the feature maps of  $D_{s,t}$  contain both spatial and temporal information, and hence are especially well suited for the perceptual loss.

To summarize, the total loss for  $G$  and  $F$  contains a  $L^2$  content loss, the adversarial loss, the PP loss, perceptual losses and a warping loss to align frames, i.e.:

$$\mathcal{L}_{G,F} = \lambda_c \mathcal{L}_{\text{content}} + \lambda_a \mathcal{L}_{\text{adv}} + \lambda_p \mathcal{L}_{\text{PP}} + \lambda_\phi \mathcal{L}_\phi + \lambda_w \mathcal{L}_{\text{warp}} , \text{ where} \quad (5.5)$$

$$\begin{aligned} \mathcal{L}_{\text{content}} &= \|g_t - b_t\|_2 , \\ \mathcal{L}_{\text{adv}} &= -\mathbb{E}_{a \sim p_a(a)}[\log D_{s,t}(\mathbf{I}_{s,t}^a)] , \\ \mathcal{L}_{\text{PP}} &= \sum_{t=1}^{n-1} \|g_t - g_t'\|_2 , \\ \mathcal{L}_\phi &= 1.0 - \frac{\Phi(\mathbf{I}_{s,t}^g) \cdot \Phi(\mathbf{I}_{s,t}^b)}{\|\Phi(\mathbf{I}_{s,t}^g)\| \cdot \|\Phi(\mathbf{I}_{s,t}^b)\|} , \\ \mathcal{L}_{\text{warp}} &= \sum \|a_t - W(a_{t-1}, F(a_{t-1}, a_t))\|_2 . \end{aligned} \quad (5.6)$$





**Figure 5.5:** In VSR of the foliage scene, adversarial models (ENet, DsOnly, DsDt, DsDtPP, TecoGAN<sup>⊙</sup> and TecoGAN) yield better perceptual quality than methods using  $L^2$  loss (FRVSR and DUF). In temporal profiles on the right, DsDt, DsDtPP and TecoGAN show significantly less temporal discontinuities compared to ENet and DsOnly. The temporal information of our discriminators successfully suppresses these artifacts.

Again  $g$ ,  $b$  and  $\Phi$  stand for generated samples, ground truth images and feature maps of VGG or  $D_{s,t}$ . Training with the total loss, we refer to our full model as *TecoGAN* below. In the following, ablation studies are offered to reveal the effect of each loss term and training details including parameters are given in Appendix A.2.

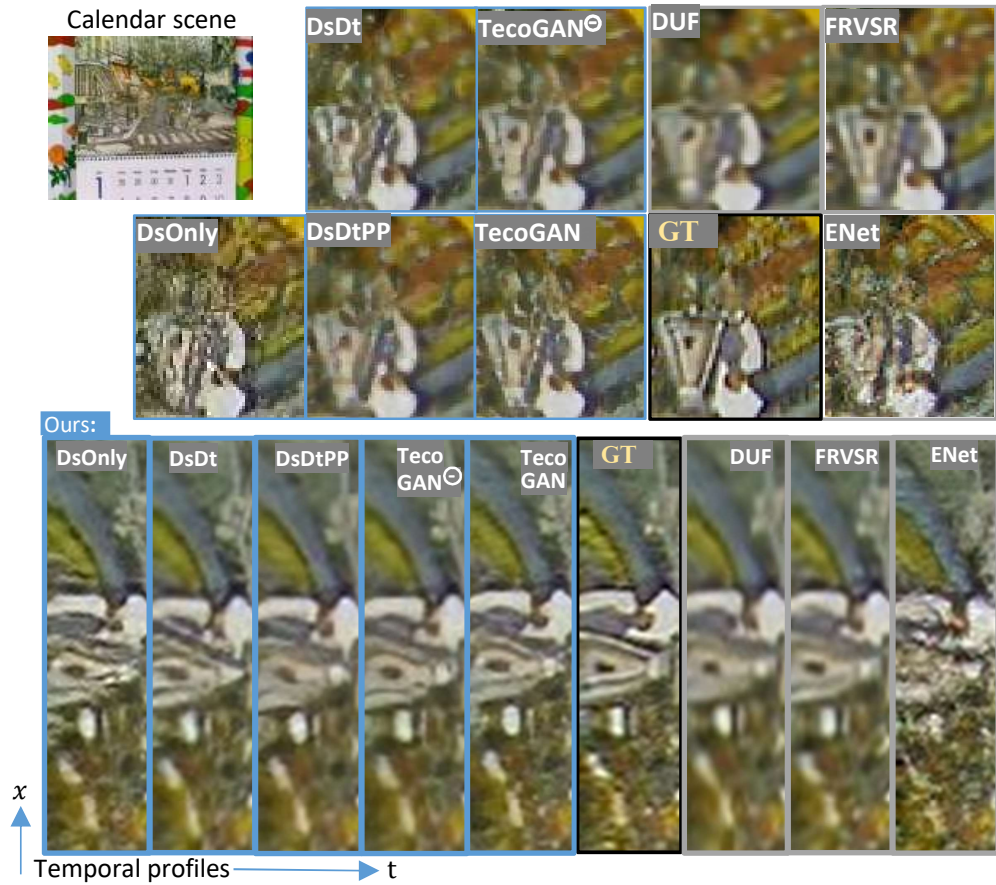
### 5.3 Learning Objectives Analysis

We will first illustrate the effects of temporal supervision, i.e.  $\mathcal{L}_{adv}$  and  $\mathcal{L}_{PP}$ , using models trained with ablated loss functions. Then, we focus on  $\mathcal{L}_{PP}$  and activate the PP augmentation separately to compare to full TecoGAN models.

#### 5.3.1 Loss Ablation Study

Below, we compare variants of our TecoGAN model to EnhanceNet (ENet) [130], FRVSR [90], and DUF [93] for VSR. While ENet represents state-of-the-art single-image adversarial model without temporal information, FRVSR and DUF are state-of-the-art VSR methods without adversarial losses.

For TecoGAN variants, we first train a *DsOnly* model with a frame-recurrent  $G$ , the  $F$ , a VGG loss and the regular spatial-only discriminator. Compared to ENet, which exhibits strong incoherence due to the lack of temporal information, DsOnly improves temporal coherence thanks to the frame-recurrent connection, but there are noticeable



**Figure 5.6:** VSR temporal profile comparisons of the calendar scene (time shown along y-axis), TecoGAN models lead to natural temporal progressions, and our final model closely matches the desired ground truth behavior over time.

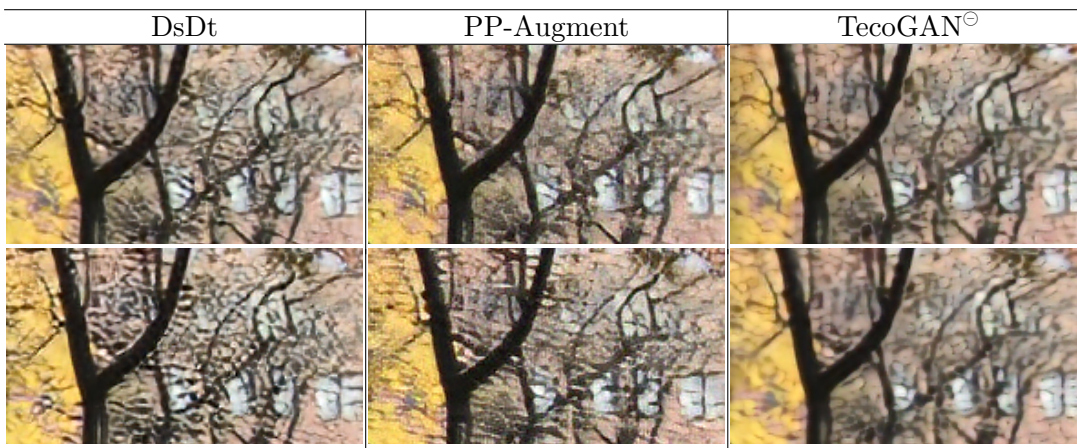
high-frequency changes between frames. The temporal profiles of DsOnly in Fig. 5.5 and 5.6, correspondingly contain sharp and broken lines. When adding a temporal discriminator in addition to the spatial one, this *DsDt* version generates more coherent results, and its temporal profiles are sharp and coherent. However, DsDt often produces the drifting artifacts discussed in Sec. 5.1.3, as the generator learns to reinforce existing details from previous frames to fool  $D_s$  with sharpness, and satisfying  $D_t$  with good temporal coherence in the form of persistent detail. While this strategy works for generating short sequences during training, the strengthening effect can lead to very undesirable artifacts for long-sequence inferences. By adding the self-supervision for long-term temporal consistency  $\mathcal{L}_{pp}$ , we arrive at the *DsDtPP* model, which effectively suppresses these drifting artifacts with an improved temporal coherence. In Fig. 5.5 and Fig. 5.6, DsDtPP results in continuous yet detailed temporal profiles without streaks from temporal drifting. Although DsDtPP generates good results, it is difficult in practice to balance the generator and the two discriminators. The results shown here were achieved only

after numerous runs manually tuning the weights of the different loss terms. By using the proposed  $D_{s,t}$  discriminator instead, we get a first complete model for our method, denoted as  $TecoGAN^\ominus$ . This network is trained with a discriminator that achieves an excellent quality with an effectively halved discriminative network size, as illustrated on the right of Fig. 7.1. The single discriminator correspondingly leads to a significant reduction in resource usage. Using two discriminators requires ca. 70% more GPU memory, and leads to a reduced training performance by ca. 20%. The  $TecoGAN^\ominus$  model yields similar perceptual and temporal quality to DsDtPP with a significantly faster and more stable training.

Since the  $TecoGAN^\ominus$  model requires less training resources, we also trained a larger generator with 50% more weights. In the Sec. 5.4, we will focus on this larger single-discriminator architecture with PP loss as our full  $TecoGAN$  model for VSR. Compared to the  $TecoGAN^\ominus$  model, it can generate more details, and the training process is more stable, indicating that the larger generator and  $D_{s,t}$  are more evenly balanced. Result images and temporal profiles are shown in Fig. 5.5 and Fig. 5.6.

### 5.3.2 Data Augmentation and Temporal Constrains in the PP loss

Since training with sequences of arbitrary length is not possible with current hardware, problems such as the “streaking artifacts” discussed above generally arise for recurrent models. In the proposed PP loss, both the Ping-Pong data augmentation and the temporal consistency constraint contribute to solving these problems. In order to show their separated contributions, we trained another  $TecoGAN$  variant that only employs the data augmentation without the constraint (i.e.,  $\lambda_p = 0$  in Eq. 5.5). Denoted as “PP-Augment”, we show its results in comparison with the DsDt and  $TecoGAN^\ominus$  models in Fig. 5.7.



**Figure 5.7:** 1st & 2nd row: Frame 15 & 40 of the *Foliage* scene. While DsDt leads to strong recurrent artifacts early on, PP-Augment shows similar artifacts later in time (2nd row, middle).  $TecoGAN^\ominus$  model successfully removes these artifacts.

During training, the generator of DsDt receives 10 frames, and generators of PP-Augment and TecoGAN<sup>⊙</sup> see 19 frames. While DsDt shows strong recurrent accumulation artifacts early on, the PP-Augment version slightly reduces the artifacts. In Fig. 5.7, it works well for frame 15, but shows artifacts from frame 32 on. Only our regular model (TecoGAN<sup>⊙</sup>) successfully avoids temporal accumulation for all 40 frames. Hence, with the PP constraint, the model avoids recurrent accumulation of artifacts and works well for sequences that are substantially longer than the training length. Among others, we have tested our model with sequences in length of 150, 166 and 233. For all of these sequences, the TecoGAN model successfully avoids temporal accumulation or streaking artifacts.

## 5.4 Results and Performance

Our model is tested on a wide range of video data, including the widely used Vid4 data-set shown in Fig. 5.5, 5.6 and 5.8, detailed scenes from the movie Tears of Steel (ToS, 2011) shown in Fig. 5.8, and others shown in Fig. 5.10. As mentioned in Sec. 5.2, our model is trained with down-sampled inputs and it can similarly work with original images that were not down-sampled or filtered, such as a data-set of real-world photos. In Fig. 5.9, we compared our results to two other methods [131, 94] that have used the same data-set. With the help of adversarial learning, our model is able to generate improved and realistic details in down-sampled as well as captured images.

While generator and discriminator are trained together, we only need the trained generator network for the inference of new outputs after training, i.e., the whole discriminator network can be discarded. Evaluated on a Nvidia GeForce GTX 1080Ti GPU with 11G memory, the resulting VSR performance is given below. Numbers are averaged over 500 images up-scaled from  $320 \times 134$  to  $1280 \times 536$ .

The TecoGAN<sup>⊙</sup> model and FRVSR have the same number of weights (843587 in the SRNet, i.e. generator network, and 1.7M in F), and thus show very similar performance characteristics with around 37 ms spent for one frame. The larger TecoGAN model with 1286723 weights in the generator is slightly slower than TecoGAN<sup>⊙</sup>, spending 42 ms per frame. However, compared with the DUF model, which has more than 6 million weights in total and spends 942 ms per frame, the TecoGAN performance is significantly better thanks to its reduced size.

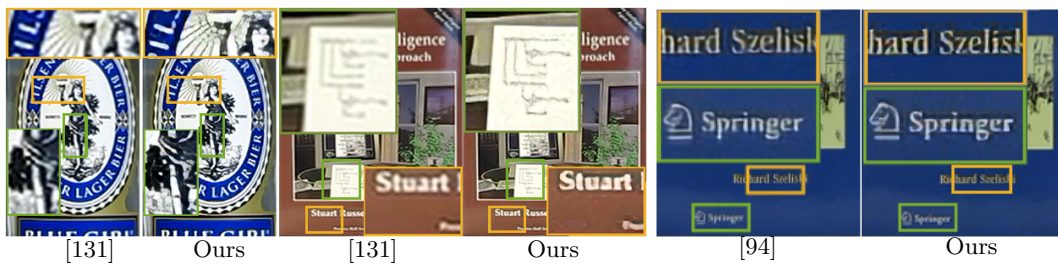
## 5.5 Limitations and Conclusions

Using the proposed discriminator architecture and PP loss, we find it is possible to learn stable temporal functions with GANs. We have shown that these temporal self-supervisions yield coherent and sharp details for VSR problems that go beyond what can be achieved with direct supervision.

While our method generates very realistic results for a wide range of natural images, it can lead to temporally coherent yet sub-optimal details in certain cases such as under-resolved faces and text. This is a typical problem for GANs and is usually resolved by



**Figure 5.8:** Detail views of the VSR results of ToS scenes (first three columns) and Vid4 scenes (last two columns) generated with methods from top to bottom: ENet, FRVSR, DUF, TecoGAN, and the ground truth.



**Figure 5.9:** VSR results on captured images comparing to previous work [131, 94].



**Figure 5.10:** Additional VSR comparisons, generated with methods from left to right: ENet, FRVSR, DUF, TecoGAN, and the ground truth. TecoGAN generates sharp details in both scenes.

introducing prior information for the content of the video. Also, the interplay of the different loss terms in the non-linear training procedure does not provide a guarantee that all goals are fully reached every time. However, we found our method to be stable over a large number of training runs, and we anticipate that it will provide a very useful basis for wide range of conditional video generation tasks. As an example, in the next chapter, we will apply these temporal supervisions to the UVT task and demonstrate that temporal cycle consistency can be established through self supervisions.

## 6 Unpaired Video Translation with Temporal Self-Supervisions



**Figure 6.1:** Being able to establish the correct temporal cycle-consistency between domains, our method generates correct blinking and speaking motions for eyes and mouths with nice details (video courtesy of the White House, public domain).

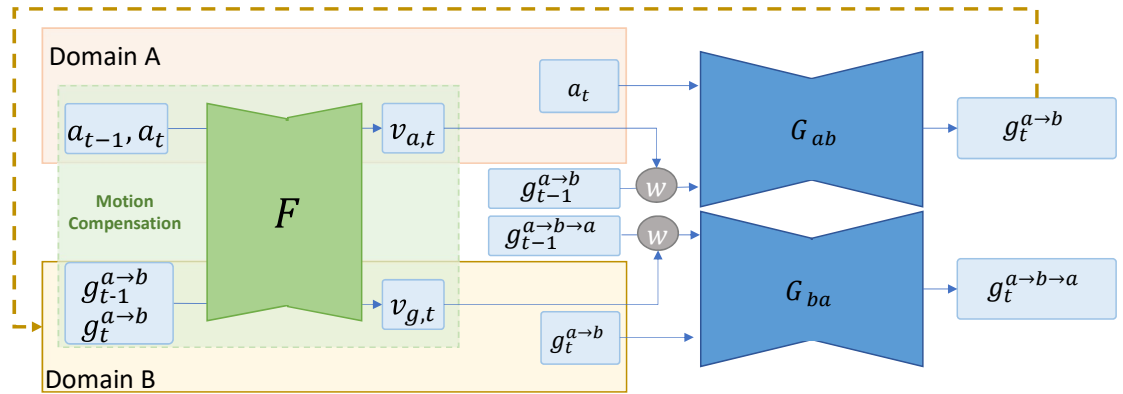
While Chap. 5 focuses on the video super-resolution tasks, in this chapter, we look at another conditional video generation task, video translation, which also requires natural temporal evolution. Different to the VSR task where paired LR and HR videos are available, it is usually hard to get training data-sets with one-to-one assignments when translating content from one video domain to another. As discussed in Sec. 2.2.3, methods have been proposed to establish spatial cycle consistency [17] as well as temporal cycle consistency [22] between two domains.

In the following, we show that instead of working on the generative networks, the temporal cycle consistency can be likewise established through temporal self-supervision. With the proposed spatio-temporal adversarial training and the Ping-Pong constraint, we achieve translations with impressive details and natural temporal evolution. Some examples are shown in Fig. 6.1, where a video of Trump is translated from a video of Obama with desired blinking and speaking motions. Below, we introduce the network architectures and learning objectives. Again, we offer ablation studies and comparisons to present a careful analysis on learning objectives.

## 6.1 Network Architectures

CycleGAN establishes the spatial cycle consistency for unpaired image-to-image translation using two generators and two discriminators [17]. Extending to video data, our UVT method contains two frame-recurrent generators translating between domain A and B, a motion estimator  $F$  and two spatio-temporal discriminators supervising in domain A and B respectively. While the spatial cycle consistency is likewise established through forward-and-backward translations from one domain to the other explicitly, we will show that temporal cycle consistency can be guaranteed through our temporal supervisions. In the following, we will describe the generative and discriminative network architectures.

### 6.1.1 Frame-Recurrent Generative Network



**Figure 6.2:** Frame-recurrent generators based on motion compensation and cycle consistency.

While we use one generator to map data from domain A to B for the VSR task, unpaired generation tasks requires a second generator to establish the spatial cycle consistency. For the UVT task, we use two recurrent generators, mapping from domain A to B and back. As shown in Fig. 6.2, given  $g_t^{a \rightarrow b} = G_{ab}(a_t, W(g_{t-1}^{a \rightarrow b}, v_t))$ , we can use  $a_t$  as the labeled data of  $g_t^{a \rightarrow b \rightarrow a} = G_{ba}(g_t^{a \rightarrow b}, W(g_{t-1}^{a \rightarrow b \rightarrow a}, v_t))$  to enforce consistency. Likewise,  $b_t$  can be used as the labeled data of  $g_t^{b \rightarrow a \rightarrow b} = G_{ba}(g_t^{b \rightarrow a}, W(g_{t-1}^{b \rightarrow a \rightarrow b}, v_t))$ , which is not shown in the figure. This time, an encoder-decoder structure is applied to both the generators and  $F$  and generators should learn the entire translation instead of a residual content as in VSR.

In Sec. 6.2.1, we will show that using this generative network in together with two spatial discriminator, it is possible to get a sequence of translated frames. Each of them will contain spatial features of the target domain, but the temporal evolution cannot be correctly transferred. This lead us to the proposed temporal self-supervisions using spatio-temporal discriminators and the PP constraint.



### 6.1.2 Spatio-Temporal Discriminative Network

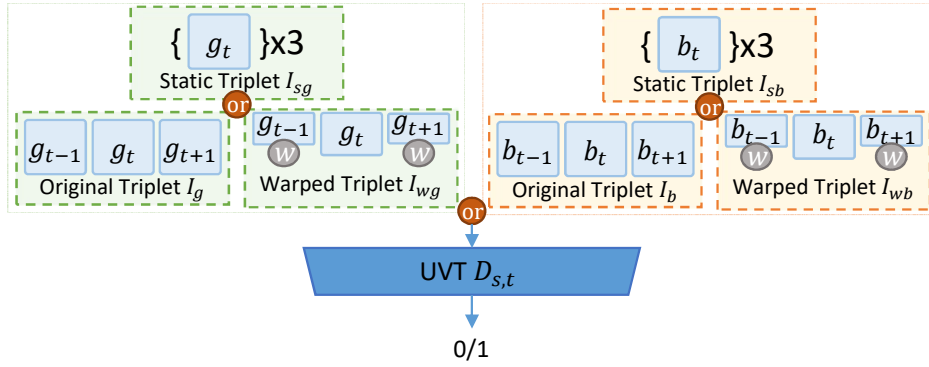


Figure 6.3: Conditional UVT  $D_{s,t}$ .

Being orthogonal to previous work which focuses on the generative networks to form spatio-temporal cycle links, we will demonstrate that the supervision using unconditional spatio-temporal discriminators has the ability to establish the temporal cycle-consistency between different domains. As an improved learning objective, this actually yields better results, as we will show in Sec. 6.3.

Extending the input frames from static ones to triplets, the UVT discriminator  $D_{s,t}$  can provide spatio-temporal supervision to resemble the target domain. As illustrated in Fig. 6.3, three types of triplets are supervised, denoted as static triplets, warped triplets and the original triplets:

$$\begin{aligned}
 I_{sg} &= \{g_t, g_t, g_t\}, & I_{sb} &= \{b_t, b_t, b_t\}; \\
 I_{wg} &= \{W(g_{t-1}, v_t), g_t, W(g_{t+1}, v_t')\}, & I_{wb} &= \{W(b_{t-1}, v_t), b_t, W(b_{t+1}, v_t')\}; \\
 I_g &= \{g_{t-1}, g_t, g_{t+1}\}, & I_b &= \{b_{t-1}, b_t, b_{t+1}\}.
 \end{aligned} \tag{6.1}$$

Again, subscript of  $a$  and  $b$  denotes the input and target domains respectively and the warping is performed via  $F$ .

Since the training data is unpaired, we train unconditional discriminators. Thus,  $D_{s,t}$  is not designed to supervise the correlation between conditional inputs and the outputs. On the contrary, it should first ensure that generators learn reasonable spatial features and based on them, it can supervise the temporal continuity as well as the correlation between spatial features and temporal motion. For example, when translating a face from one person to another, the  $D_{s,t}$  should first teach the generator to learn spatial features including eyes, hairs, and so on. Then, it should penalize  $G$  for temporal discontinuity such as high frequency jitters. At last, it should understand spatio-temporal correlations, e.g. there is a high possibility for mouths or eyes to open and close, while it is less likely for parts like shoulders and noses to move a lot. In practice, we found it crucial to ensure that the  $D_{s,t}$  supervision starts from spatial features and temporal information is considered gradually.

Therefore, different to the  $D_{s,t}$  of VST that always receives 3 concatenated triplets as an input, the unconditional  $D_{s,t}$  of UVT only takes one triplet at a time. Taking the generated data as the example, the input for a single batch can either be a static triplet of  $I_{sg}$  the warped triplet  $I_{wg}$ , or the original triplet  $I_g$ . The same holds for the reference data of the target domain. With sufficient but complex information contained in these triplets, transition techniques are applied so that the network can consider the spatio-temporal information step by step, i.e., we initially start with 100% static triplets  $I_{sg}$  as the input. Then, over the course of training, 25% of them transit to  $I_{wg}$  triplets with simpler temporal information, and another 25% transit to  $I_g$  afterwards, leading to a (50%,25%,25%) distribution of triplets. The transitions of the warped triplets are computed with linear interpolation:  $(1 - \alpha)I_{cg} + \alpha I_{wg}$ , with  $\alpha$  growing from 0 to 1. For the original triplets, we additionally fade the ‘‘warping’’ operation out by using  $(1 - \alpha)I_{cg} + \alpha\{W(g_{t-1}, v_t * \beta), g_t, W(g_{t+1}, v'_t * \beta)\}$ , again with  $\alpha$  growing from 0 to 1 and  $\beta$  decreasing from 1 to 0. These smooth transitions are important for discriminators to learn spatio-temporal data distributions in a gradual and stable manner. A detailed analysis on different discriminator inputs will be given in Sec. 6.2.2.

## 6.2 Losses and Analysis

While a spatio-temporal adversarial learning based on the proposed networks offers reasonable video translation results for short sequences, the long-term PP constraint  $\mathcal{L}_{PP}$  is necessary to avoid temporal accumulation of artifacts in longer sequences. In addition, we use a perceptual loss  $\mathcal{L}_\phi$  based on discriminators to accelerate the convergence of the training. Without paired ground truth data,  $\mathcal{L}_{PP}$  is calculated with feature correlations measured by the Gram matrix, similar to the style loss used for traditional style transfer tasks [128]. It helps the generators to match the distribution of features in the target domain.

To summarize, the loss we use for the UVT  $D_{s,t}$  is

$$\mathcal{L}_{D_{s,t}} = \mathbb{E}_{b \sim p(b)} [D(I_{s,t}^b) - 1]^2 + \mathbb{E}_{a \sim p(a)} [D(I_{s,t}^a)]^2 \quad (6.2)$$

and the generators are trained with the content loss  $\mathcal{L}_{\text{content}}$  based on the spatial cycle consistency, adversarial losses  $\mathcal{L}_{\text{adv}}$ , perceptual losses  $\mathcal{L}_\phi$ , and the PP loss  $\mathcal{L}_{PP}$ , i.e.:

$$\mathcal{L}_{G,F} = \lambda_c \mathcal{L}_{\text{content}} + \lambda_a \mathcal{L}_{\text{adv}} + \lambda_p \mathcal{L}_{PP} + \lambda_\phi \mathcal{L}_\phi, \text{ where} \quad (6.3)$$

$$\begin{aligned} \mathcal{L}_{\text{content}} &= \left\| g_t^{a \rightarrow b \rightarrow a} - a_t \right\|_2 + \left\| g_t^{b \rightarrow a \rightarrow b} - b_t \right\|_2, \\ \mathcal{L}_{\text{adv}} &= - \mathbb{E}_{a \sim p_a(a)} [D_{s,t}^b(I_{s,t}^{a \rightarrow b}) - 1]^2, \\ \mathcal{L}_{PP} &= \sum_{t=1}^{n-1} \|g_t - g'_t\|_2, \\ \mathcal{L}_\phi &= \left\| GM(\Phi(I_{s,t}^g)) - GM(\Phi(I_{s,t}^b)) \right\|_2. \end{aligned} \quad (6.4)$$

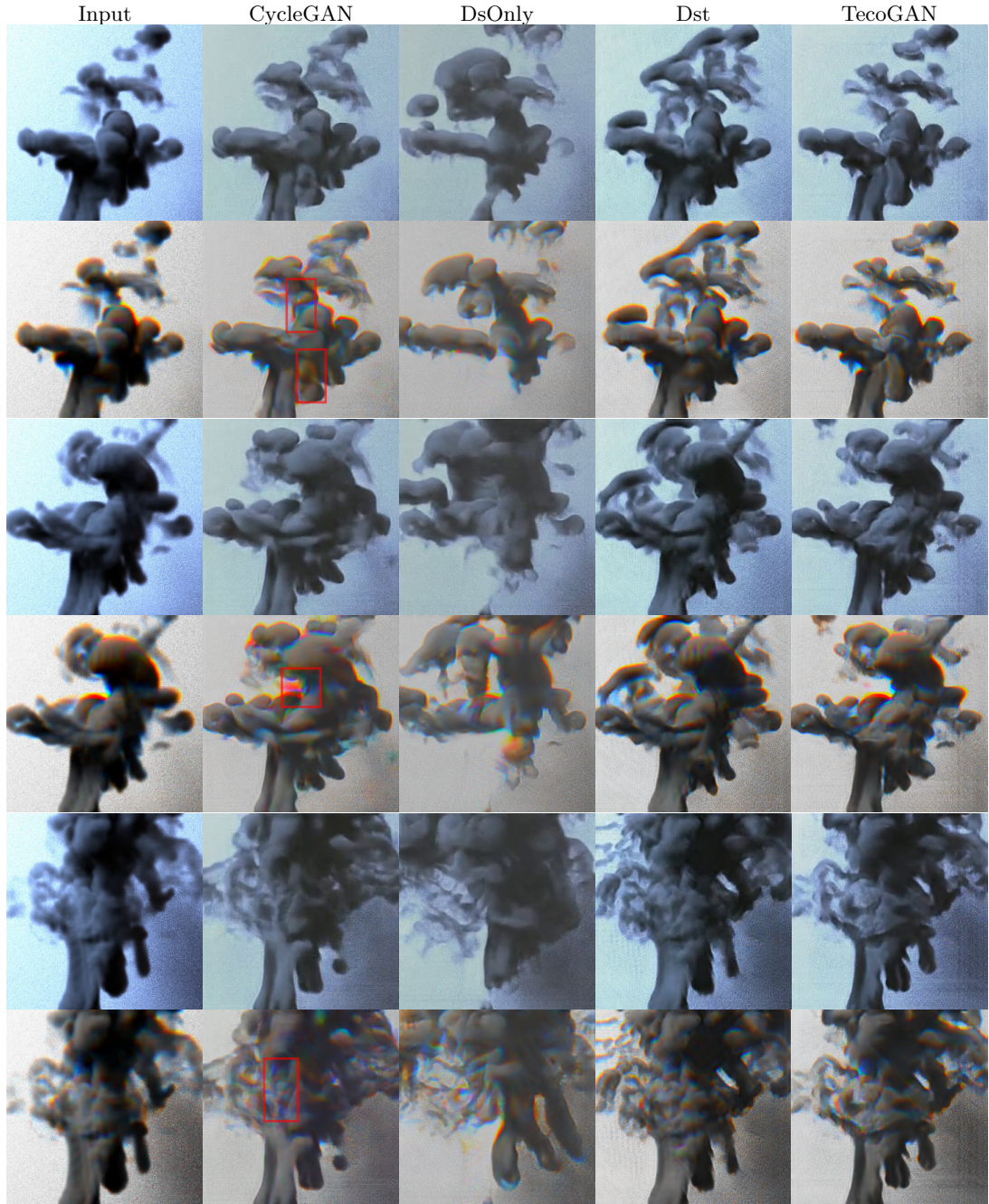
Notations  $g$ ,  $b$  and  $\Phi$  stand for generated samples, ground truth images and feature maps of  $D_{s,t}$ . We only show losses for the mapping from A to B, as the backward mapping simply mirrors the terms. Note that we train least-square GANs [71] to avoid the vanishing gradients of the vanilla GANs. To simplify the adversarial training, we load a pre-trained model as  $F$  and the loss function  $\mathcal{L}_{\text{warp}} = \sum \|a_t - W(a_{t-1}, F(a_{t-1}, a_t))\|_2$  was used in the pre-training stage.

The UVT data-sets are obtained from previous work [22] and each data domain has around 2400 to 3600 unpaired frames. Again, we refer to our full model as *TecoGAN* and analyze learning objectives using ablation studies below. Training details and parameters are given in Appendix A.2.

### 6.2.1 Loss Ablation Study

Similar to the ablation study shown for the VSR task, we train *TecoGAN* variants to evaluate the effect of each loss terms. Again, we start from a single-image GAN-based model, a *CycleGAN* variant which already has two pairs of spatial generators and discriminators. Then, we train the *DsOnly* variant by adding flow estimation via  $F$  and extending the spatial generators to frame-recurrent ones. By augmenting the two discriminators to use the triplet inputs proposed in Sec. 6.1.2, we arrive at the *Dst* model with spatio-temporal discriminators, which does not yet use the PP loss. Although UVT tasks are substantially different from VSR tasks, the comparisons in Fig. 6.4 yield similar conclusions. In these tests, we use renderings of 3D fluid simulations of rising smoke as our unpaired training data. These simulations are generated with randomized numerical simulations using a resolution of  $64^3$  for domain A and  $256^3$  for domain B, and both are visualized with images of size  $256^2$ . Therefore, video translation from domain A to B is a tough task, as the latter contains significantly more turbulent and small-scale motions. With no temporal information available, the *CycleGAN* variant generates HR smoke that strongly flickers. The *DsOnly* model offers better temporal coherence by relying on its frame-recurrent input, but it learns a solution that largely ignores the current input and fails to keep reasonable spatio-temporal cycle-consistency links between the two domains. On the contrary, our  $D_{s,t}$  enables the *Dst* model to learn the correlation between the spatial and temporal aspects, thus improving the cycle-consistency. However, without  $\mathcal{L}_{pp}$ , the *Dst* model (like the *DsDt* model of VSR) reinforces detail over time in an undesirable way. This manifests itself as inappropriate smoke density in empty regions. Using our full *TecoGAN* model which includes  $\mathcal{L}_{pp}$ , yields the best results, with detailed smoke structures and correct spatio-temporal cycle-consistency.

For comparison, a *DsDtPP* model involving a larger number of separate networks, i.e. four discriminators and two frame-recurrent generators, are trained. By weighting the temporal adversarial losses from *Dt* with 0.3 and the spatial ones from *Ds* with 0.5, we arrived at a balanced training run. Although this model performs similarly to the *TecoGAN* model on the smoke dataset, the proposed spatio-temporal  $D_{s,t}$  architecture represents a more preferable choice in practice, as it learns a natural balance of temporal and spatial components by itself, and requires fewer resources. Continuing along this



**Figure 6.4:** In every two rows, we show translations on top and the bottom row contains three adjacent frames as RGB channels, thus fully aligned triplets would be gray. When learning a mapping from renderings of LR smoke volumes to HR ones, CycleGAN gives good spatial features, but is temporally incoherent. It generates triplets with strong colors while the input one only shows color around smoke surfaces. DsOnly relies too much on the recurrent input and diverges from the conditional input. Dst generates smoke in originally empty regions. TecoGAN outperforms other variants with desired results.



**Figure 6.5:** When learning a mapping from Trump to Obama, CycleGAN gives good spatial features but collapses to essentially static Obama outputs. It manages to transfer expressions back to Trump using tiny differences encoded in its Obama outputs, without understanding the cycle-consistency between two domains. DsOnly also fails and generates weird motions. Being able to establish the correct temporal cycle-consistency, ours and RecycleGAN generate correct blinking motions, while ours contains more coherent details.

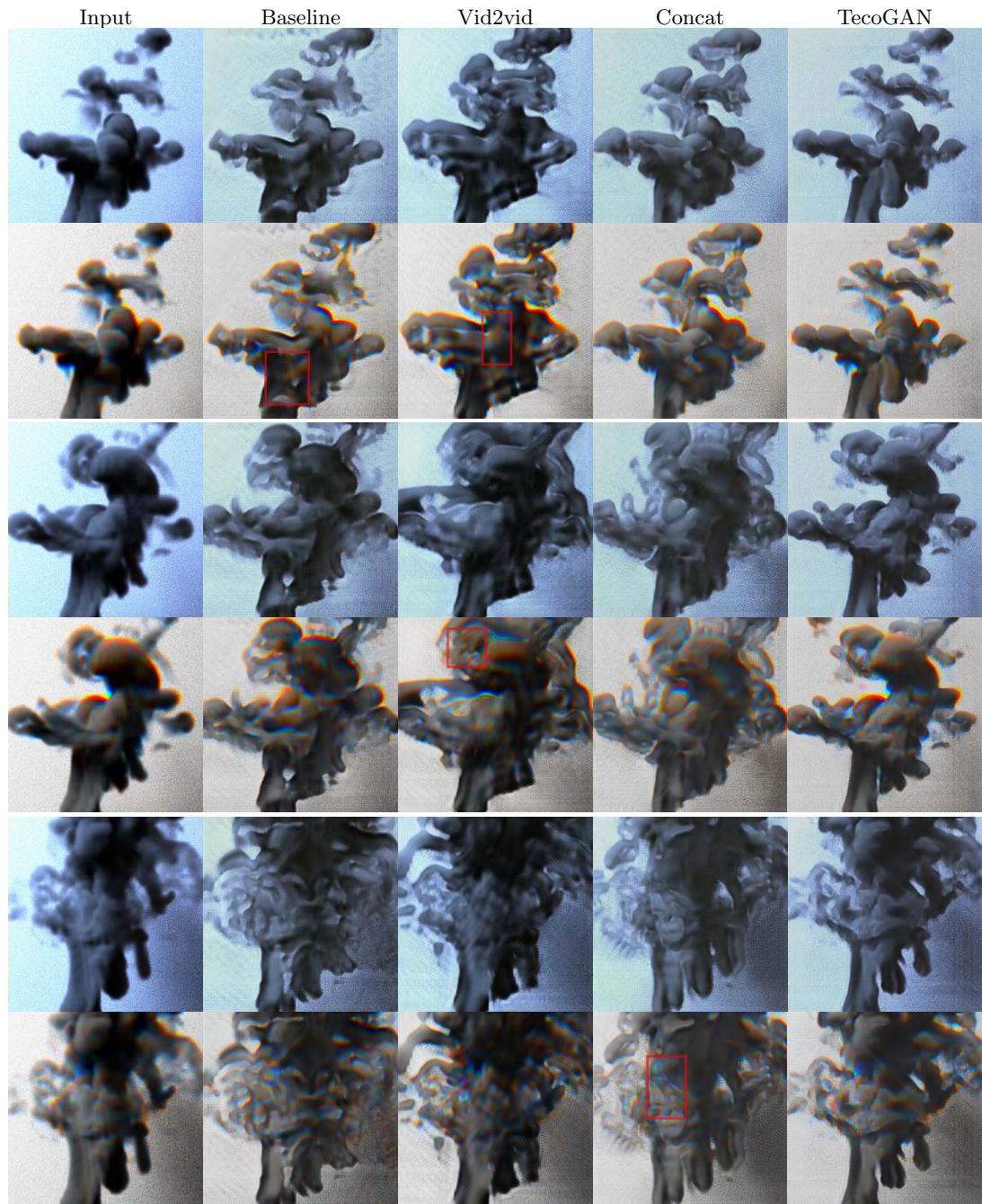
direction, an interesting future work is to evaluate variants, such as a shared  $D_{s,t}$  for both domains, i.e. a multi-class classifier network.

Besides the smoke dataset, an ablation study for the Obama and Trump translation in Fig. 6.5 shows a very similar behavior.

### 6.2.2 Spatio-temporal Adversarial Equilibriums

Our evaluation so far highlights that temporal adversarial learning is crucial for generators to learn the spatio-temporal correlation between domains. Next, we will shed light on the complex spatio-temporal adversarial learning objectives by varying the information provided to the discriminator network. In the following tests, shown in Fig. 6.6,  $D_{s,t}$  networks are identical apart from changing inputs, and we focus on the smoke translation.

In order to learn the spatial and temporal features of the target domain as well as their correlation, the simplest input for  $D_{s,t}$  consists of only the original, unwarped triplets, i.e.  $\{I_g$  or  $I_b\}$ . Using these, we train a *baseline* model, which yields a sub-optimal quality: it lacks sharp spatial structures, and contains coherent but dull motions. Despite containing the full information, these input triplets prevent  $D_{s,t}$  from providing the desired supervision. For paired video translation tasks, the *vid2vid* network achieves improved temporal coherence by using a video discriminator to supervise the output sequence conditioned with the motion estimated from a paired ground-truth sequence. With no paired data available, we train a *vid2vid* variant by using the estimated motions and original triplets, i.e.  $\{I_g + F(g_{t-1}, g_t) + F(g_{t+1}, g_t)$  or  $I_b + F(b_{t-1}, b_t) + F(b_{t+1}, b_t)\}$ , as the input for  $D_{s,t}$ . However, the result do not significantly improve. The motions are



**Figure 6.6:** Adversarial training arrives at different equilibriums with different discriminator inputs. The baseline model (supervised on original triplets) and the vid2vid variant (supervised on original triplets and estimated motions) fail to learn the complex temporal dynamics of a high-resolution smoke. The warped triplets improve the result of the concat model and the full TecoGAN model performs better spatio-temporally.



**Figure 6.7:** Results of UVT tasks on different data-sets.

only partially reliable, and hence cannot help for the difficult unpaired translation task. Therefore, the discriminator still fails to fully correlate spatial and temporal features.

We then train a third model, *concat*, using the original triplets and the warped ones, i.e.  $\{I_g + I_{wg}$  or  $I_b + I_{wb}\}$ . In this case, the model learns to generate more spatial details with a more vivid motion. I.e., the improved temporal information from the warped triplets gives the discriminator important cues. However, the motion still does not fully resemble the target domain. We arrive at our final *TecoGAN* model for UVT by controlling the composition of the input data: as outlined above, we first provide only static triplets  $\{I_{sg}$  or  $I_{sb}\}$ , and then apply the transitions of warped triplets  $\{I_{wg}$  or  $I_{wb}\}$ , and original triplets  $\{I_g$  or  $I_b\}$  over the course of training. In this way, the network can first learn to extract spatial features, and build on them to establish temporal features. Finally, discriminators learn features about the correlation of spatial and temporal content by analyzing the original triplets, and provide gradients such that the generators learn to use the motion information from the input and establish a correlation between the motions in the two unpaired domains. Consequently, the discriminator, despite receiving only a single triplet at once, can guide the generator to produce detailed structures that move coherently.

### 6.3 Results and Conclusions

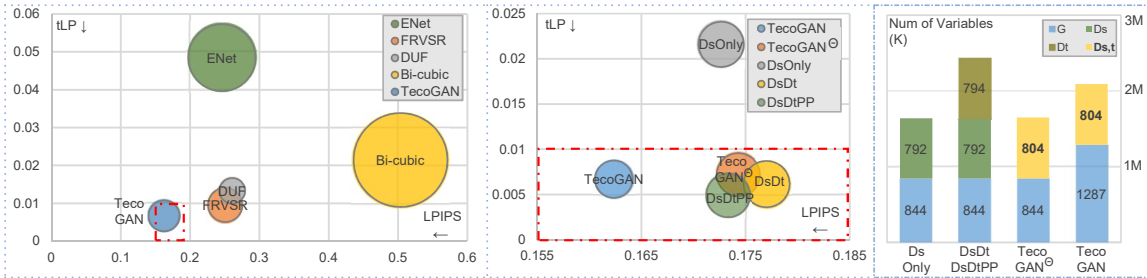
Using the same loss function and hyper parameters, we train models for the following video translation tasks: Obama and Trump translations, LR- and HR- smoke simulation translations, as well as translations between smoke simulations and real-smoke captures. While smoke simulations usually contain strong numerical viscosity with details limited by the simulation resolution, the real smoke from Eckert et al. [132], contains vivid fluid motions with many vortices and high-frequency details. As shown in Fig. 6.7, our method can be used to narrow the gap between simulations and the real-world phenomena.

Thanks to the proposed temporal self-supervisions, TecoGAN outperforms previous work on the Obama and Trump data-set with sharp spatial features and coherent motion, as shown in Fig. 6.5. It achieves good temporal coherence on par with RecycleGAN and its spatial detail is on par with CycleGAN. While we achieve very realistic results on these translation tasks, TecoGAN may generate sub-optimal results when the motion of two domains strongly differs. For this case, it would be interesting to apply both our method and motion translation from MecycleGAN [96] which learns the motion translations in addition to image translations. This can make it easier for the generator to learn from our temporal self supervision.

In the previous and this chapters, we show that for both VSR and UVT tasks, the proposed temporal supervision provides results with natural temporal evolution and realistic details. In the next chapter, we will apply existing spatial metrics and propose temporal metrics to quantify the spatio-temporal quality of video results. User studies are also conducted, which confirms the metric evaluations.



## 7 Spatio-Temporal Evaluations for Videos



**Figure 7.1:** Visual summary of VSR models with LPIPS (X-axis) measuring spatial detail and temporal coherence measured by tLP (y-axis) & tOF (bubble size with smaller as better). The right graph zooms in the red-dashed-box region on the left, containing models in our VSR ablation study.

While the visual results discussed in previous chapters provide a first indicator of the quality TecoGAN achieves, quantitative evaluations are crucial for automated qualification across larger numbers of samples. Below we consider both VSR and UVT tasks, but focus more on the former task where ground-truth data is available. We evaluate models discussed above using established spatial metrics. Then, we motivate and propose two novel temporal metrics to quantify temporal coherence. A visual summary of the spatial and temporal metrics on the Vid4 data-set is shown in Fig. 7.1, where TecoGAN outperforms previous methods spatio-temporally. In addition to the spatio-temporal metric evaluations, we conduct user studies and the resulting scores confirm the conclusion of evaluations.

### 7.1 Evaluations Using Spatial Metrics

For evaluating images, Blau and Michaeli [105] demonstrated that there is an inherent trade-off between the perceptual quality and the distortion measured with vector norms or low-level structures such as PSNR and SSIM. On the other hand, metrics based on deep feature maps such as LPIPS [104] can capture more semantic similarities. Therefore, we evaluate all VSR methods discussed above with the purely spatial metrics PSNR together with the human-calibrated LPIPS metric. While higher PSNR values indicate a better pixel-wise accuracy, lower LPIPS values represent better perceptual quality and closer semantic similarity. Note that both metrics are agnostic to changes over time, and hence do not suffice to fully evaluate video data.

**Table 7.1:** Metrics evaluated for the VSR Vid4 scenes.

PSNR $\uparrow$	BIC	ENet	FRVSR	DUF	TecoGAN	TecoGAN $^{\ominus}$	DsOnly	DsDt	DsDtPP	
calendar	20.27	19.85	23.86	24.07	23.21	23.35	22.23	22.76	22.95	
foliage	23.57	21.15	26.35	26.45	24.26	25.13	22.33	22.73	25.00	
city	24.82	23.36	27.71	28.25	26.78	26.94	25.86	26.52	27.03	
walk	25.84	24.90	29.56	30.58	28.11	28.14	26.49	27.37	28.14	
average	23.66	22.31	26.91	<b>27.38</b>	25.57	25.89	24.14	24.75	25.77	
LPIPS $\downarrow\times 10$	BIC	ENet	FRVSR	DUF	TecoGAN	TecoGAN $^{\ominus}$	DsOnly	DsDt	DsDtPP	
calendar	5.935	2.191	2.989	3.086	1.511	2.142	1.532	2.111	2.112	
foliage	5.338	2.663	3.242	3.492	1.902	1.984	2.113	2.092	1.902	
city	5.451	3.431	2.429	2.447	2.084	1.940	2.120	1.889	1.989	
walk	3.655	1.794	1.374	1.380	1.106	1.011	1.215	1.057	1.051	
average	5.036	2.458	2.506	2.607	<b>1.623</b>	1.743	1.727	1.770	1.733	
tOF $\downarrow\times 10$	BIC	ENet	FRVSR	DUF	TecoGAN	TecoGAN $^{\ominus}$	DsOnly	DsDt	DsDtPP	
calendar	4.956	3.450	1.537	1.134	1.342	1.403	1.609	1.683	1.583	
foliage	4.922	3.775	1.489	1.356	1.238	1.444	1.543	1.562	1.373	
city	7.967	6.225	2.992	1.724	2.612	2.905	2.920	2.936	3.062	
walk	5.150	3.203	2.569	2.127	2.571	2.765	2.745	2.796	2.649	
average	5.578	4.009	2.090	<b>1.588</b>	1.897	2.082	2.157	2.198	2.103	
tLP $\downarrow\times 100$	BIC	ENet	FRVSR	DUF	TecoGAN	TecoGAN $^{\ominus}$	DsOnly	DsDt	DsDtPP	
calendar	3.258	2.957	1.067	1.603	0.165	1.087	0.872	0.764	0.670	
foliage	2.434	6.372	1.644	2.034	0.894	0.740	3.422	0.493	0.454	
city	2.193	7.953	0.752	1.399	0.974	0.347	2.660	0.490	0.140	
walk	0.851	2.729	0.286	0.307	0.653	0.635	1.596	0.697	0.613	
average	2.144	4.848	0.957	1.329	<b>0.668</b>	0.718	2.160	0.614	0.489	
T-diff $\downarrow\times 100$	BIC	ENet	FRVSR	DUF	TecoGAN	TecoGAN $^{\ominus}$	DsOnly	DsDt	DsDtPP	GT
calendar	2.271	9.153	3.212	2.750	4.663	3.496	6.287	4.347	4.167	6.478
foliage	3.745	11.997	3.478	3.115	5.674	4.179	8.961	6.068	4.548	4.396
city	1.974	7.788	2.452	2.244	3.528	2.965	4.929	3.525	2.991	4.282
walk	4.101	7.576	5.028	4.687	5.460	5.234	6.454	5.714	5.305	5.525
average	3.152	9.281	3.648	3.298	4.961	4.076	6.852	5.071	4.369	<b>5.184</b>

Evaluation results on the Vid4 scenes [133] are shown on the top of Table 7.1 and LPIPS scores are used as the values of the X-axis in Fig. 7.1. Specifically, trained with direct vector norms losses, FRVSR and DUF achieve high PSNR scores. However, the undesirable smoothing induced by these losses manifests themselves in larger LPIPS distances. ENet, on the other hand, with no information from neighboring frames, yields the lowest PSNR and achieves an LPIPS score that is only slightly better than DUF and FRVSR. The TecoGAN model with adversarial training achieves an excellent LPIPS score, with a PSNR decrease of less than 2dB over DUF which has twice the model size of ours. This is very reasonable, since PSNR and perceptual quality were shown to be anti-correlated [105], especially in regions where PSNR is very high. Based on good perceptual quality and reasonable pixel-wise accuracy, TecoGAN outperforms all other methods by more than 40% for LPIPS.

So far, we compared the VSR TecoGAN model with published models of ENet, DUF and FRVSR. All these methods are trained and evaluated on LR images generated from Gaussian-blur and down-sampling steps. However, we have not directly compared to methods from the NTIRE19 VSR challenges [134], as these models are trained on linear-

**Table 7.2:** Metrics evaluated for VSR of ToS scenes.

PSNR↑	BIC	ENet	FRVSR	DUF	TecoGAN	tOF ↓×10	BIC	ENet	FRVSR	DUF	TecoGAN
room	26.90	25.22	29.80	30.85	29.31	room	1.735	1.625	0.861	0.901	0.737
bridge	28.34	26.40	32.56	33.02	30.81	bridge	5.485	4.037	1.614	1.348	1.492
face	33.75	32.17	39.94	40.23	38.60	face	4.302	2.255	1.782	1.577	1.667
average	29.58	27.82	34.04	<b>34.60</b>	32.75	average	4.110	2.845	1.460	<b>1.296</b>	1.340
LPIPS ↓×10	BIC	ENet	FRVSR	DUF	TecoGAN	tLP ↓×100	BIC	ENet	FRVSR	DUF	TecoGAN
room	5.167	2.427	1.917	1.987	1.358	room	1.320	2.491	0.366	0.307	0.590
bridge	4.897	2.807	1.761	1.684	1.263	bridge	2.237	6.241	0.821	0.526	0.912
face	2.241	1.784	0.586	0.517	0.590	face	1.270	1.613	0.290	0.314	0.379
average	4.169	2.395	1.449	1.414	<b>1.086</b>	average	1.696	3.827	0.537	<b>0.403</b>	0.664

interpolated LR images. When applied to inputs with linear-interpolation, the winning method, EDVR [92], produces results on par with DUF and FRVSR with filtered, down-sampled inputs: On the Vid4 dataset, EDVR gets a PSNR of 27.35 and an LPIPS of 0.233, while the DUF method has a PSNR of 27.38 and an LPIPS of 0.261. Thus, the performance of EDVR is close to DUF, which our TecoGAN model clearly outperforms.

## 7.2 Temporal Metrics and Evaluations

For both VSR and UVT, evaluating temporal coherence without ground-truth motion is very challenging. While traditional temporal metrics based on vector norm differences of warped frames, e.g. T-diff =  $\|g_t - W(g_{t-1}, v_t)\|_1$  [98], can be easily deceived by very blurry results, e.g. bi-cubic interpolated ones, we propose to use a tandem of two new metrics, tOF and tLP, to measure the consistence over time. tOF measures the pixel-wise difference of motions estimated from sequences, and tLP measures perceptual changes over time using deep feature maps:

$$\begin{aligned} \text{tOF} &= \|OF(b_{t-1}, b_t) - OF(g_{t-1}, g_t)\|_1 \quad \text{and} \\ \text{tLP} &= \|LP(b_{t-1}, b_t) - LP(g_{t-1}, g_t)\|_1. \end{aligned} \quad (7.1)$$

$OF$  represents an optical flow estimation with Lucas-Kanade [135] and  $LP$  is the perceptual LPIPS metric. In tLP, the behavior of the reference is considered, as natural videos exhibit a certain degree of change over time. In conjunction, both pixel-wise differences and perceptual changes are crucial for quantifying realistic temporal coherence. While they could be combined into a single score, we list both measurements separately, as their relative importance could vary in different application settings. While tLP scores are used as values of Y-axis in Fig. 7.1, the tOF is displayed by bubble sizes.

For comparison, the T-diff numbers are given at the bottom of Table 7.1. Due to its local nature, is easily deceived by blurry method and can not correlate well with visual assessments of coherence. By using the proposed metrics, i.e. measuring the pixel-wise motion differences using tOF together with the perceptual changes over time using tLP, a more nuanced evaluation can be achieved, as shown for the VSR task in the middle of Table 7.1.

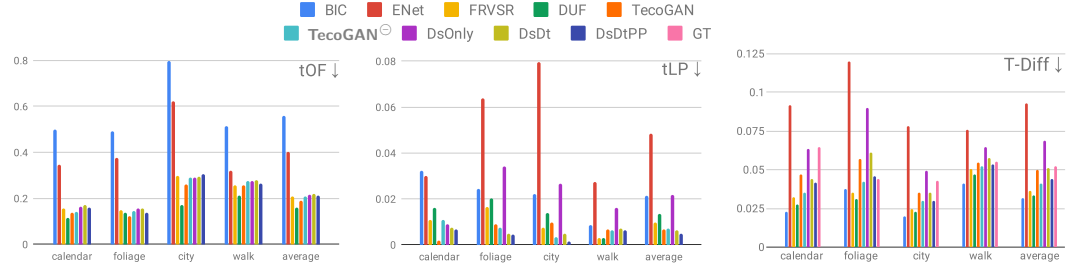


Figure 7.2: Bar graphs of temporal metrics for Vid4.

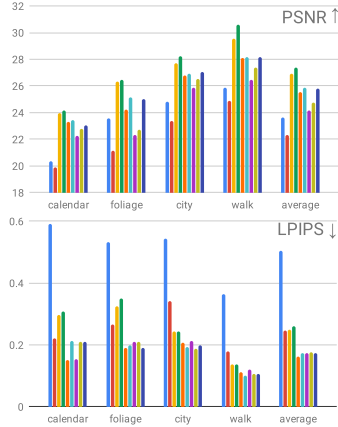


Figure 7.3: Vid4 spatial metrics.

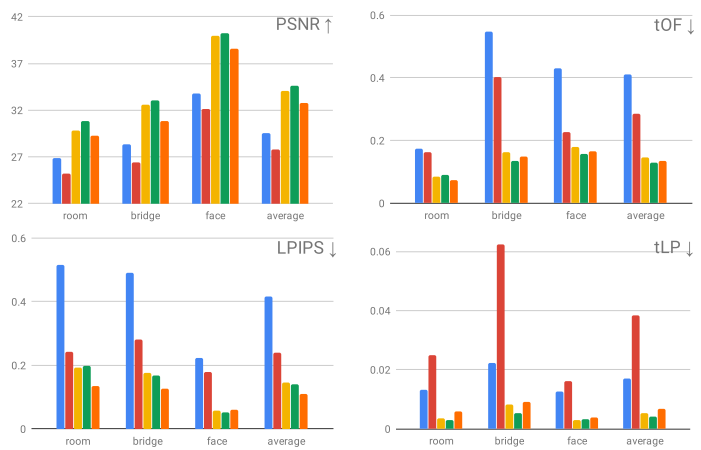


Figure 7.4: Metrics for ToS.

Not surprisingly, the results of ENet show larger errors for all metrics due to their strongly flickering content. Bi-cubic up-sampling, DUF, and FRVSR achieve very low T-diff errors due to their smooth results, representing an easy, but undesirable avenue for achieving coherency. However, the overly smooth changes of the former two are identified by the tLP scores. While our DsOnly model generates sharper results at the expense of temporal coherence, it still outperforms ENet there. By adding temporal information to discriminators, our DsDt, DsDtPP, TecoGAN<sup>ominus</sup> and TecoGAN improve in terms of temporal metrics.

In conclusion, all temporal adversarial models outperform spatial adversarial ones, and the full TecoGAN model performs very well: With a large amount of spatial detail, it still achieves good temporal coherence, on par with non-adversarial methods such as DUF and FRVSR. As shown in Fig. 7.1, with good spatial and temporal scores, TecoGAN model locates at the bottom left part of the figure with a small bubble size. These results are also visualized in Fig. 7.2 and 7.3.

Besides the previously evaluated the Vid4 dataset, we also get similar evaluation results on the *Tears of Steel* data-sets (room, bridge, and face, in the following referred to as *ToS* scenes) and corresponding results are shown in Table 7.2 and Fig. 7.4. In all tests, we follow the procedures of previous work [93, 90] to make the outputs of all methods comparable, i.e., for all result images, we first exclude spatial borders with a

**Table 7.3:** For the Obama&Trump dataset, the averaged tLP and tOF evaluations closely correspond to our user studies. The table below summarizes user preferences as Bradley-Terry scores. Details are given in Sec. 7.3.

UVT scenes	Trump→Obama		Obama→Trump		AVG		User Studies ↑, ref. to	
metrics	tLP↓	tOF↓	tLP↓	tOF↓	tLP↓	tOF↓	original input	arbitrary target
CycleGAN	0.0176	0.7727	0.0277	1.1841	0.0234	0.9784	0.0	0.0
RecycleGAN	<b>0.0111</b>	0.8705	0.0248	1.1237	0.0179	0.9971	0.994	0.202
TecoGAN	0.0120	<b>0.6155</b>	<b>0.0191</b>	<b>0.7670</b>	<b>0.0156</b>	<b>0.6913</b>	<b>1.817</b>	<b>0.822</b>

distance of 8 pixels to the image sides, then further shrink borders such that the LR input image is divisible by 8 and for spatial metrics, we ignore the first two and the last two frames, while for temporal metrics, we ignore first three and last two frames, as an additional previous frame is required for inference.

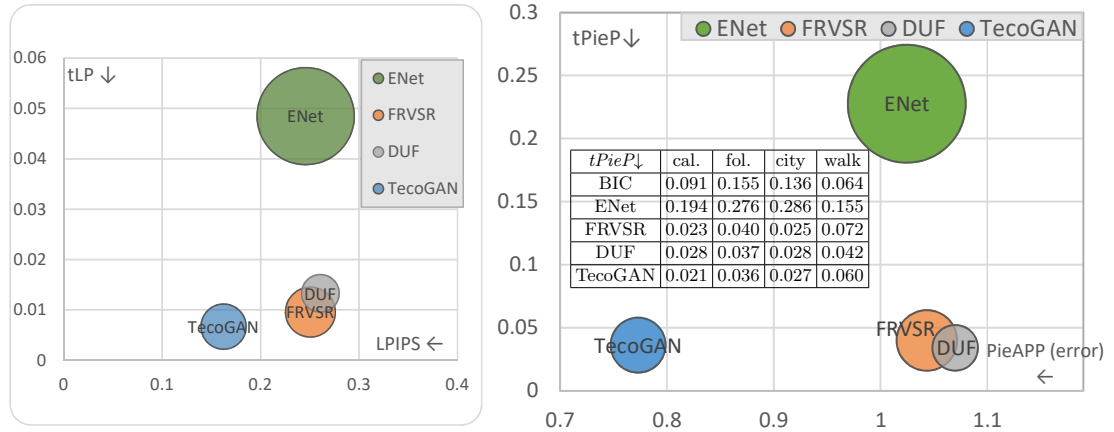
For the UVT tasks, where no ground-truth data is available, we can still evaluate tOF and tLP metrics by comparing the motion and the perceptual changes of the output data w.r.t. the ones from the input data, i.e.,

$$\begin{aligned} \text{tOF} &= \left\| OF(a_{t-1}, a_t) - OF(g_{t-1}^{a \rightarrow b}, g_t^{a \rightarrow b}) \right\|_1 \quad \text{and} \\ \text{tLP} &= \left\| LP(a_{t-1}, a_t) - LP(g_{t-1}^{a \rightarrow b}, g_t^{a \rightarrow b}) \right\|_1. \end{aligned} \quad (7.2)$$

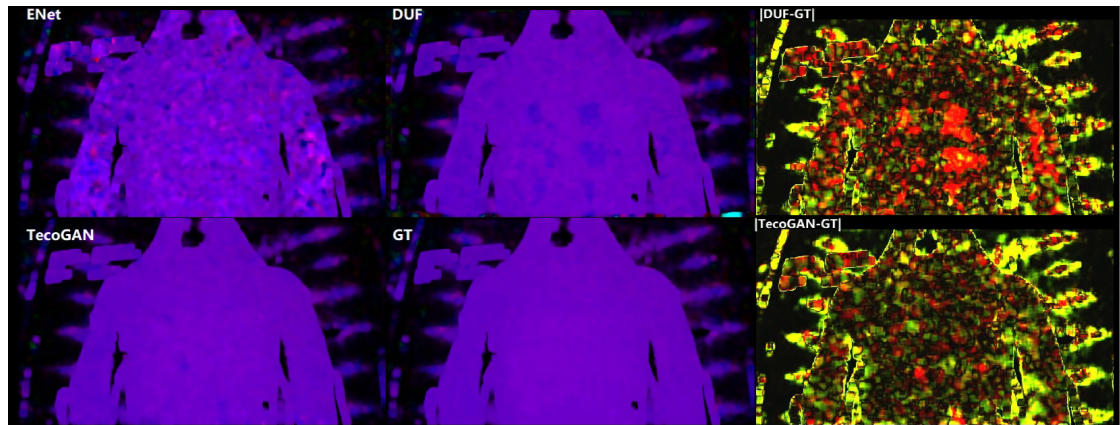
With sharp spatial features and coherent motion, TecoGAN outperforms previous work on the Obama&Trump dataset, as shown in Table 7.3, although it is worth pointing out that tOF is less informative in this case, as the motion in the target domain is not necessarily pixel-wise aligned with the input.

Similar to spatial evaluations where both pixel-wise metrics (e.g. PSNR) and perceptual metrics (e.g. LPIPS) are considered, tOF and tLP are proposed to describe different aspects for the temporal coherence. Specifically, we found that our formulation of tLP is a general concept that can work reliably with different perceptual metrics: When repeating the tLP evaluation with the PieAPP metric [136] instead of  $LP$ , i.e.,  $tPieP = \|f(b_{t-1}, b_t) - f(g_{t-1}, g_t)\|_1$ , where  $f(\cdot)$  indicates the perceptual error function of PieAPP, we get close to identical results, as shown in Fig. 7.5. The conclusions from  $tPieP$  also closely match the LPIPS-based evaluation: our network architecture can generate realistic and temporally coherent detail, and the metrics we propose allow for a stable, automated evaluation of the temporal perception of a generated video sequence. On the other hand, as a pixel-wise metric, tOF is more robust than T-diff as it compares motions instead of image content. In Fig. 7.6, we visualize the motion difference and it can well reflect the visual inconsistencies.

While spatial metrics are not sufficient to evaluate video data, good temporal metrics should be able to capture user preferences more, since temporal evolution is part of the core information in sequential data-sets. Thus, we conduct user studies as perceptual evaluations for both VSR and UVT tasks, in addition to the spatial metrics and temporal metrics above. Across all of them, we find that the majority of the participants considered TecoGAN results to be closest to the ground truth. The right column of



**Figure 7.5:** On the right, we show tables and visualization of perceptual metrics computed with PieAPP [136] (instead of LPIPS used in Fig. 7.1 previously, which is repeated on the left) on ENet, FRVSR, DUF and TecoGAN for the VSR of Vid4. Bubble size indicates the tOF score.

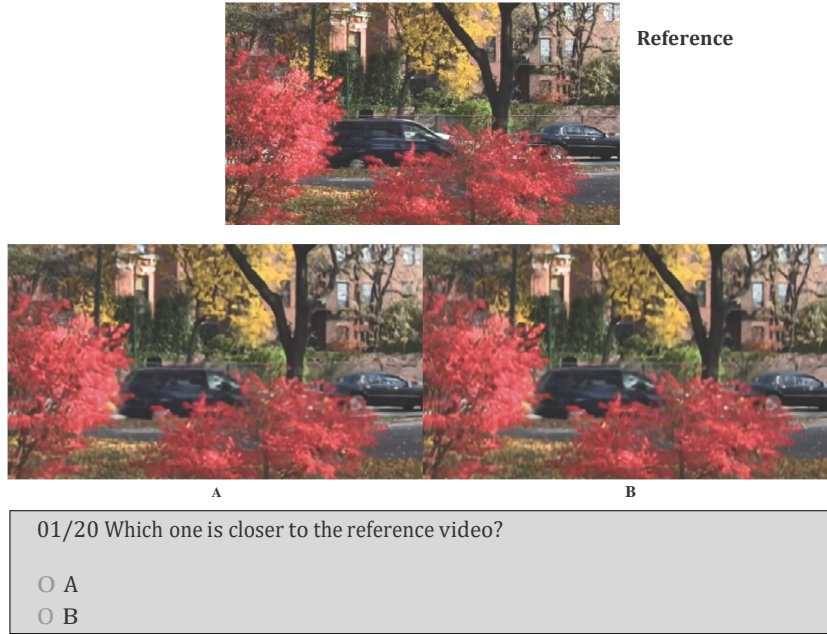


**Figure 7.6:** Optical flow (left two columns) and tOF (right column) visualization of the armor scene. While the optical flow of ENet is very noisy, the results of DUF and TecoGAN are closer to the ground truth (GT). The per-pixel differences shown on the right indicate that DUF has larger errors on the chest. Values are shown in HSV color space, with H representing directions, S being a constant one, and V representing the  $l_1$  vector norm.

Table 7.3 shows user-study scores for UVT tasks. More details about user studies and perceptual evaluation for VSR tasks can be found in Sec. 7.3.

### 7.3 User Studies

We conducted several user studies for the VSR task comparing five different methods: bi-cubic interpolation, ENet, FRVSR, DUF and TecoGAN. The established 2AFC design [137, 138] is applied, i.e., participants have a pair-wise choice, with the ground-truth



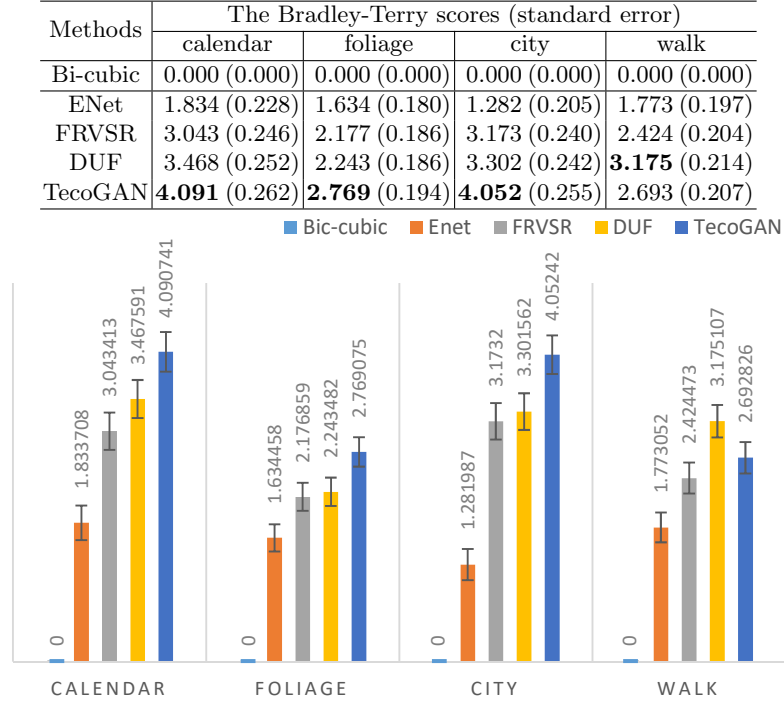
**Figure 7.7:** A sample setup of user study.

video shown as reference. One example setup can be seen in Fig. 7.7. The videos are synchronized and looped until participants make the final decision. With no control to stop videos, users cannot stop or influence the playback, and hence can focus more on the whole video, instead of specific spatial details. Videos positions (left/A or right/B) are randomized.

After collecting 1000 votes from 50 users for every scene, i.e. twice for all possible pairs ( $5 \times 4/2 = 10$  pairs), we follow common procedure and compute scores for all models with the Bradley-Terry model (1952). The outcomes for the Vid4 scenes can be seen in Fig. 7.8 and overall scores are listed in Table 7.4.

From the Bradley-Terry scores for the Vid4 scenes we can see that the TecoGAN model performs very well, and achieves the first place in three cases, as well as a second place in the walk scene. The latter is most likely caused by the overall slightly smoother images of the walk scene, in conjunction with the presence of several human faces, where our model can lead to the generation of unexpected details. However, overall the user study shows that users preferred the TecoGAN output over the other two deep-learning methods with a 63.5% probability.

By comparing the user study results and the metric breakdowns shown in Table 7.1 and Table 7.3, we find our metrics to reliably capture the human temporal perception. In Table 7.1, while TecoGAN achieves spatial (LPIPS) improvements in all scenes, DUF and FRVSR are not far behind in the walk scene. In terms of temporal metrics tOF and tLP, TecoGAN achieves similar or lower scores compared to FRVSR and DUF for calendar, foliage and city scenes. The lower performance of our model for the walk scene is likewise captured by higher tOF and tLP scores. Overall, the metrics confirm the



**Figure 7.8:** Tables and bar graphs of Bradley-Terry scores and standard errors for Vid4 VSR.

performance of our TecoGAN approach and match the results of the user studies, which indicate that our proposed temporal metrics successfully capture important temporal aspects of human perception.

For UVT tasks which have no ground-truth data, we carried out two sets of user studies: One uses an arbitrary sample from the target domain as the reference and the other uses the actual input from the source domain as the reference. On the Obama&Trump data-sets, we evaluate results from CycleGAN, RecycleGAN, and TecoGAN following the same modality, i.e. a 2AFC design with 50 users for each run. E.g., on the top left of Fig. 7.9, users evaluate the generated Obama in reference with the input Trump on the y-axis, while an arbitrary Obama video is shown as the reference on the x-axis. Ultimately, the Y-axis is more important than the X-axis as it indicates whether the translated result preserves the original expression. A consistent ranking of TecoGAN > RecycleGAN > CycleGAN is shown on the Y-axis with clear separations, i.e. standard errors don't overlap. The X-axis indicates whether the inferred result matches the general spatio-temporal content of the target domain. Our TecoGAN model also receives the highest scores here, although the responses are slightly more spread out. On the bottom of Fig. 7.9, we summarize both studies in a single graph highlighting that the TecoGAN model is consistently preferred by the participants of our user studies.

Fig. 7.10 shows user-study scores along X-axis and normalized scores of tLP and tOF are added up and displayed along Y-axis. From this figure, we can see that evaluations using proposed temporal metrics correlate well with user studies.



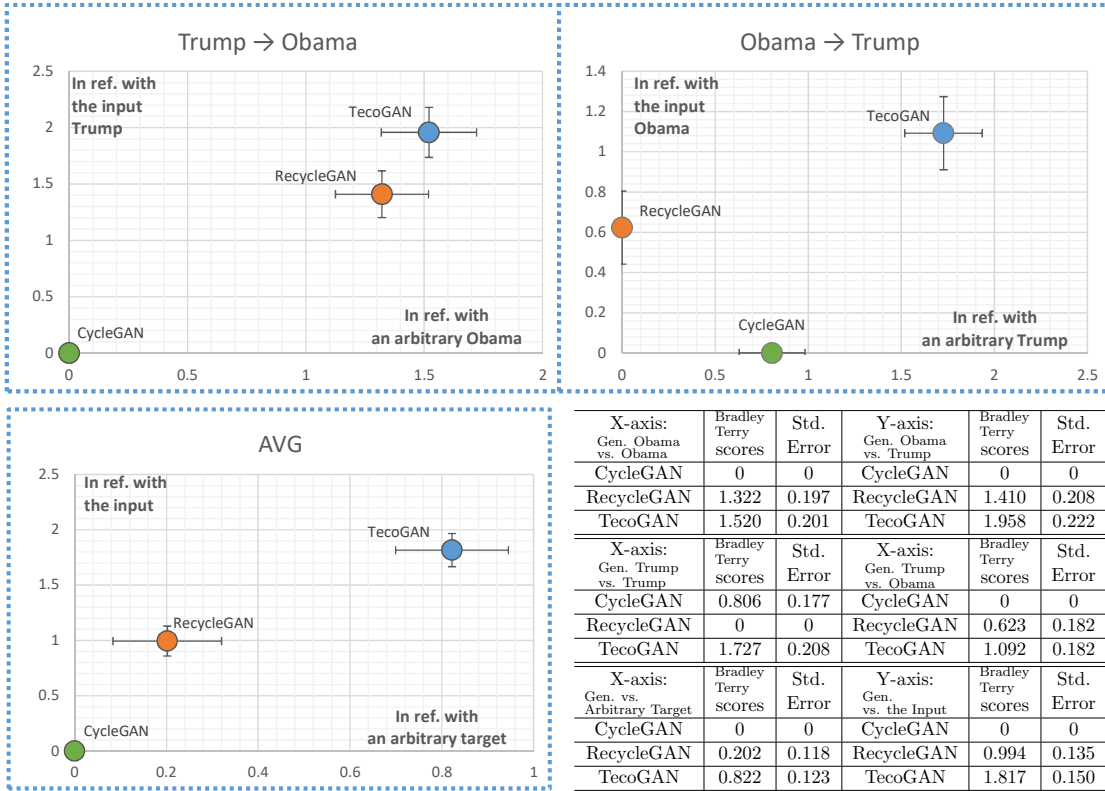


Figure 7.9: Tables and graphs of Bradley-Terry scores and standard errors for Obama&Trump UVT.

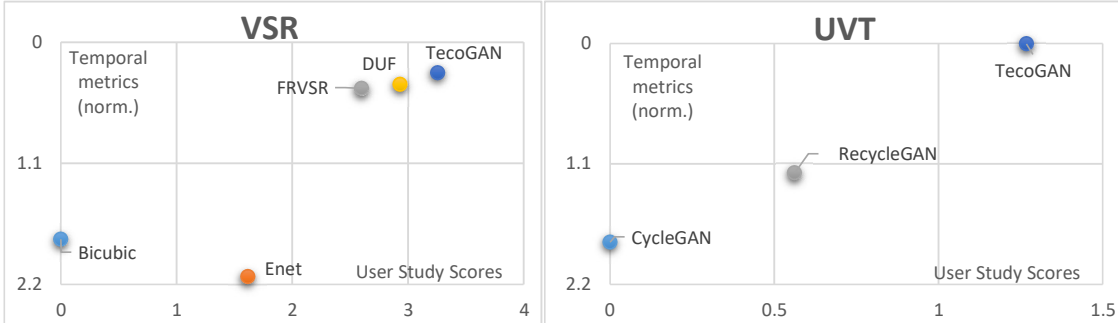


Figure 7.10: Evaluations using proposed temporal metrics correlate well to the user-study scores for both VSR and UVT tasks.

## 7.4 Discussion and Conclusions

Although temporal metrics and perceptual scores correlate well with each other in our tests, temporal metrics can still trivially be reduced for blurry image content. Thus, we found it important to evaluate results with a combination of spatial and temporal metrics for an automatic qualification. As can be seen in Fig. 7.1 at the beginning

**Table 7.4:** Averaged VSR metric evaluations for the *Vid4* data set with the following metrics, PSNR: pixel-wise accuracy. LPIPS (AlexNet): perceptual distance to the ground truth. T-diff: pixel-wise differences of warped frames. tOF: pixel-wise distance of estimated motions. tLP: perceptual distance between consecutive frames. User study: Bradley-Terry scores [139]. Performance is averaged over 500 images up-scaled from 320x134 to 1280x536.

Methods	PSNR $\uparrow$	LPIPS $\downarrow$ $\times 10$	T-diff $\downarrow$ $\times 100$	tOF $\downarrow$ $\times 10$	tLP $\downarrow$ $\times 100$	User Study $\uparrow$	Model Size(M) $\downarrow$	Processing Time(ms/frame) $\downarrow$
DsOnly	24.14	1.727	6.852	2.157	2.160	-	-	-
DsDt	24.75	1.770	5.071	2.198	0.614	-	-	-
DsDtPP	25.77	1.733	4.369	2.103	0.489	-	-	-
TecoGAN <sup>⊙</sup>	25.89	1.743	4.076	2.082	0.718	-	0.8(G)+1.7(F)	37.07
<b>TecoGAN</b>	25.57	<b>1.623</b>	4.961	1.897	<b>0.668</b>	<b>3.258</b>	1.3(G)+1.7(F)	41.92
ENet	22.31	2.458	9.281	4.009	4.848	1.616	-	-
FRVSR	26.91	2.506	3.648	2.090	0.957	2.600	0.8(SRNet)+1.7(F)	36.95
DUF	<b>27.38</b>	2.607	3.298	<b>1.588</b>	1.329	2.933	6.2	942.21
Bi-cubic	23.66	5.036	3.152	5.578	2.144	0.0	-	-

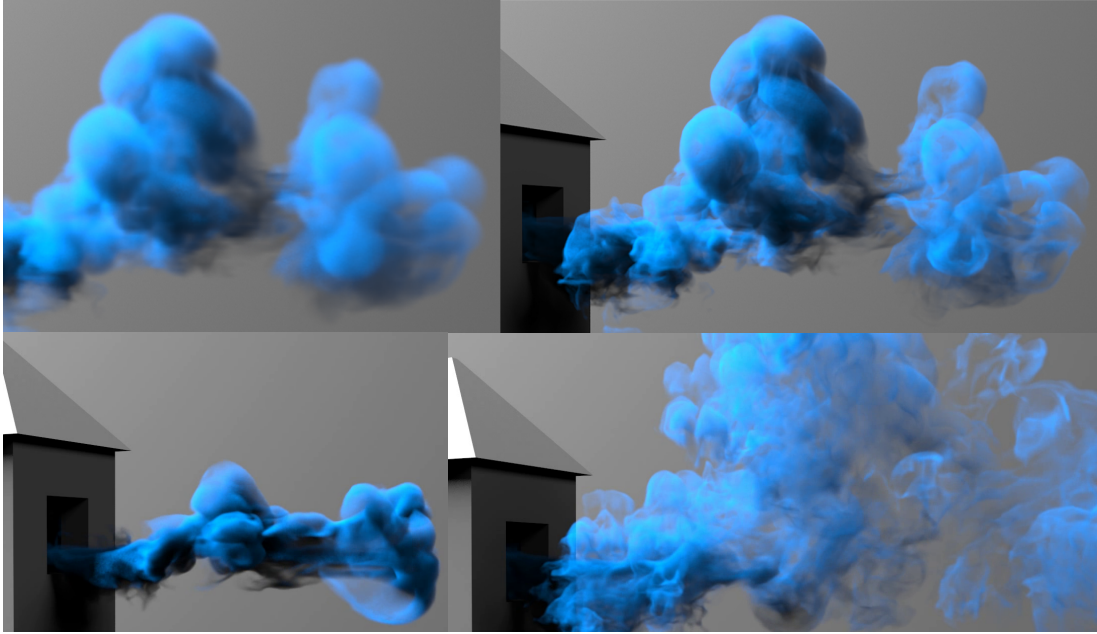
of this chapter, TecoGAN clearly outperforms previous methods on the Vid4 data-set considering both the spatial and temporal quality. Interestingly, all TecoGAN variants except the DsOnly model are located close to each other in the region highlighted with a red dashed box. This indicates that the spatio-temporal adversarial training offers better learning objectives and is the key to temporally coherent results with good spatial details. In Table 7.4, we summarize all VSR methods using the spatial and temporal metrics, user-study scores, model sizes in terms of weight numbers, and the performance. Using relatively short processing time, TecoGAN manages to generate good results in terms of temporal coherence and perceptual quality.

Based on established spatial metrics and proposed temporal metrics, we have evaluated the VSR and UVT results discussed in the previous chapters. These evaluations are consistent and match with the user studies. Given that perceptual metrics are already widely used for image evaluations, we believe it is the right time to consider perceptual changes in temporal evaluations, as we did with our proposed temporal coherence metrics. Although not perfect, they are not as easily deceived as simpler metrics. While there is still a large room to improve, we anticipate that these metrics will provide useful insights into reliable temporal evaluations for sequential data-sets.

## **Part IV**

# **Conclusions of Detail Synthesis for Sequential Data**





*“La vie est mouvement.”*

*-- Voltaire*

## 8 Coherent Details for Sequential Data

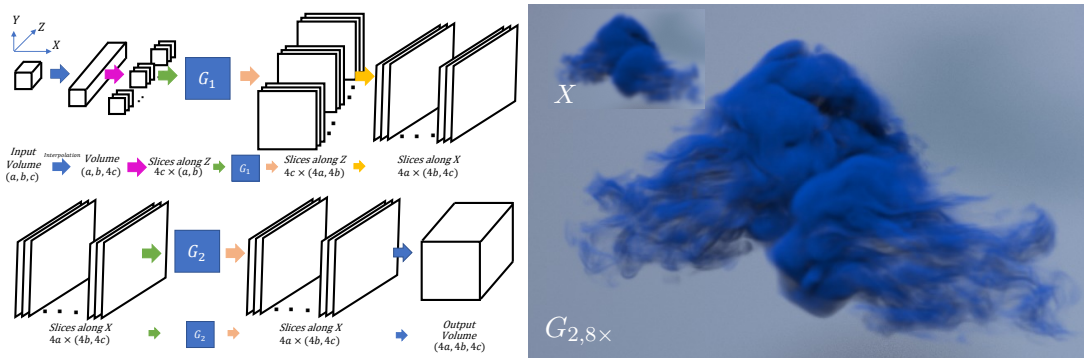
While it is not easy to add proper details to a static object, it is much harder to generate realistic details that move coherently with dynamic objects. In Part II, we have presented a smoke synthesis method based on discriminative learning. In order to achieve coherent details, we first select details according to fluid similarity considering both density and motion information and then we move them using patches that deform with the underlying fluid. In Part III, we propose temporal self-supervisions for generative learning methods and produce coherent video results. Compared to the generative methods, a discriminative model is less flexible, therefore requires several other modules to cooperate together. Generative learning, on the other hand, is more applicable to similar tasks with modifications. In this chapter, we look into the generation of different sequential data-sets using generative learning methods.

Besides the aforementioned discriminative method, the goal of detail synthesis for fluids can be likewise achieved using a fluid super-resolution method based on generative learning. We now consider the Eulerian representation because it can generalize more easily to other sequential data. Allowing for local-wise operations e.g. difference approx-

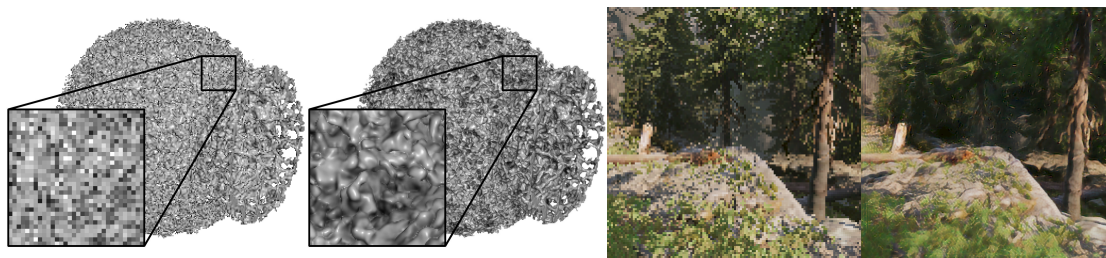
imations of derivatives and convolutional filters, the Eulerian grid is a good candidate to combine physics-based simulations with neural-network methods. In practice, memory becomes the bottleneck for volumetric super-resolution of fluid sequences. For VSR, a frame-recurrent generator is necessary because a image only contain a 2D projection of the 3D space and information from the current frame is usually not enough due to occlusions. Thankfully, for fluid simulations, we can input full volumetric density and velocity fields of the current time-step, while historical information is less important. The motion estimation network  $F$  can also be replaced by pre-computed velocity fields. Thus, by using a single-frame generator, a spatial discriminator, and a temporal discriminator, the spatio-temporal adversarial training produces a coherent generative model addressing the SR problem for fluid flows. Processing every frame of the LR smoke separately, a sequence of HR smoke volumes can be generated in parallel. On the top left of the teaser image, we show one frame of the rendered LR input. Three frames of the rendered results are shown on the bottom left, top right and bottom right.

While the method above managed to increase the resolution of 3D fluids by a factor of 4, larger factors such as 8 are very hard since the number of weights required in a neural network makes the adversarial training extremely difficult. Instead of fulfilling the goal within one pass, we propose to decompose the problem of generating a Cartesian field function into two orthogonal passes. As shown on the left of Fig. 8.1, two separate GANs are used, where the first one up-scales slices in XY-plane and the second one refines the whole volume along the Z-axis. Since these sub-problems can be learned more efficiently, a generative model with a SR factor of eight is achieved for the first time. Considering all dimensions, the multi-pass GAN can increase the degrees of freedom by 512. A result is shown on the right of Fig. 8.1. From an input in resolution of  $50^3$ , a coherent HR smoke is generated in resolution of  $400^3$ .

Besides fluids and videos, we have likewise studied the generation of rendering sequences for iso-surfaces and strongly aliased video games. Restricted to iso-surfaces, we found that regular losses behave reasonably well and adversarial training is not necessary since the multi-modality is reduced. Instead of working on RGB colors, a SR of the nor-



**Figure 8.1:** A multi-pass GAN. The workflow on the left is drawn for a super-resolution factor of 4. With the help of progressive growing [75], we can increase the SR factor to 8 and a result is shown on the right. The input on the top left corner has a resolution of  $50^3$ .



**Figure 8.2:** SR for rendering sequences of iso-surfaces (the left two) and video games (the right two). From left to right, we show a direct illumination of a LR iso-surface as input, the generated HR iso-surface rendering with AO, a strongly aliased video game input, and the generated SR result.

mal and depth fields in image space leads to a large improvement on the results. Along with the SR of normal and depth fields for direct illuminations, the network managed to learn the ambient occlusion (AO) simultaneously from the geometry information for global illumination, even though the LR AO is not provided as an input. Together, the network has demonstrated its potential for real-time rendering applications, e.g. remote visualization tasks. For video games, the computation required for photo-realistic rendering of complex scenes is still blocking remote applications including cloud gaming. Taking strongly aliased LR images from a rasterization-based real-time renderer as inputs, it is very difficult to generate coherent super-resolution results with realistic details. While there is a large variety of objects in games, the fact that only one sample is traced for one pixel results in increased multi-modality. Based on the frame-recurrent generator and a spatio-temporal discriminator, we add recurrent connections on the internal state of the generator’s residual blocks. After warping, these depth-recurrent residuals allow the network to re-use deep features from historical frames and result in better perceptual quality and temporal coherence. We also found that it is better to remove the LR images as the conditional input for the discriminator. Otherwise, it is too difficult for the generator to find a coherent solution that can correlate with the noisy input well. Results of the two projects are shown Fig. 8.2.

Across videos, fluids, iso-surface and real-time renderings, we have presented generative learning methods to synthesize detailed and coherent sequential results. The exploration across different data domains allow us to understand the capability of spatio-temporal learning algorithms better. In the next chapter, we will conclude all methods presented in the dissertation, draw connections and discuss on promising future directions.







*“In an increasingly Virtual world, audience interaction, not content, is King!”*

*— Clyde DeSouza*

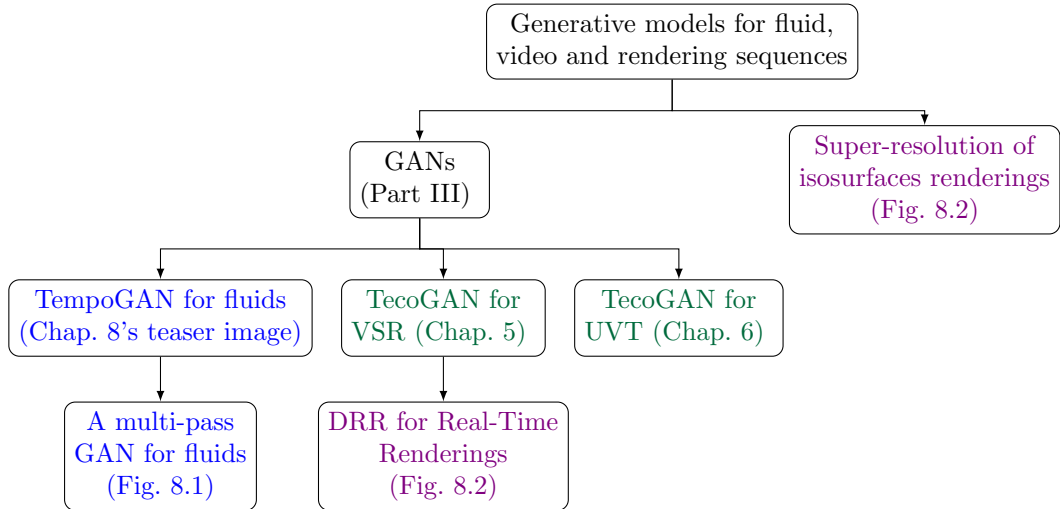
## 9 Conclusions and Future Work

In this dissertation, we focus on detail synthesis for sequential data-sets using deep-learning-based algorithms. In the following, we will summarize all methods discussed in previous chapters. Afterwards, limitations and future directions are presented.

### 9.1 Method Summary

In previous chapters, we have presented a discriminative-learning method for data-driven synthesis of smoke flows and several generative methods for fluids, videos and renderings. Based on supervised learning, a discriminative model usually requires smaller sets of weights and training data. When synthesizing details for fluids, the aforementioned discriminative method relies on other modules such as the deformable patch advection. This, however, makes it hard to generalize to other sequential data-sets. In our projects, we demonstrate that generative models can be successfully applied to different data-sets.

In Fig. 9.1, we summarize the aforementioned generative models. From these projects, we obtain the following practical experiences. While regular losses should be considered for specific data-sets, e.g. the iso-surface renderings, adversarial learning behaves better for data-sets with strong multi-modality. When using GANs for sequential data gener-



**Figure 9.1:** The connections of generative methods for fluids, videos and renderings, repeated from Fig. 1.4. Algorithms for videos, fluids and renderings are highlighted in green, blue, and purple respectively.

ation, spatio-temporal adversarial learning surpasses normal temporal losses based on low-level metrics: It can offer coherent solutions without reducing spatial realism. In this case, the input of a spatio-temporal discriminator should be carefully considered since the learning objectives are dynamically explored from them. In practice, we find that curriculum learning, progressive growing, and dynamic update rules contribute to stable GAN training. While curriculum learning helps to break down the overall learning objective into smaller sub-problems, the progressive growing strategy works similarly by introducing the spatial complexity step by step. By using dynamic update rules, generators and discriminators are updated differently in order to keep a good balance. While Heusel et al. propose to use different learning rates for generators and discriminators [107], we choose to update discriminators only when necessary. Details for this update rule is introduced in Appendix A.2.2. In general, our work demonstrates that deep learning algorithms have great potential in learning spatio-temporal representations. Given that deep learning has been extremely successful at learning complex static distributions, we anticipate more and more exploration on temporal evolution.

## 9.2 Future Work

To conclude, we have motivated and introduced the detail synthesis for computer graphics and vision tasks. Focusing on the sequential data-sets, we developed discriminative and generative learning methods for fluid and video synthesis. These methods successfully generate sharp details with coherent motions. By applying the temporally coherent GAN to the generation of fluid, video, and rendering sequences, we show that spatio-temporal learning is capable to represent sequential data with realistic temporal

evolution. We also studied images and video evaluation methods. With the proposed temporal metrics, we call attention to the perceptual evaluation of temporal changes. While achieving realistic results, we are aware of the limitations including unstable GAN training, sub-optimal results on samples different to the training data-set, and the large amount of resources required for training.

Despite these drawbacks, it is still amazing what deep learning methods have achieved. If we consider books as the beginning of a virtual world created by a human, deep-learning-based techniques in computer graphics and vision have started to bring visual content into the virtual world for everyone and they are greatly enriching the way we express and communicate. Along this direction, we still lack powerful tools to synthesize realistic sequential content according to users' intentions. While it is interesting to learn from large data-sets for synthesis and other computer graphic problems, one promising direction for vision tasks is to use prior knowledge such as physical rules to enhance the understanding. From both areas, we believe that combining the advantages of physical simulations and machine learning is a promising direction. In the future, we will continue working in this direction and conduct further research to solve many more challenging temporal synthesis problems, such as temporal super-resolution of sequential data-sets and physical simulations with intuitive user controls.

**Temporal Super-resolution** While we have been working on spatial super-resolution tasks for both fluid and video data, temporal super-resolution requires more understanding of the temporal evolution. In previous work, Super Slomo [140] focuses on the motion estimation between input video frames and assumes in-between frames are linear interpolations of warped frames. Up to now, how to learn the non-linear temporal evolution is still an open question. While adversarial learning is important for multi-modal problems like this, a spatio-temporal discriminator could be applied to achieve good performance on both spatial and temporal aspects. Still, there are large differences between different sequential data domains, e.g. fluid and video. It is beneficial to focusing on different data-sets to form a better understanding on general temporal learning problems.

**Conditional Generation for Fluid Motion** It is always a difficult task to synthesize fluid motion with considerations on user control as well as physically-based fluid behavior. While there are previous work using primal-dual optimization [141] and procedural turbulence [2], new technologies on differentiable tools for simulation [65, 66] and rendering [142] opens up new possibilities for intuitive user interactions including videos [143] and sketches [144, 145]. Besides convenient user controls, these technologies provide new connections between real-world captures and physics simulations, which could further our physical understandings of the captured natural phenomena from a data-driven perspective.

Overall, sequential data-sets are vital components of human lives. To understand and generate the temporal evolution represented in these data-sets, it is interesting and important to work at the interface between deep learning and physically-based rendering

and simulations. The results presented in this dissertation demonstrate that data-driven and deep-learning-based synthesis are promising directions. They provide powerful tools with state-of-the-art performances. With many more open problems beyond what have been discussed above, we hope that our methods can provide insights into a wide variety of sequential generation tasks.

# Bibliography

- [1] D. Sýkora, M. Ben-Chen, M. Čadík, B. Whited, and M. Simmons, “TexToons: Practical texture mapping for hand-drawn cartoon animations,” in *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, 2011, pp. 75–83.
- [2] T. Kim, N. Thuerey, D. James, and M. Gross, “Wavelet Turbulence for Fluid Simulation,” *ACM Trans. Graph.*, 2008.
- [3] Wikipedia, The Free Encyclopedia, *Live streaming*. [Online]. Available: [https://en.wikipedia.org/wiki/Live\\_streaming](https://en.wikipedia.org/wiki/Live_streaming).
- [4] Wikipedia, The Free Encyclopedia, *Cloud gaming*. [Online]. Available: [https://en.wikipedia.org/wiki/Cloud\\_gaming](https://en.wikipedia.org/wiki/Cloud_gaming).
- [5] Dronepicr@flickr, *Google Stadia Cloud Gaming*. [Online]. Available: <https://www.flickr.com/photos/132646954@N02/48605754611/>.
- [6] NASA, *Eyes on the Earth*. [Online]. Available: [https://climate.nasa.gov/climate\\_resources/110/interactive-eyes-on-the-earth/](https://climate.nasa.gov/climate_resources/110/interactive-eyes-on-the-earth/).
- [7] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, “Image analogies,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 2001, pp. 327–340.
- [8] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, “Texture optimization for example-based synthesis,” in *ACM Transactions on Graphics (ToG)*, ACM, vol. 24, 2005, pp. 795–802.
- [9] M. B. Nielsen, B. B. Christensen, N. B. Zafar, D. Roble, and K. Museth, “Guiding of smoke animations through variational coupling of simulations at different resolutions,” in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, 2009, pp. 217–226.
- [10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 2672–2680.
- [11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

- [12] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, and Others, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [14] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [15] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [16] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style,” *arXiv preprint arXiv:1508.06576*, 2015.
- [17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks,” in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.
- [18] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-To-Image Translation With Conditional Adversarial Networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [19] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” *arXiv preprint arXiv:1809.11096*, 2018.
- [20] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro, “Video-to-Video Synthesis,” in *Advances in Neural Information Processing Systems (NIPS)*, 2018.
- [21] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, “Deep fluids: A generative network for parameterized fluid simulations,” in *Computer Graphics Forum*, Wiley Online Library, vol. 38, 2019, pp. 59–70.
- [22] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh, “Recycle-gan: Unsupervised video retargeting,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 119–135.
- [23] Y. Xie, E. Franz, M. Chu, and N. Thuerey, “tempoGAN: A Temporally Coherent, Volumetric GAN for Super-Resolution Fluid Flow,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 95, 2018.
- [24] O. Jamriška, J. Fišer, P. Asente, J. Lu, E. Shechtman, and D. Sýkora, “LazyFluids: appearance transfer for fluid animations,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 92, 2015.
- [25] M.-L. Eckert, K. Um, and N. Thuerey, “ScalarFlow: A Large-Scale Volumetric Data Set of Real-world Scalar Transport Flows for Computer Animation and Machine Learning,” *ACM Transactions on Graphics*, vol. 38(6):239, 2019.
- [26] M. Kass and G. Miller, “Rapid, Stable Fluid Dynamics for Computer Graphics,” *ACM Transactions on Graphics (TOG)*, vol. 24, no. 4, pp. 49–55, 1990.

- [27] R. Bridson, *Fluid Simulation for Computer Graphics*. AK Peters/CRC Press, 2008.
- [28] M. Lentine, W. Zheng, and R. Fedkiw, “A novel algorithm for incompressible flow using only a coarse grid projection,” in *ACM Transactions on Graphics (TOG)*, ACM, vol. 29, 2010, p. 114.
- [29] R. Ando, N. Thuerey, and C. Wojtan, “A Dimension-reduced Pressure Solver for Liquid Simulations,” *Comput. Graph. Forum*, vol. 34, no. 2, p. 10, 2015.
- [30] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, “Accelerating eulerian fluid simulation with convolutional networks,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 3424–3433.
- [31] S. Wiewel, M. Becher, and N. Thuerey, “Latent Space Physics: Towards Learning the Temporal Evolution of Fluid Flow,” in *Computer Graphics Forum*, Wiley Online Library, vol. 38, 2019, pp. 71–82.
- [32] A. McAdams, E. Sifakis, and J. Teran, “A parallel multigrid Poisson solver for fluids simulation on large grids,” in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, 2010, pp. 65–74.
- [33] K. Museth, “VDB: High-resolution sparse volumes with dynamic topology,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, p. 27, 2013.
- [34] M. Aanjaneya, M. Gao, H. Liu, C. Batty, and E. Sifakis, “Power diagrams and sparse paged grids for high resolution adaptive liquids,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 140, 2017.
- [35] J. Stam, “Stable Fluids,” in *Siggraph*, vol. 99, 1999, pp. 121–128.
- [36] B. Kim, Y. Liu, I. Llamas, and J. R. Rossignac, “Flowfixer: Using bfec for fluid simulation,” Georgia Institute of Technology, Tech. Rep., 2005.
- [37] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac, “An unconditionally stable MacCormack method,” *Journal of Scientific Computing*, vol. 35, no. 2-3, pp. 350–371, 2008.
- [38] Y. Zhu and R. Bridson, “Animating Sand as a Fluid,” *ACM Trans. Graph.*, vol. 24, no. 3, pp. 965–972, Jul. 2005.
- [39] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, “The Affine Particle-in-Cell Method,” *ACM Trans. Graph.*, vol. 34, no. 4, Jul. 2015.
- [40] C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran, “A Polynomial Particle-in-Cell Method,” *ACM Trans. Graph.*, vol. 36, no. 6, Nov. 2017.
- [41] J. Zehnder, R. Narain, and B. Thomaszewski, “An Advection-Reflection Solver for Detail-Preserving Fluid Simulation,” *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018.

- [42] Z. Qu, X. Zhang, M. Gao, C. Jiang, and B. Chen, “Efficient and conservative fluids using bidirectional mapping,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.
- [43] R. Fedkiw, J. Stam, and H. W. Jensen, “Visual simulation of smoke,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 15–22.
- [44] A. Selle, N. Rasmussen, and R. Fedkiw, “A Vortex Particle Method for Smoke, Water and Explosions,” in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH ’05, Association for Computing Machinery, 2005, pp. 910–914.
- [45] T. Pfaff, N. Thuerey, A. Selle, and M. Gross, “Synthetic Turbulence Using Artificial Boundary Layers,” *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–10, 2009.
- [46] T. Pfaff, N. Thuerey, J. Cohen, S. Tariq, and M. Gross, “Scalable Fluid Simulation Using Anisotropic Turbulence Particles,” in *ACM SIGGRAPH Asia 2010 Papers*, ser. SIGGRAPH ASIA ’10, New York, NY, USA: Association for Computing Machinery, 2010.
- [47] T. Pfaff, N. Thuerey, and M. Gross, “Lagrangian Vortex Sheets for Animating Fluids,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 112, 2012.
- [48] R. Narain, J. Sewall, M. Carlson, and M. C. Lin, “Fast Animation of Turbulence Using Energy Transport and Procedural Synthesis,” *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5, article 166, 2008.
- [49] S. Sato, Y. Dobashi, T. Kim, and T. Nishita, “Example-Based Turbulence Style Transfer,” *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018.
- [50] J. J. Monaghan, “Smoothed particle hydrodynamics,” *Annual review of astronomy and astrophysics*, vol. 30, no. 1, pp. 543–574, 1992.
- [51] M. Becker and M. Teschner, “Weakly compressible SPH for free surface flows,” in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, 2007, pp. 209–217.
- [52] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner, “Implicit incompressible SPH,” *IEEE transactions on visualization and computer graphics*, vol. 20, no. 3, pp. 426–435, 2013.
- [53] N. Foster and D. Metaxas, “Realistic animation of liquids,” *Graphical models and image processing*, vol. 58, no. 5, pp. 471–483, 1996.
- [54] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle, “A Material Point Method for Snow Simulation,” *ACM Trans. Graph.*, vol. 32, no. 4, Jul. 2013.
- [55] C. Jiang, T. Gast, and J. Teran, “Anisotropic elastoplasticity for cloth, knit and hair frictional contact,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–14, 2017.
- [56] A. P. Tampubolon, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, and K. Museth, “Multi-species simulation of porous sand and water mixtures,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.



- [57] M. Ihmsen, N. Akinici, G. Akinici, and M. Teschner, “Unified spray, foam and air bubbles for particle-based fluids,” *The Visual Computer*, vol. 28, no. 6-8, pp. 669–677, 2012.
- [58] O. Mercier, C. Beauchemin, N. Thuerey, T. Kim, and D. Nowrouzezahrai, “Surface turbulence for particle-based liquid simulations,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–10, 2015.
- [59] F. Ferstl, R. Ando, C. Wojtan, R. Westermann, and N. Thuerey, “Narrow band FLIP for liquid simulations,” in *Computer Graphics Forum*, Wiley Online Library, vol. 35, 2016, pp. 225–232.
- [60] L. Ladický, S. Jeong, B. Solenthaler, M. Pollefeys, and M. Gross, “Data-driven fluid simulations using regression forests,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–9, 2015.
- [61] C. Yang, X. Yang, and X. Xiao, “Data-driven projection method in fluid simulation,” *Computer Animation and Virtual Worlds*, vol. 27, no. 3-4, pp. 415–424, 2016.
- [62] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, “Towards physics-informed deep learning for turbulent flow prediction,” *preprint arXiv:1911.08655*, 2019.
- [63] L. Prantl, B. Bonev, and N. Thuerey, “Generating liquid simulations with deformation aware neural networks,” in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=HyeGBj09Fm>.
- [64] P. Ma, Y. Tian, Z. Pan, B. Ren, and D. Manocha, “Fluid directed rigid body control using deep reinforcement learning,” *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018.
- [65] P. Holl, N. Thuerey, and V. Koltun, “Learning to control pdes with differentiable physics,” in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=HyeSin4FPB>.
- [66] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, “DiffTaichi: Differentiable programming for physical simulation,” in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=B1eB5xSFvr>.
- [67] K. Um, X. Hu, and N. Thuerey, “Liquid splash modeling with neural networks,” in *Computer Graphics Forum*, Wiley Online Library, vol. 37, 2018, pp. 171–182.
- [68] B. Kim, V. C. Azevedo, M. Gross, and B. Solenthaler, “Transport-Based Neural Style Transfer for Smoke Simulations,” *ACM Trans. Graph.*, vol. 38, no. 6, 2019.
- [69] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [70] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, USA, 2015, vol. 2018.

- [71] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. Paul Smolley, “Least squares generative adversarial networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802.
- [72] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in *Advances in neural information processing systems*, 2017, pp. 5767–5777.
- [73] A. Jolicoeur-Martineau, “The relativistic discriminator: a key element missing from standard GAN,” *arXiv preprint arXiv:1807.00734*, 2018.
- [74] I. Goodfellow, “NIPS 2016 tutorial: Generative adversarial networks,” *preprint arXiv:1701.00160*, 2016.
- [75] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *arXiv preprint arXiv:1710.10196*, 2017.
- [76] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.
- [77] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” *arXiv preprint arXiv:1805.08318*, 2018.
- [78] J. Kim, J. Kwon Lee, and K. Mu Lee, “Accurate image super-resolution using very deep convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.
- [79] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, “Deep laplacian pyramid networks for fast and accurate superresolution,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017, p. 5.
- [80] M. He, D. Chen, J. Liao, P. V. Sander, and L. Yuan, “Deep Exemplar-Based Colorization,” *ACM Trans. Graph.*, vol. 37, no. 4, 2018.
- [81] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and Others, “Photo-realistic single image super-resolution using a generative adversarial network,” *arXiv:1609.04802*, 2016.
- [82] O. Jamriška, Š. Sochorová, O. Texler, M. Lukáč, J. Fišer, J. Lu, E. Shechtman, and D. Sýkora, “Stylizing Video by Example,” *ACM Transactions on Graphics*, vol. 38, no. 4, 2019.
- [83] B. Wronski, I. Garcia-Dorado, M. Ernst, D. Kelly, M. Krainin, C.-K. Liang, M. Levoy, and P. Milanfar, “Handheld Multi-Frame Super-Resolution,” *ACM Trans. Graph.*, vol. 38, no. 4, 2019.
- [84] V. Sitzmann, S. Diamond, Y. Peng, X. Dun, S. Boyd, W. Heidrich, F. Heide, and G. Wetzstein, “End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–13, 2018.

- [85] B. Zhang, M. He, J. Liao, P. V. Sander, L. Yuan, A. Bermak, and D. Chen, “Deep Exemplar-based Video Colorization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8052–8061.
- [86] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, “Flownet: Learning optical flow with convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.
- [87] P. Liu, M. Lyu, I. King, and J. Xu, “Selflow: Self-supervised learning of optical flow,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4571–4580.
- [88] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.
- [89] J. Caballero, C. Ledig, A. P. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi, “Real-Time Video Super-Resolution with Spatio-Temporal Networks and Motion Compensation.,” in *CVPR*, vol. 1, 2017, p. 7.
- [90] M. S. M. Sajjadi, R. Vemulapalli, and M. Brown, “Frame-Recurrent Video Super-Resolution,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, 2018.
- [91] X. Zhu, H. Hu, S. Lin, and J. Dai, “Deformable convnets v2: More deformable, better results,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9308–9316.
- [92] X. Wang, K. C. K. Chan, K. Yu, C. Dong, and C. C. Loy, “EDVR: Video restoration with enhanced deformable convolutional networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.
- [93] Y. Jo, S. W. Oh, J. Kang, and S. J. Kim, “Deep Video Super-Resolution Network Using Dynamic Upsampling Filters Without Explicit Motion Compensation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3224–3232.
- [94] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia, “Detail-Revealing Deep Video Super-Resolution,” in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [95] D. Liu, Z. Wang, Y. Fan, X. Liu, Z. Wang, S. Chang, and T. Huang, “Robust video super-resolution with learned temporal dynamics,” in *Computer Vision (ICCV), 2017 IEEE International Conference on*, IEEE, 2017, pp. 2526–2534.
- [96] Y. Chen, Y. Pan, T. Yao, X. Tian, and T. Mei, “Mocycle-GAN: Unpaired Video-to-Video Translation,” *arXiv preprint arXiv:1908.09514*, 2019.
- [97] M. Ruder, A. Dosovitskiy, and T. Brox, “Artistic style transfer for videos,” in *German Conference on Pattern Recognition*, Springer, 2016, pp. 26–36.

- [98] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua, “Coherent online video style transfer,” in *Proc. Intl. Conf. Computer Vision (ICCV)*, 2017.
- [99] K. Park, S. Woo, D. Kim, D. Cho, and I. S. Kweon, “Preserving Semantic and Temporal Consistency for Unpaired Video-to-Video Translation,” *arXiv preprint arXiv:1908.07683*, 2019.
- [100] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [101] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment,” in *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Ieee, vol. 2, 2003, pp. 1398–1402.
- [102] A. Mittal, R. Soundararajan, and A. C. Bovik, “Making a “completely blind” image quality analyzer,” *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 209–212, 2012.
- [103] C. Ma, C.-Y. Yang, X. Yang, and M.-H. Yang, “Learning a no-reference quality metric for single-image super-resolution,” *Computer Vision and Image Understanding*, vol. 158, pp. 1–16, 2017.
- [104] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” *arXiv preprint*, 2018.
- [105] Y. Blau and T. Michaeli, “The perception-distortion tradeoff,” in *Proc. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, Utah, USA, 2018*, pp. 6228–6237.
- [106] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor, “The 2018 pirm challenge on perceptual image super-resolution,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.
- [107] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Advances in neural information processing systems*, 2017, pp. 6626–6637.
- [108] T. Unterthiner, S. van Steenkiste, K. Kurach, R. Marinier, M. Michalski, and S. Gelly, “Towards accurate generative models of video: A new metric & challenges,” *arXiv preprint arXiv:1812.01717*, 2018.
- [109] K. Um, X. Hu, B. Wang, and N. Thuerey, “Spot the difference: Accuracy of numerical simulations via the human visual system,” *preprint arXiv:1907.04179*, 2019.
- [110] G. Kohl, K. Um, and N. Thuerey, *Learning similarity metrics for numerical simulations*, 2020. [Online]. Available: <https://openreview.net/forum?id=Hy19ahVFwH>.
- [111] E. Erturk, T. C. Corke, and C. Gökçöl, “Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers,” *International Journal for Numerical Methods in Fluids*, vol. 48, no. 7, pp. 747–774, 2005.

- [112] H. Mobahi, R. Collobert, and J. Weston, “Deep learning from temporal coherence in video,” in *International Conference on Machine Learning*, ACM, 2009, pp. 737–744.
- [113] S. Zagoruyko and N. Komodakis, “Learning to compare image patches via convolutional neural networks,” in *Proc. Computer Vision and Pattern Recognition*, IEEE, 2015, pp. 4353–4361.
- [114] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, “Signature verification using a “Siamese” time delay neural network,” *Int. J. of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 04, pp. 669–688, 1993.
- [115] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning, ser. Adaptive computation and machine learning*. Cambridge, MA: The MIT Press, 2006.
- [116] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.
- [117] S. B. Pope, *Turbulent Flows*. Cambridge University Press, 2000.
- [118] C. Ma, L.-Y. Wei, B. Guo, and K. Zhou, “Motion field texture synthesis,” *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 110, 2009.
- [119] A. Bargteil, T. Goktekin, J. O’Brien, and J. Strain, “A semi-Lagrangian contouring method for fluid simulation,” *ACM Transactions on Graphics (TOG)*, vol. 25, no. 1, pp. 19–38, 2006.
- [120] V. Kwatra, D. Adalsteinsson, T. Kim, N. Kwatra, M. Carlson, and M. C. Lin, “Texturing fluids,” *Visualization and Computer Graphics*, vol. 13, no. 5, pp. 939–952, 2007.
- [121] R. Narain, V. Kwatra, H.-P. Lee, T. Kim, M. Carlson, and M. C. Lin, “Feature-guided dynamic texture synthesis on continuous flows,” in *Proc. Rendering Techniques*, Eurographics, 2007, pp. 361–370.
- [122] M. Wrenninge, *Production volume rendering: design and implementation*. AK Peters/CRC Press, 2012.
- [123] Q. Yu, F. Neyret, E. Bruneton, and N. Holzschuch, “Lagrangian Texture Advection: Preserving both Spectrum and Velocity Field,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17(11), pp. 1612–1623, 2010.
- [124] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, “Laplacian Surface Editing,” in *Proc. Symposium on Geometry Processing*, ACM/Eurographics, 2004, pp. 175–184.
- [125] T. Igarashi, T. Moscovich, and J. F. Hughes, “As-rigid-as-possible Shape Manipulation,” *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1134–1141, Jul. 2005.
- [126] P. Scovanner, S. Ali, and M. Shah, “A 3-dimensional sift descriptor and its application to action recognition,” in *Proc. Int. Conf. Multimedia*, ACM, 2007, pp. 357–360.

- [127] (CC) Blender Foundation | mango.blender.org, *Tears of Steel*, <https://mango.blender.org/>, 2011.
- [128] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *European Conference on Computer Vision*, Springer, 2016, pp. 694–711.
- [129] C. Wang, C. Xu, C. Wang, and D. Tao, “Perceptual adversarial networks for image-to-image transformation,” *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4066–4079, 2018.
- [130] M. S. M. Sajjadi, B. Schölkopf, and M. Hirsch, “Enhancenet: Single image super-resolution through automated texture synthesis,” in *Computer Vision (ICCV), 2017 IEEE International Conference on*, IEEE, 2017, pp. 4501–4510.
- [131] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia, “Video super-resolution via deep draft-ensemble learning,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 531–539.
- [132] M.-L. Eckert, W. Heidrich, and N. Thuerey, “Coupled fluid density and motion from single views,” in *Computer Graphics Forum*, Wiley Online Library, vol. 37(8), 2018, pp. 47–58.
- [133] C. Liu and D. Sun, “A Bayesian approach to adaptive video super resolution,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 209–216.
- [134] S. Nah, S. Baik, S. Hong, G. Moon, S. Son, R. Timofte, and K. Mu Lee, “NTIRE 2019 Challenge on Video Deblurring and Super-Resolution: Dataset and Study,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.
- [135] B. D. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision (DARPA),” in *Proceedings of the 1981 DARPA Image Understanding Workshop*, 1981, pp. 121–130.
- [136] E. Prashnani, H. Cai, Y. Mostofi, and P. Sen, “PieAPP: Perceptual Image-Error Assessment through Pairwise Preference,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1808–1817.
- [137] G. T. Fechner and W. M. Wundt, *Elemente der Psychophysik: erster Theil*. Breitkopf & Härtel, 1889.
- [138] K. Um, X. Hu, and N. Thuerey, “Perceptual evaluation of liquid simulation methods,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 143, 2017.
- [139] R. A. Bradley and M. E. Terry, “Rank analysis of incomplete block designs: I. The method of paired comparisons,” *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

- [140] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, “Super slomo: High quality estimation of multiple intermediate frames for video interpolation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9000–9008.
- [141] T. Inglis, M.-L. Eckert, J. Gregson, and N. Thuerey, “Primal-dual optimization for fluids,” in *Computer Graphics Forum*, Wiley Online Library, vol. 36, 2017, pp. 354–368.
- [142] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, “Mitsuba 2: A retargetable forward and inverse renderer,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, p. 203, 2019.
- [143] T. Takahashi and M. C. Lin, “Video-guided real-to-virtual parameter transfer for viscous fluids,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, p. 237, 2019.
- [144] X. Huang, “Sketch-based 4d prototyping for smoke simulations,” M.S. thesis, ETH Zurich, 2019.
- [145] Z. Hu, H. Xie, T. Fukusato, T. Sato, and T. Igarashi, “Sketch2vf: Sketch-based flow design with conditional generative adversarial network,” *Computer Animation and Virtual Worlds*, vol. 30, no. 3-4, e1889, 2019.





# Appendix



# A Technical Details for Video Generation

In the following, we give details of network architectures and training parameters used for the VSR and UVT tasks in Part III.

## A.1 Network Architecture

In this section, we use the following notation to specify all network architectures we proposed:  $\text{conc}()$  represents the concatenation of two tensors along the channel dimension;  $C/CT(\text{input}, \text{kernel\_size}, \text{output\_channel}, \text{stride\_size})$  stands for the convolution and transposed convolution operation, respectively; “+” denotes element-wise addition;  $\text{BilinearUp2}$  up-samples input tensors by a factor of 2 using bi-linear interpolation;  $\text{BicubicResize4}(\text{input})$  increases the resolution of the input tensor to 4 times higher via bicubic up-sampling;  $\text{Dense}(\text{input}, \text{output\_size})$  is a densely-connected layer, which uses Xavier initialization for the kernel weights.

The architecture of our VSR generator  $G$  is:

$$\begin{aligned} & \text{conc}(a_t, W(g_{t-1}, v_t)) \rightarrow l_{in}; C(l_{in}, 3, 64, 1), \text{ReLU} \rightarrow l_0; \\ & \quad \text{ResidualBlock}(l_i) \rightarrow l_{i+1} \text{ with } i = 0, \dots, n-1; \\ & CT(l_n, 3, 64, 2), \text{ReLU} \rightarrow l_{up2}; CT(l_{up2}, 3, 64, 2), \text{ReLU} \rightarrow l_{up4}; \\ & C(l_{up4}, 3, 3, 1), \text{ReLU} \rightarrow l_{res}; \text{BicubicResize4}(a_t) + l_{res} \rightarrow g_t. \end{aligned}$$

In  $\text{TecoGAN}^\odot$ , there are 10 sequential residual blocks in the generator ( $l_n = l_{10}$ ), while the  $\text{TecoGAN}$  generator has 16 residual blocks ( $l_n = l_{16}$ ). Each  $\text{ResidualBlock}(l_i)$  contains the following operations:  $C(l_i, 3, 64, 1), \text{ReLU} \rightarrow r_i; C(r_i, 3, 64, 1) + l_i \rightarrow l_{i+1}$ .

The VSR  $D_{s,t}$ ’s architecture is:

$$\begin{aligned} & I_{s,t}^g \text{ or } I_{s,t}^b \rightarrow l_{in}; C(l_{in}, 3, 64, 1), \text{Leaky ReLU} \rightarrow l_0; \\ & \quad C(l_0, 4, 64, 2), \text{BatchNorm}, \text{Leaky ReLU} \rightarrow l_1; \\ & \quad C(l_1, 4, 64, 2), \text{BatchNorm}, \text{Leaky ReLU} \rightarrow l_2; \\ & \quad C(l_2, 4, 128, 2), \text{BatchNorm}, \text{Leaky ReLU} \rightarrow l_3; \\ & \quad C(l_3, 4, 256, 2), \text{BatchNorm}, \text{Leaky ReLU} \rightarrow l_4; \\ & \quad \text{Dense}(l_4, 1), \text{sigmoid} \rightarrow l_{out}. \end{aligned}$$

VSR discriminators used in our variant models,  $D_sDt$ ,  $D_sDtPP$  and  $D_sOnly$ , have the same architecture as  $D_{s,t}$ . They only differ in terms of their inputs.

The flow estimation network  $F$  has the following architecture:

$$\begin{aligned} & \text{conc}(a_t, a_{t-1}) \rightarrow l_{in}; C(l_{in}, 3, 32, 1), \text{Leaky ReLU} \rightarrow l_0; \\ & C(l_0, 3, 32, 1), \text{Leaky ReLU}, \text{MaxPooling} \rightarrow l_1; C(l_1, 3, 64, 1), \text{Leaky ReLU} \rightarrow l_2; \\ & C(l_2, 3, 64, 1), \text{Leaky ReLU}, \text{MaxPooling} \rightarrow l_3; C(l_3, 3, 128, 1), \text{Leaky ReLU} \rightarrow l_4; \end{aligned}$$

## Appendix

$C(l_4, 3, 128, 1)$ , Leaky ReLU, MaxPooling  $\rightarrow l_5$ ;  $C(l_5, 3, 256, 1)$ , Leaky ReLU  $\rightarrow l_6$ ;  
 $C(l_6, 3, 256, 1)$ , Leaky ReLU, BilinearUp2  $\rightarrow l_7$ ;  $C(l_7, 3, 128, 1)$ , Leaky ReLU  $\rightarrow l_8$ ;  
 $C(l_8, 3, 128, 1)$ , Leaky ReLU, BilinearUp2  $\rightarrow l_9$ ;  $C(l_9, 3, 64, 1)$ , Leaky ReLU  $\rightarrow l_{10}$ ;  
 $C(l_{10}, 3, 64, 1)$ , Leaky ReLU, BilinearUp2  $\rightarrow l_{11}$ ;  $C(l_{11}, 3, 32, 1)$ , Leaky ReLU  $\rightarrow l_{12}$ ;  
 $C(l_{12}, 3, 2, 1)$ , tanh  $\rightarrow l_{out}$ ;  $l_{out} * \text{MaxVel} \rightarrow v_t$  .

Here, MaxVel is a constant vector, which scales the network output to the normal velocity range.

With the same motion estimator F, generators in UVT tasks have an encoder-decoder structure:

$\text{conc}(a_t, W(g_{t-1}, v_t)) \rightarrow l_{in}$  ;  $C(l_{in}, 7, 32, 1)$ , InstanceNorm, ReLU  $\rightarrow l_0$ ;  
 $C(l_0, 3, 64, 2)$ , InstanceNorm, ReLU  $\rightarrow l_1$ ;  
 $C(l_1, 3, 128, 2)$ , InstanceNorm, ReLU  $\rightarrow l_2$ ;  
 $\text{ResidualBlock}(l_2 + i) \rightarrow l_{3+i}$  with  $i = 0, \dots, n - 1$ ;  
 $CT(l_{n+2}, 3, 64, 2)$ , InstanceNorm, ReLU  $\rightarrow l_{n+3}$ ;  
 $CT(l_{n+3}, 3, 32, 2)$ , InstanceNorm, ReLU  $\rightarrow l_{n+4}$ ;  
 $CT(l_{n+4}, 7, 3, 1)$ , tanh  $\rightarrow l_{out}$

$\text{ResidualBlock}(l_2+i)$  contains the following operations:  $C(l_{2+i}, 3, 128, 1)$ , InstanceNorm, ReLU  $\rightarrow t_{2+i}$  ;  $C(t_{2+i}, 3, 128, 1)$ , InstanceNorm  $\rightarrow r_{2+i}$ ;  $r_{2+i} + l_{2+i} \rightarrow l_{3+i}$ . We use 10 residual blocks for all UVT generators.

Since UVT generators are larger than the VSR generator, we also use a larger  $D_{s,t}$  architecture:

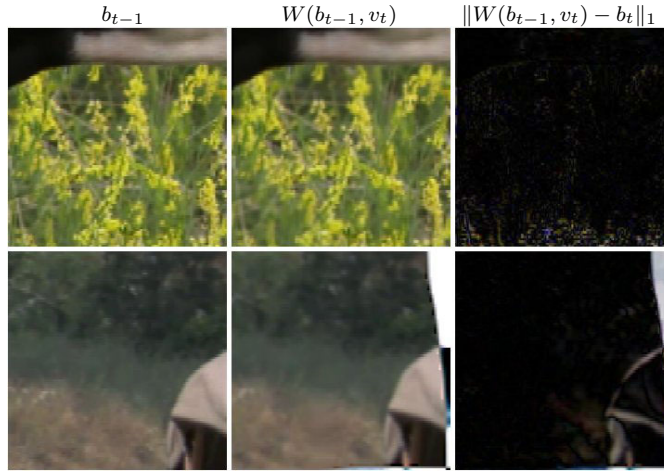
$I_{s,t}^g$  or  $I_{s,t}^b \rightarrow l_{in}$ ;  $C(l_{in}, 4, 64, 24)$ , ReLU  $\rightarrow l_0$ ;  
 $C(l_0, 4, 128, 2)$ , InstanceNorm, Leaky ReLU  $\rightarrow l_1$ ;  
 $C(l_1, 4, 256, 2)$ , InstanceNorm, Leaky ReLU  $\rightarrow l_2$ ;  
 $C(l_2, 4, 512, 2)$ , InstanceNorm, Leaky ReLU  $\rightarrow l_3$ ;  $Dense(l_3, 1) \rightarrow l_{out}$  .

Again, all ablation studies use the same architecture and only differ in terms of their inputs.

## A.2 Training Details

### A.2.1 Technical Details of the Spatio-Temporal Discriminator in VSR

In  $I_{wg}$  and  $I_{wb}$ ,  $F$  is used to warp frames. While contents are better aligned most of the time, at the boundary of images, the output of F is usually less accurate due to the lack of reliable neighborhood information. There is a higher chance that objects move into the field of view, or leave suddenly, which significantly affects the images warped with the inferred motion. An example is shown in Fig. A.1. This increases the difficulty for  $D_{s,t}$ , as it cannot fully rely on the images being aligned via warping. To alleviate this problem, we only use the center region of  $I_{wg}$  and  $I_{wb}$  as the input of the discriminator, and we reset a boundary of 16 pixels. Thus, for an input resolution of  $I_{wg}$  and  $I_{wb}$  of



**Figure A.1:** Near image boundaries, flow estimation is less accurate and warping often fails to align content. The first two columns show original and warped frames, the third one shows differences after warping (ideally all black). The top row shows that structures moving into the view can cause problems, visible at the bottom of the images, while the second row has no such problems (the objects are moving out of the view).

$128 \times 128$  for the VSR task, the inner part in size of  $96 \times 96$  is left untouched, while the border regions are overwritten with zeros.

The flow estimation network  $F$  with the loss  $\mathcal{L}_{G,F}$  should only be trained to support  $G$  in reaching the output quality as determined by  $D_{s,t}$ , but not the other way around. The latter could lead to  $F$  networks that confuse  $D_{s,t}$  with strong distortions of  $I_{wg}$  and  $I_{wb}$ . In order to avoid this undesirable case, we stop the gradient back propagation from  $I_{wg}$  and  $I_{wb}$  to  $F$ . In this way, gradients from  $D_{s,t}$  to  $F$  are only back propagated through the generated samples  $g_{t-1}, g_t$  and  $g_{t+1}$  into the generator network. As such,  $D_{s,t}$  can guide  $G$  to improve the image content, and  $F$  learns to warp the previous frame in accordance with the detail that  $G$  can synthesize. However,  $F$  does not adjust the motion estimation only to reduce the adversarial loss.

## A.2.2 Training Details for VSR and UVT

In the adversarial training of VSR and UVT, We employ a dynamic discriminator updating strategy, i.e. discriminators are not updated when there is a large difference between  $D(I_{s,t}^g)$  and  $D(I_{s,t}^b)$ . While our training runs are generally very stable, the training process could potentially be further improved with modern GAN algorithms, e.g. Wasserstein GAN [72].

To improve the stability of the adversarial training for the VSR task, we pre-train  $G$  and  $F$  together with a simple  $L^2$  loss of  $\mathbb{E} \|g_t - b_t\|_2 + \lambda_w \mathcal{L}_{warp}$  for 500k batches. Based on the pre-trained models, we use 900k batches for the proposed spatio-temporal adversarial training stage. Our training sequences has a length of 10 and a batch size

of 4. A black image is used as the first previous frame of each video sequence. I.e., one batch contains 40 frames and with the PP loss formulation, the NN receives gradients from 76 frames in total for every training iteration.

**Table A.1:** Training parameters

VSR Param	DsOnly	DsDt	DsDtPP	TecoGAN <sup>⊖</sup>	TecoGAN
$\lambda_a$	1e-3	Ds: 1e-3, Dt: 3e-4		1e-3	1e-3
$\lambda_p$	0.0	0.0		0.5	
$\lambda_\phi$	0.2 for VGG and 1.0 for Discriminator				
$\lambda_\omega, \lambda_c$	1.0, 1.0				
learning-rate	5e-5	5e-5 for Ds, 1.5e-5 for Dt.		5e-5	5e-5
		5e-5 for G, F.			
UVT Param	DsOnly	Dst	DsDtPP	TecoGAN	
$\lambda_a$	0.5		Ds: 0.5 Dt: 0.3	0.5	
$\lambda_p$	0.0	0.0		100.0	
$\lambda_\phi$	from $10^6$ decays to 0.0				
$\lambda_\omega$	0.0, a pre-trained F is used for UST tasks				
$\lambda_c$	10.0				

In the pre-training stage of VSR, we train the F and a generator with 10 residual blocks. An ADAM optimizer with  $\beta = 0.9$  is used throughout. The learning rate starts from  $10^{-4}$  and decays by 50% every 50k batches until it reaches  $2.5 * 10^{-5}$ . This pre-trained model is then used for all TecoGAN variants as initial state. In the adversarial training stage of VSR, all TecoGAN variants are trained with a fixed learning rate of  $5 * 10^{-5}$ . The generators in DsOnly, DsDt, DsDtPP and TecoGAN<sup>⊖</sup> have 10 residual blocks, whereas the TecoGAN model has 6 additional residual blocks in its generator. Therefore, after loading 10 residual blocks from the pre-trained model, these additional residual blocks are faded in smoothly with a factor of  $2.5 * 10^{-5}$ . We found this growing training methodology [75], to be stable and efficient in our tests. When training the VSR DsDt and DsDtPP, extra parameters are used to balance the two cooperating discriminators properly. Through experiments, we found  $D_t$  to be stronger. Therefore, we reduce the learning rate of  $D_t$  to  $1.5 * 10^{-5}$  in order to keep both discriminators balanced. At the same time, a factor of 0.0003 is used on the temporal adversarial loss to the generator, while the spatial adversarial loss has a factor of 0.001. During the VSR training, input LR video frames are cropped to a size of  $32 \times 32$ . In all VSR models, the Leaky ReLU operation uses a tangent of 0.2 for the negative half space. Additional training parameters are listed in Table A.1.

For the UVT task, a pre-training is not necessary for generators and discriminators since temporal triplets are gradually faded in. Only a pre-trained  $F$  model is reused. Trained on specialized data-sets, we found UVT models to converge well with 100k batches of sequences in length of 6 and batch size of 1.

For all UVT tasks, we use a learning rate of  $10^{-4}$  to train the first 90k batches and the last 10k batches are trained with the learning rate decay from  $10^{-4}$  to 0. Images of the input domain are cropped into a size of  $256 \times 256$  when training, the original size

being  $288 \times 288$ . The additional training parameters are also listed in Table A.1. For UVT,  $\mathcal{L}_{\text{content}}$  and  $\mathcal{L}_{\phi}$  are only used to improve the convergence of the training process. We fade out the  $\mathcal{L}_{\text{content}}$  in the first 10k batches and the  $\mathcal{L}_{\phi}$  is used for the first 80k and faded out in last 20k.