

Data-driven Non-rigid Reconstruction

Aljaž Božič

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Stefan Leutenegger

Prüfende der Dissertation:

1. Prof. Dr.-Ing. Matthias Nießner
2. Prof. Dr. Lourdes Agapito
3. Ass. Prof. Dr. Tolga Birdal

Die Dissertation wurde am 16.06.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 10.07.2024 angenommen.

Acknowledgement

My PhD journey has been an unforgettable experience, filled with learning and personal growth. It wouldn't have been the same without the company and support of many people around me, who shared with me the joyful moments, and also helped me get through any struggles on the way. While not an exhaustive list at all, I just want to express my gratitude to a few key people that contributed the most to my PhD story.

Firstly, I want to thank my supervisor, Matthias Nießner, for showing me how to aim high to achieve high-quality research results and teaching me how to effectively present my research ideas. Big thanks also to Michael Zollhöfer, who always patiently discussed any research challenges and whose technical advice helped me a lot to iterate on and improve research ideas. Further thanks to Justus Thies, for all the insightful research discussions, clever suggestions and positivity he always brought to the office. Additionally, I want to express my gratitude to Angela Dai, who taught me how to approach problems from a data-driven perspective, and Christian Theobalt, for his guidance in my early research years. I also want to thank defense committee members, Prof. Dr. Stefan Leutenegger, Prof. Dr. Lourdes Agapito and Prof. Dr. Tolga Birdal, for their valuable feedback and discussions.

Furthermore, I owe huge thanks to Pablo Palafox, a great colleague and collaborator on many projects, for always bringing his motivating attitude to the office. Also, special thanks to Armen Avetisyan, who shared with me invaluable insights about conducting experiment-driven research, and always made my days with his up-lifting stories. For all the happy moments we spent together, I want to thank my close colleagues, Ji Hou, for his often hilarious attitude to life, Dejan Azinovic, for numerous life discussions, Manuel Dahnert, for often leading a healthy example, Andreas Rössler, Yawar Siddiqui, Norman Müller, Dave Chen, Guy Gafni, Andrei Burov, Christian Diller, Alexey Bokhovkin, Artem Sevastopolsky, Lukas Höllein, David Rozenberszki, Peter Kocsis, Jiapeng Tang, Marc Benedí, Shivangi Aneja, Barbara Rössle, Can Gümeli and Yinyu Nie. I also want to express thanks to the visiting students and interns, Yang Li, Xiaochen Fan, Vassilis Choutas, Shijie Li, Jingwei Huang, Hassan Abu Alhaija, Yizhak Ben-Shabat and Cheng Lin, it was great spending summers together with you. Also to the colleagues beyond our lab, from the Prof. Westermann's and Prof. Thürey's groups, thanks for sharing with me all those special joyful moments. Finally, I am also grateful to Zhaoyang Lv, he was a great mentor during my internship in Meta Reality Labs Research, and he taught me how to approach research in the industry.

Lastly, I want to deeply thank my parents, Mateja and Branko Božič, for their enduring support and kindness, and my sisters, Sara and Julija, for always cheering me up. I am extremely grateful to my wife, Nan Liu, for always being there for me, also during sometimes stressful and challenging times, and making many sacrifices on the way.

Overall, huge thanks to my family, for all those peaceful moments we spent together and quick distractions, which always put me in a good mood, and helped me focus on a bigger picture, often resulting in the most creative research ideas.

Abstract

Reconstruction of 3D environments from a single moving sensor is an important problem in computer vision, enabling augmented and virtual reality applications, robust autonomous robot navigation in novel environments, and is a basis for 3D scene understanding. While static 3D reconstruction has progressed significantly in recent years, our everyday environments are inherently dynamic, resulting in a significantly more challenging non-rigid reconstruction problem, additionally requiring to track the non-rigid shape deformations through time. In our work we investigated data-driven priors for non-rigid 3D reconstruction and found them to be crucial for robust reconstruction performance in the highly underconstrained non-rigid setting.

First, we introduced DeepDeform that considerably improved non-rigid reconstruction from a single RGB-D camera by learning non-rigid correspondences for general deformable objects. Due to the lack of real recordings of diverse non-rigid objects, an important step was a collection of a large-scale dataset of non-rigid RGB-D videos. Dense correspondences between different frames were acquired in a semi-supervised fashion, requiring only sparse manual annotations, resulting in an efficient aggregation of ground truth data. Using these ground truth matches a heatmap-based network was trained to predict non-rigid correspondences, which were added as a data term to traditional non-rigid tracking, significantly outperforming state-of-the-art non-rigid reconstruction approaches.

Furthermore, we explored data-driven priors for other aspects of non-rigid reconstruction as well, such as correspondence outlier rejection, where no dense supervision is available. To that end we made the complete non-rigid tracking optimization end-to-end differentiable, and learned importance weights for all correspondences in a self-supervised fashion. The approach, called Neural Non-rigid Tracking, predicts robustly weighted correspondences best suited for the task of non-rigid tracking, which further improves non-rigid tracking performance and also considerably increases the execution speed. We additionally introduce a novel deformation graph and shape representation, where both are represented implicitly by a neural network, resulting in an end-to-end differentiable non-rigid reconstruction. The proposed Neural Deformation Graphs do not require any direct supervision for globally-consistent graph construction and shape reconstruction, and significantly improve the tracking robustness while using four RGB-D capture sensors.

Finally, instead of using RGB-D sensors, we also investigated reconstruction from a single RGB sensor, where no depth input is available. Since this setting is significantly more challenging, we restricted our algorithm to reconstruction of static environments. We proposed TransformerFusion, a transformer-based 3D scene reconstruction approach. Given an RGB video as input, the method learns to attend to the most important input

observations for each location in the scene, resulting in state-of-the-art reconstruction quality at interactive frame rates.

Overall, we thoroughly evaluate the contributions of aforementioned approaches and also list limitations that are left to be explored as future work.

Zusammenfassung

Die Rekonstruktion von 3D-Umgebungen aus einem sich bewegenden Sensor ist ein wichtiges Problem in der Computervision, das Augmented- und Virtual-Reality Anwendungen und robuste autonome Roboternavigation in neuartigen Umgebungen ermöglicht und als Grundlage für das Szenenverständnis dient. Obwohl es in den letzten Jahren große Fortschritte bei der statischen 3D-Rekonstruktion gegeben hat, sind unsere Umgebungen dynamisch, was zu einem viel anspruchsvolleren Problem der nicht-rigiden Rekonstruktion führt, das zusätzlich die Registrierung von Verformungen in Videos erfordert. In unserer Arbeit haben wir datengesteuerte Ansätze zur nicht-rigiden 3D-Rekonstruktion untersucht und festgestellt, dass sie für eine robuste Rekonstruktionsleistung von entscheidender Bedeutung sind.

Zuerst haben wir DeepDeform vorgestellt, das die nicht-rigide Rekonstruktion von einer einzelnen RGB-D-Kamera erheblich verbesserte, indem es nicht-rigide Korrespondenzen für verformbare Objekte lernte. Aufgrund des Mangels an realen Aufnahmen verschiedener deformierbarer Objekte war ein wichtiger Schritt die Sammlung eines großen Datensatzes nicht-rigider RGB-D-Videos. Korrespondenzen zwischen verschiedenen Frames wurden auf halbüberwachte Weise erfasst, was nur spärliche manuelle Annotation erforderte und zu einer effizienten Aggregation von Ground-Truth-Daten führte. Unter Verwendung dieser Ground-Truth-Korrespondenzen wurde ein Heatmap-basiertes Netzwerk trainiert, um Korrespondenzen auszugeben, wodurch die nicht-rigide Registrierung im Vergleich zu bestehenden Ansätzen erheblich verbessert wurde.

Wir haben Deep Learning auch für andere Aspekte der nicht-rigide Rekonstruktion untersucht, beispielsweise für die Ablehnung von Korrespondenzausreißern, bei denen keine Ground-Truth-Daten verfügbar sind. Zu diesem Zweck haben wir die komplette nicht-rigide Registrierung differenzierbar gemacht und selbstüberwachtes Lernen verwendet, um Konfidenzwerte für alle Korrespondenzen zu erhalten. Der als Neural Non-Rigid Tracking bezeichnete Ansatz gibt robust gewichtete Korrespondenzen aus, was die Leistung der nicht-rigiden Registrierung weiter verbessert und auch die Laufzeitgeschwindigkeit erheblich erhöht. Wir stellten zusätzlich eine neuartige Deformations- und Geometrirepräsentation vor, bei der beide implizit durch ein neuronales Netzwerk repräsentiert werden, was zu einer komplett differenzierbaren nicht-rigide Rekonstruktion führt. Die vorgeschlagenen Neural Deformation Graphs benutzen selbstüberwachtes Lernen für eine global konsistente Graphoptimierung und Geometriekonstruktion, und verbessern die Registrierungsrobustheit bei Verwendung von vier RGB-D-Sensoren erheblich.

Anstatt RGB-D-Sensoren zu verwenden, untersuchten wir schließlich auch die Rekonstruktion von einem einzelnen RGB-Sensor, bei dem keine Tiefeneingabe verfügbar ist. Da dies deutlich anspruchsvoller ist, haben wir unseren Ansatz auf die Rekonstruk-

tion statischer Umgebungen beschränkt. Wir haben TransformerFusion vorgeschlagen, einen auf Transformer-Netzwerk basierenden Ansatz zur 3D-Rekonstruktion. Bei einem RGB-Video als Eingabe lernt der Ansatz, wichtige Eingabebeobachtungen für jede 3D-Position in der Umgebung zu berücksichtigen, was zu einer hochmodernen Rekonstruktionsqualität bei interaktiven Bildraten führt.

Insgesamt bewerten wir die Beiträge der oben genannten Ansätze gründlich und listen auch Einschränkungen auf, die als zukünftige Arbeit noch untersucht werden müssen.

Contents

Acknowledgement	iii
Abstract	v
Zusammenfassung	vii
I Introduction	1
1 Introduction	3
1.1 Dissertation Overview	7
1.2 Contributions	7
1.3 List of Publications	9
2 Background	11
2.1 Shape Representations	11
2.1.1 Mesh	11
2.1.2 Point Cloud	11
2.1.3 Signed Distance Field	12
2.1.4 Occupancy Field	13
2.2 Efficient Nonlinear Optimization	13
2.2.1 Nonlinear Least Squares Problem	14
2.2.2 Gauss-Newton Algorithm	14
2.2.3 Conjugate Gradient Descent	16
2.2.4 Differentiable Optimization	18
2.3 Non-rigid Tracking	19
2.3.1 Deformation Graph	19
2.3.2 Motion Interpolation	21
2.3.3 Motion Regularization	21
2.3.4 Iterative Closest Points	22
2.3.5 Global Correspondences	23
2.3.6 Optimization Energy	24
2.4 Non-rigid Reconstruction	25
2.4.1 Non-rigid Volumetric Fusion	25
2.4.2 Neural Shape Representations	28
2.4.3 Learning Shape Fusion	30

II	Non-rigid Reconstruction using Data-driven Priors	33
3	Learning Non-rigid Reconstruction	35
3.1	Introduction	35
3.2	Related Work	37
3.3	Data-driven Non-Rigid Matching	39
3.4	Non-Rigid Reconstruction Pipeline	41
3.4.1	Deformation Model	42
3.4.2	Optimization Terms	42
3.4.3	Energy Optimization	43
3.5	Semi-supervised Data Acquisition	43
3.5.1	Data Acquisition	43
3.5.2	Data Annotation	43
3.5.3	Dense Data Alignment	44
3.6	Experiments	45
3.6.1	Non-Rigid Matching Evaluation	45
3.6.2	Non-Rigid Reconstruction Results	47
3.6.3	Ablation Study	52
3.6.4	Data Generation Evaluation	52
3.6.5	Limitations	53
3.7	Conclusion	53
4	Neural Non-rigid Tracking	55
4.1	Introduction	55
4.2	Related Work	57
4.3	Non-Rigid Reconstruction Notation	58
4.4	Neural Non-rigid Tracking	59
4.4.1	Dense Correspondence Prediction	59
4.4.2	Correspondence Importance Weights	60
4.4.3	Differentiable Optimizer	61
4.4.4	End-to-end Optimization	62
4.4.5	Neural Non-rigid Tracking for 3D Reconstruction	65
4.5	Experiments	65
4.5.1	Training Scheme	65
4.5.2	Non-rigid Tracking Evaluation	66
4.5.3	Non-rigid Reconstruction Evaluation	68
4.6	Conclusion	69
5	Neural Deformation Graphs	73
5.1	Introduction	73
5.2	Related Work	75
5.3	Method	77
5.3.1	Neural Deformation Graphs	78
5.3.2	Global Deformation Optimization	80

5.3.3	Implicit Surface Reconstruction	81
5.3.4	Training Details	81
5.4	Results	83
5.5	Conclusion	89
6	Learning RGB Reconstruction	91
6.1	Introduction	91
6.2	Related Work	93
6.3	End-to-end 3D Reconstruction using Transformers	94
6.3.1	Learning Temporal Feature Fusion via Transformers	95
6.3.2	Spatial Feature Refinement	96
6.3.3	Surface Occupancy Prediction	97
6.3.4	View Selection for Online Scene Reconstruction	97
6.3.5	Training Scheme	99
6.4	Experiments	100
6.4.1	Comparison with State of the Art	100
6.4.2	Ablations	104
6.4.3	Runtime Analysis	106
6.4.4	Limitations	108
6.5	Conclusion	108
III	Conclusion & Outlook	111
7	Conclusion	113
8	Limitations and Future Work	115
	Bibliography	117
	Appendix	133
A	Open-source Code & Videos	135
A.1	DeepDeform	135
A.2	Neural Non-Rigid Tracking	135
A.3	Neural Deformation Graphs	135
A.4	TransformerFusion	135
B	Authored and Co-authored Publications	137
C	Original Publications	139
C.1	DeepDeform: Learning Non-rigid RGB-D Reconstruction with Semi-supervised Data	140
C.2	Neural Non-Rigid Tracking	153

C.3	Neural Deformation Graphs for Globally-consistent Non-rigid Reconstruction	166
C.4	TransformerFusion: Monocular RGB Scene Reconstruction using Transformers	178
D	Differentiable Non-rigid Optimization	193
D.1	Partial Derivatives	193
D.2	Differentiable Linear Solve Operation	195
	Acronyms	197
	List of Tables	199
	List of Figures	201

Part I

Introduction

1 Introduction

One of the most important problems in computer vision is 3D reconstruction. When our everyday environment is recorded by a camera, only a 2D projection of the world is observed, in a format of an image or a video. Reconstructing a physically accurate 3D model of the environment that generated these 2D observations is a crucial task that serves as a basis for autonomous agent localization and navigation in novel environments, digitalization of real-world scenes for augmented and virtual reality applications, and 3D semantic understanding.

In recent years, there has been a tremendous progress in 3D reconstruction of static environments. When recording a scene with a moving camera, the task is to estimate camera poses, i.e. a camera rotation and translation for every frame, while at the same time reconstructing the scene geometry. Structure-from-Motion (SfM) approaches [1] extract sparse features from images, and simultaneously optimize the camera poses and 3D positions of feature keypoints, resulting in a sparse map of the scene. With the introduction of affordable depth sensors, such as Microsoft Kinect, the sparse point clouds were extended into a complete dense 3D reconstruction via depth map fusion, as presented in a seminal work of KinectFusion [2].

However, our natural environments are inherently dynamic. Humans and animals are almost always in motion, and also influence their surroundings, e.g. by opening doors, moving a chair when sitting down, wearing a coat, etc. Furthermore, the motion can be caused by external factors as well, such as a car engine, gravity, or just wind. To accurately capture everyday environments for use in augmented and virtual reality, it's very



Figure 1.1: Applications of non-rigid reconstruction. Possible use cases range from telepresence in VR/AR (*left*, from [3]) to safe human-robot interactions (*right*).

important to account for dynamics in the scene. In robotics and autonomous driving, dynamic objects have to be properly taken care for to achieve robust localization and obstacle avoidance of autonomous agents. For accurate robot manipulation of deformable objects, a 3D model of the object needs to be reconstructed and the deformations have to be tracked. As soon as the objects move non-rigidly, i.e. not only related by a rotation and translation, but instead including more complex non-rigid deformations, we make use of *non-rigid reconstruction* to track and reconstruct the objects.

Given a video recording of a deformable object in motion, the task of non-rigid reconstruction is to track the dense deformations, and at the same time reconstruct the shape of the object. A seminal work is DynamicFusion [4], an approach for real-time non-rigid reconstruction from a single RGB-D sensor. A core building block is the incremental reconstruction of a *canonical shape*, i.e. the object’s shape in the initial frame of the video. Initialized with a partial shape from the first depth map, a canonical shape is non-rigidly tracked frame-by-frame, and every newly observed depth map is used to update the canonical shape, making it more complete (as visualized in Fig. 1.2). The method doesn’t make any strong assumptions about the object or deformation type, and achieves impressive tracking and reconstruction results for various deformable objects. Its tracking can however fail in cases where motion is too fast, or the object undergoes severe (self-)occlusions, which in turn corrupts the shape reconstruction as well. We propose several ways to make non-rigid reconstruction significantly more robust by using *data-driven priors*.

A very common example of non-rigid objects are humans. A wide range of approaches were developed that focus on tracking and reconstruction of humans, or more specifically, of human faces and bodies. Large-scale datasets of human body shapes [5] and poses [6] were collected, and a principal-component analysis (PCA) was used to model the changes in body shape across different identities. To model the body motion, the body shape is deformed by a sparse human skeleton [6]. Similarly for faces, sparse facial keypoints are used to guide the face deformations [7]. These approaches are *data-driven*, i.e. they rely on priors from collected data, which enables them to feature robust tracking and reconstruction performance even in the underconstrained settings with limited sensors, e.g. using only an RGB video [7]. However, they are very object-specific (e.g. relying on human skeletons or facial keypoints), and cannot be easily extended to other non-rigid objects. To be able to also robustly track and reconstruct deforming clothes, bags, pillows, blankets, toys, etc., we need a more general approach for *data-driven non-rigid reconstruction* without object-specific assumptions.

For body and face tracking, reliable data-driven skeleton and face keypoint trackers [8] have a significant impact on robust tracking and reconstruction. We extend the notion of object-specific keypoints to any non-rigid object by exploring general *non-rigid correspondences*. The idea is to not limit the tracker to detect only a specific set of predefined keypoints, but instead to learn to predict a corresponding pixel in the target image for any chosen pixel from the source image. Adding the predicted correspondences as additional constraints at non-rigid tracking enables considerably more robust frame-to-frame alignment compared to DynamicFusion, especially in the case of fast motion. Since we

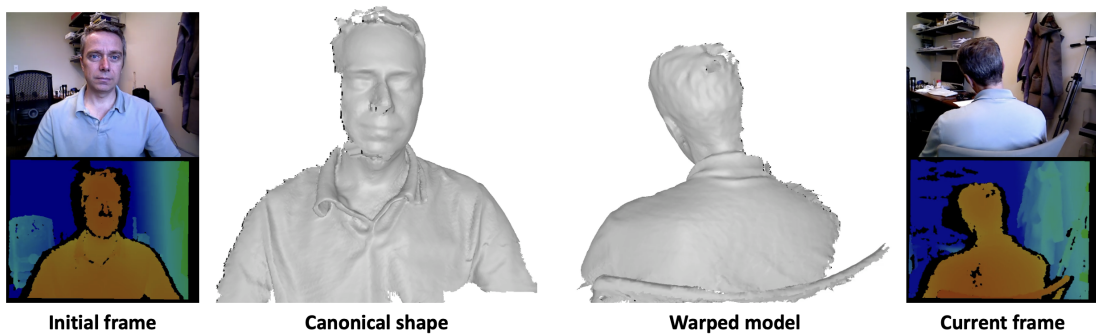


Figure 1.2: Non-rigid reconstruction pipeline. A seminal work of DynamicFusion [4] incrementally reconstructs a canonical shape and at the same time tracks deformations over input RGB-D frames.

use a single camera, a large part of the object is always occluded. When the occluded parts of the object become visible again, the relative deformation can be extremely large. To correctly re-detect the occluded surface, and compensate for large deformation that happened under occlusion, global correspondences are crucially important. Since the local geometry and appearance change under extreme deformation, but the semantic meaning usually persists (e.g. we can identify corresponding points on the shirt’s sleeve even under a large deformation), this motivates exploring data-driven priors for non-rigid correspondences.

To learn non-rigid correspondences, we require a dataset with ground truth matches on non-rigid objects undergoing strong deformation. There exist some datasets with correspondence ground truth, but they are either of smaller scale and synthetic [9], or focus on a very specific scenario, such as autonomous driving [10] or human bodies [11]. Therefore, we introduce a novel dataset with 400 RGB-D recordings of general deformable objects, featuring loosely-clothed humans, cloth pieces, pillows, blankets, animals, bags, backpacks, etc. In every sequence, sparse correspondences between frame pairs were manually annotated and extended to dense ground truth motion using correspondence-guided non-rigid alignment. A convolutional neural network (CNN) is trained for non-rigid correspondence matching, taking as input an image patch centered around any point of interest in source frame, and predicting the corresponding pixel location in the target frame in form of a probability heatmap. These predicted non-rigid correspondences are added to the non-rigid tracking objective, which significantly improves the tracking and reconstruction performance in comparison to existing state-of-the-art approaches.

While learned correspondences considerably improve the robustness and quality of the non-rigid reconstruction, they come at a cost – predicting a match for each pixel requires a network evaluation, and more than a hundred correspondences are estimated at each frame, sacrificing the real-time performance. To improve the runtime, we replaced the sequential match prediction with a CNN that, given an entire source image, directly predicts dense correspondences for all pixels with a single network evaluation. Furthermore, when using predicted correspondences for non-rigid tracking, some pre-

diction error is inevitable. To minimize the error, we additionally detect outliers with another CNN that predicts the confidence of correspondences. To keep only correspondences that are best suited for non-rigid alignment, we make the complete non-rigid solver differentiable, and learn the correspondence weighting that causes the solver to converge to the correct deformations. Not only we significantly speed-up the runtime, but also improve the tracking robustness and reconstruction quality, thanks to learning the *outlier rejection* in a self-supervised manner.

Incremental frame-to-frame tracking and reconstruction enables real-time performance, and with introduction of data-driven priors it results in a high-quality reconstruction. However, any errors that arise at any frame get propagated to all next frames. In certain cases, e.g. for ground truth data collection or non-rigid template optimization for re-animation, a slower runtime is completely acceptable, if it delivers higher quality tracking and reconstruction results. We introduce a *global optimization strategy*, optimizing object’s shape and deformations in an end-to-end manner, over all input frames at the same time. The main challenges are to make the entire non-rigid reconstruction pipeline differentiable and to come up with an efficient optimization algorithm, since considering all frames at the same time is computationally very expensive. To this end, we propose to use specifically designed neural networks to represent both the object’s shape as well as the deformations, which naturally results in end-to-end differentiable model. We use the ADAM solver [12] that stochastically evaluates the loss at random batch samples and is well-suited for neural network training, offering efficient and scalable batch-wise optimization. Focusing on non-rigid capture with a sparse set of RGB-D sensors, the proposed approach is able to recover accurate object reconstruction when undergoing very complex deformations, even topology changes.

Finally, there is an even stronger need for priors if we make the problem more challenging and decide to use only an RGB camera, without any depth sensor. That makes the approach widely applicable, but dense reconstruction from an RGB video is highly underconstrained even when capturing static scenes, so we decided to assume a static environment and focus on rigid 3D reconstruction. We explored learning the fusion of 2D RGB observations directly into a 3D feature representation, which is decoded by a multi-layer perceptron (MLP) into occupancy field with values of 1 inside the surface in 0 in free space. We found the transformer architecture especially effective for feature fusion – it’s able to aggregate important 2D information between frames that can be very far apart in time, while existing fusion methods proceed in temporal domain and can easily forget previous observations. By processing scenes chunk-wise, the approach can handle larger scenes, only fusing observations in the current view frustum. Attention weights from transformer architecture enable efficient observation selection, which makes it possible to execute reconstruction incrementally and at interactive speeds. The approach achieves state-of-the-art reconstruction performance, and the learned fusion could be extended to non-rigid domain in the future.

Overall, we explored a wide variety of data-driven priors for (non-)rigid tracking and reconstruction in this thesis, ranging from learning correspondences and outlier rejection for robust deformation tracking, optimizing both shape and deformation model in

an end-to-end manner, and learning fusion of 2D observations into 3D features for shape reconstruction, even when no depth input is available. We show that there is a clear benefit of using data-driven priors to add better constraints to the highly underconstrained problem of non-rigid reconstruction from a single sensor. Data-driven non-rigid reconstruction is a very exciting direction, and we hope this thesis is a good introduction and basis for all future works in the non-rigid reconstruction field.

1.1 Dissertation Overview

The thesis consists of 8 chapters that are grouped into three parts:

- **Part I:** Introduction (Chapters 1-2)
 - Chapter 1 (Introduction) motivates the importance of non-rigid reconstruction and highlights key challenges of existing works.
 - Chapter 2 (Background) explains non-rigid tracking and reconstruction approaches that build upon core shape representations and efficient nonlinear optimization.
- **Part II:** Non-rigid Reconstruction using Data-driven Priors (Chapters 3-6)
 - Chapter 3 (Learning Non-rigid Reconstruction) introduces a novel dataset and a robust reconstruction approach using learned non-rigid correspondences.
 - Chapter 4 (Neural Non-rigid Tracking) presents a differentiable non-rigid tracking solver enabling fast correspondence prediction with self-supervised outlier rejection.
 - Chapter 5 (Neural Deformation Graphs) introduces a globally-consistent deformation graph and shape optimization, both represented by networks.
 - Chapter 6 (Learning RGB Reconstruction) proposes a learned transformer-based RGB video fusion for interactive 3D scene reconstruction.
- **Part III:** Conclusion & Outlook (Chapters 7-8)
 - Chapter 7 (Conclusion) concludes the thesis and summarizes the key contributions.
 - Chapter 8 (Limitations and Future Work) lists limitations of proposed works and presents promising directions left for future work.

1.2 Contributions

This thesis explores the use of data-driven priors for non-rigid 3D reconstruction of deformable objects. To enable learning of a variety of priors, a large-scale dataset of 400 non-rigid RGB-D recordings was collected, including object masks and dense correspondences. We designed a novel non-rigid correspondence network trained on these

correspondences that became a core building block of *DeepDeform* [13], a non-rigid reconstruction approach that significantly improved reconstruction and tracking performance over existing methods that don't use any learned priors. The non-rigid reconstruction robustness and accuracy was further enhanced in *Neural Non-rigid Tracking* [14], where we proposed an end-to-end differentiable non-rigid tracker that learns dense correspondence best-suited for the task of non-rigid tracking, and at the same time detects outliers in a self-supervised manner. With *Neural Deformation Graphs* [15] we went beyond priors for non-rigid tracking and introduced a novel globally-consistent deformation graph and shape optimization, where both are represented implicitly by a neural network, resulting in a fully differentiable non-rigid reconstruction framework that considerably surpasses the state of the art. Finally, to enable reconstruction in an even more underconstrained setting, with only a monocular RGB video given as input, we proposed *TransformerFusion* [16], where the fusion of 2D color observations is learned using an efficient transformer-based architecture, achieving accurate reconstructions of 3D scenes at interactive frame-rates. In the following we summarize the contributions of each publication in more detail:

- To address the lack of data for learning priors for non-rigid reconstruction, we introduce a large-scale dataset of 400 non-rigid sequences with over 390,000 frames and 5,533 densely aligned frame pairs, with correspondences computed in a semi-supervised fashion, using manually annotated sparse matches to guide the dense non-rigid alignment. Data capturing and processing pipeline was built up with the help of Angela Dai, since we followed the capture infrastructure used for the ScanNet dataset [17]. We propose a novel non-rigid correspondence matching strategy that outperforms previously used hand-crafted descriptors. When integrated into a non-rigid reconstruction framework, it is able to handle more complex and faster deformable motions, resulting in a significantly improved non-rigid tracking and reconstruction performance. Discussions with the co-authors led to the final paper [13].
- We propose a novel, end-to-end differentiable, non-rigid tracker that achieves state-of-the-art non-rigid reconstruction performance by learning a robust optimization. The differentiable Gauss-Newton solver provides valuable gradients for learning the dense non-rigid correspondences, and enables the prediction of importance weights for outlier rejection in a purely self-supervised manner. Additionally, the prediction of robust dense correspondences is $85\times$ faster compared to previously used deep-learned sparse matching. This is a shared publication with Pablo Palafox, who explored optical flow network architectures in depth, and after the initial setup of the differentiable solver, he extensively experimented to end up with an optimal model for predicting non-rigid correspondences and outlier rejection. Discussions with the co-authors led to the final paper [14].
- We introduce a novel network-based deformation graph and shape representation that enables globally-consistent non-rigid tracking and reconstruction of de-

formable objects. The deformation graph is implicitly represented by a neural network and globally-optimized using the proposed per-frame viewpoint consistency and inter-frame surface consistency losses, resulting in robust tracking of fast deformations. The shape is reconstructed using a multi-MLP architecture anchored at the deformation nodes, featuring accurate and deformation-dependent surface geometry. We conducted extensive experiments on both synthetic and real recordings, surpassing the state of the art by more than 60%. Pablo Palafox accurately calibrated the RGB-D sensors and set up the recording pipeline for capturing all real sequences. Discussions with the co-authors led to the final paper [15].

- To enable 3D reconstruction using only a monocular RGB video, without a depth sensor, we propose a transformer-based learned fusion of color observations from different video frames. For each 3D location in the scene, the transformer network attends to only the most informative features in the image views, and efficiently aggregates the information in the temporal domain. We fuse features in a coarse-to-fine fashion, storing fine-level features only where needed, enabling online reconstruction running at interactive frame-rates. The proposed approach achieves accurate 3D scene reconstructions, outperforming existing state-of-the-art multi-view stereo depth estimation approaches, fully-convolutional 3D reconstruction methods, and LSTM- or GRU-based recurrent networks for RGB video fusion. Discussions with the co-authors led to the final paper [16].

1.3 List of Publications

A. Božič, M. Zollhöfer, C. Theobalt, and M. Nießner, “Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020

A. Božič*, P. Palafox*, M. Zollhöfer, A. Dai, J. Thies, and M. Nießner, “Neural non-rigid tracking,” in *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2020

A. Božič, P. Palafox, M. Zollhöfer, J. Thies, A. Dai, and M. Nießner, “Neural deformation graphs for globally-consistent non-rigid reconstruction,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021

A. Božič, P. Palafox, J. Thies, A. Dai, and M. Nießner, “Transformerfusion: Monocular rgb scene reconstruction using transformers,” in *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2021

2 Background

2.1 Shape Representations

When modeling 3D objects and scenes, we make use of a variety of surface representations that encode the geometry of the 3D structures. The best-fitting representation can depend on the given task, and it's common to convert from one representation to another when solving a complex 3D computer vision problem. In the following we introduce commonly used shape representations in state-of-the-art (non-)rigid tracking and reconstruction approaches, and we visualize (some of) them in Fig. 2.1.

2.1.1 Mesh

A very common surface representation is a mesh, a collection of *vertices* and *faces*. The vertices are a fixed number of 3D points that can beside positions include also other properties, such as a vertex color, a texture (u, v) coordinate, or a normal vector. To extend the surface continuously beyond the discrete set of vertices, faces connect different vertices into triangles, quadrilaterals, etc., defining a continuous 2D surface. At any point on these 2D shapes the vertex properties are linearly interpolated.

Meshes are a very efficient surface representation, storing only a sparse set of surface elements, while still being able to represent the geometry to any desired level of detail (by introducing a finer discretization). That is why they are often considered as a default representation for rendering in graphics, and very efficient hardware-optimized routines exist in the graphics pipeline for executing rendering-related tasks using meshes. For real-time non-rigid tracking we make heavy use of these routines to accelerate the algorithm's performance.

2.1.2 Point Cloud

In certain cases it's hard to define the connectivity between mesh vertices. For example, if we are dealing with a recorded frame from an RGB-D sensor, then for each pixel we read the pixel color and depth values, but there is no information about how different pixels are related to each other. A simpler representation is more suitable in this case, where we only store 3D points with their attributes, such as color, normal, etc., resulting in a point cloud. This representation is not directly usable for surface rendering, since sparse points do not densely cover all pixels in a rendered view (unless extended into a volumetric representation, e.g. via spheres). However, it still contains a lot of useful information about the surface geometry and has less strict design requirements compared

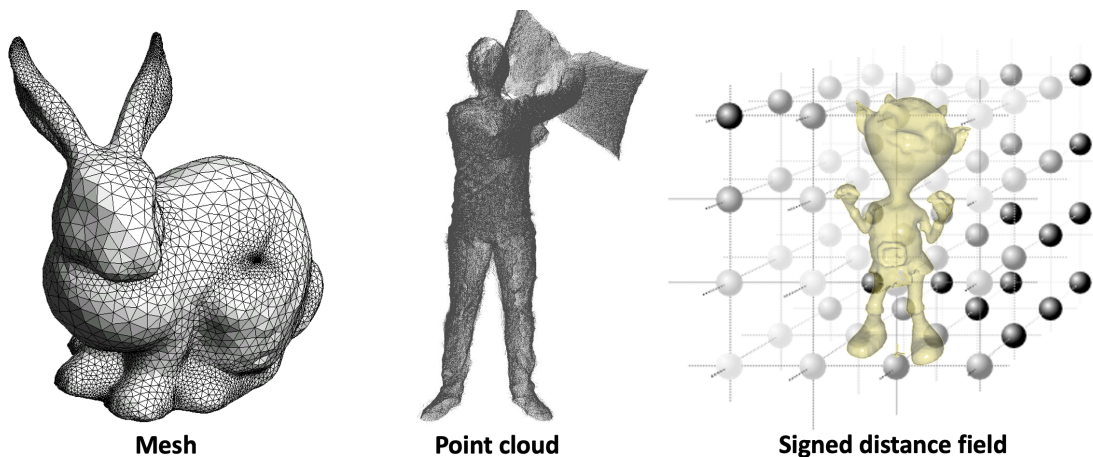


Figure 2.1: Shape representations. We can model the surface of a 3D object as a mesh (*left*), a point cloud (*middle*) or a signed distance field (*right*).

to meshes, which makes it very suitable for representing data captured by an RGB-D or LiDAR scanner.

2.1.3 Signed Distance Field

The above described representations define the surface *explicitly*, i.e. by directly specifying the surface points (or faces). In certain scenarios it is beneficial to instead represent the surface *implicitly*, e.g. by storing a distance value to the surface. Often a dense 3D voxel grid is used to discretize the 3D space and for each voxel center a distance to the nearest surface geometry is computed, resulting in a distance field. We can further differentiate the voxels inside and outside the shape surface by adding a sign to the distance values, considering positive distance for voxels outside the shape and negative distances in the shape interior, defining the signed distance field (SDF).

Grid-based SDF is a volumetric representation, its memory requirements grow cubically with the grid resolution. But it's a well-suited representation for many reconstruction algorithms, since with a fixed grid resolution it can represent shapes with varying number of vertices, connectivity and topology, making geometry optimization over very different shapes possible. Using an algorithm called Marching cubes [18] a triangular mesh can be efficiently constructed from an SDF grid, extracting mesh vertices and triangles at the zero level set of the SDF.

Beside using a discrete 3D voxel grid, the SDF can also be represented by a multi-layer perceptron (MLP) that takes as input a 3D position and outputs its signed distance value. This can significantly lower the memory requirements and at the same time it enables the continuous modeling of SDF, avoiding the loss of geometry detail that is inevitable when using a fixed grid resolution. The downside is the slower evaluation of the SDF value at any point, since instead of directly querying the voxel grid values a

network evaluation is required. When slower evaluation is acceptable, MLP-based SDF representation is a good alternative to voxel grids.

2.1.4 Occupancy Field

Instead of modeling the surface as a continuous signed distance field, we can define a discrete occupancy field with a value of 1 inside the shape and value 0 outside. It shares many advantages of SDF, e.g. the field resolution doesn't limit the object topology. At the same time, it can be constructed with less information about the surface, since the distance values are not needed, only the sign (i.e. inside or outside) is important. Therefore, occupancy field offers an alternative representation useful in cases when distances to the surface are hard or expensive to compute.

2.2 Efficient Nonlinear Optimization

A majority of problems in computer vision involve fitting a problem-specific model to a given set of observations, and a core component is solving the optimization problem. In a Structure-from-Motion problem, the model consists of camera poses (one for each frame) and sparse 3D points (corresponding to discriminative keypoints in the scene), and the observations are the 2D keypoint detections in each frame – the goal is to find the optimal camera poses and 3D feature positions that minimize the 2D reprojection error over all frames. When we train a neural network for a specific task, the model is the network architecture, and the observations are the ground truth labels, e.g. object detections – we are again solving an optimization problem by minimizing the task-specific training loss. Similarly, as will be presented in the following sections, we encounter optimization problems in non-rigid tracking and reconstruction algorithms.

While certain optimization problems are linear, we are mostly interested in solving *nonlinear optimization problems*, i.e. where the transformation of model parameters during the computation of the optimization energy or loss function involves nonlinear operations. For example, a multi-layer perceptron (MLP) consists of nonlinear activation functions, such as a rectified linear unit (ReLU), making any optimization problem involving MLPs nonlinear. Furthermore, the optimization loss is often nonlinear, such as the 2D reprojection error that involves nonlinear projection of 3D feature positions to the image plane. Non-rigid tracking and reconstruction also include highly nonlinear optimization energies. Therefore, the optimization algorithms presented in this chapter are applied to nonlinear optimization problems.

When it comes to many applications in VR/AR or robotics, we are interested in solving the optimization problems in *real-time*, i.e. at the frequency of the measurement device (e.g. an RGB-D sensor) or the display (e.g. VR glasses). We will focus on exploring *efficient* nonlinear optimization algorithms, both in terms of convergence speed and efficient execution. To achieve *fast convergence*, i.e. to solve an optimization problem in as few iterations as possible, we decided to use a second-order nonlinear optimization algorithm. At the same, to minimize the execution time of each iteration, we use an

approximate second-order *Gauss-Newton solver*. In each iteration of the Gauss-Newton algorithm a linear system needs to be solved, and we make use of a highly parallelizable *conjugate gradient descent* that iteratively solves the linear system. Both algorithms are described in more detail in the following sections, and we also explain how to make the algorithms' modules differentiable, for the use in *differentiable optimization*.

2.2.1 Nonlinear Least Squares Problem

In a nonlinear optimization problem we can denote the N -dimensional problem variables as $\mathbf{x} \in \mathbb{R}^N$ and the objective function as $\mathcal{L} : \mathbb{R}^N \rightarrow \mathbb{R}$. The goal is to find the optimal values for \mathbf{x} that minimize the objective function \mathcal{L} :

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^N} \mathcal{L}(\mathbf{x})$$

While the objective function \mathcal{L} can take any form in general, it turns out it's often beneficial to reformulate nonlinear optimization problems into a more specific type of the objective function, denoted as *nonlinear least squares*. In that case, we introduce M functions $r_m : \mathbb{R}^N \rightarrow \mathbb{R}$, with $m \in \{1, \dots, M\}$, converting objective function \mathcal{L} into the following form:

$$\mathcal{L}(\mathbf{x}) = \sum_{m=1}^M r_m^2(\mathbf{x})$$

Functions r_m are also called *residuals*, since they often measure the fitting error, i.e. the difference between the observed value and the estimated value, predicted by the model.

A common choice to minimize the nonlinear objective function \mathcal{L} is a Newton's method. It is a second-order iterative optimization approach that offers fast convergence, but it suffers from high memory and compute requirements at every iteration of the algorithm. The nonlinear least squares formulation of the objective \mathcal{L} makes it possible to develop more computationally and memory efficient algorithms, such as Gauss-Newton, enabling real-time performance of non-rigid tracking and reconstruction approaches.

2.2.2 Gauss-Newton Algorithm

In order to derive the Gauss-Newton algorithm for efficient nonlinear optimization, we start by describing the Newton's method, a second-order nonlinear optimization approach. Given the variables $\mathbf{x} = (x_1, \dots, x_N)^T$ and a loss function \mathcal{L} , we define the gradient $\mathbf{g} \in \mathbb{R}^N$ of \mathcal{L} wrt. variables $\mathbf{x} \in \mathbb{R}^N$ as a vector of partial derivatives:

$$g_i(\mathbf{x}) = \frac{\partial \mathcal{L}(\mathbf{x})}{\partial x_i}$$

Furthermore, we compute the second-order partial derivatives and define the Hessian matrix $\mathbf{H} \in \mathbb{R}^{N \times N}$ as:

$$H_{ij}(\mathbf{x}) = \frac{\partial^2 \mathcal{L}(\mathbf{x})}{\partial x_i \partial x_j}$$

Given an initial value of parameters \mathbf{x}_0 , the Newton's method *iteratively* updates the parameter values to minimize the loss function \mathcal{L} using the following update step:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{H}(\mathbf{x}_n)^{-1} \mathbf{g}(\mathbf{x}_n)$$

The algorithm proceeds with iterative updates until the update steps are smaller than some predefined value ϵ , i.e. $\|\mathbf{x}_{n+1} - \mathbf{x}_n\|_2 < \epsilon$ – at that point the method converges to a (local) minimum. A nice property of a second-order approach is the *quadratic convergence*, i.e. the distance of the current estimate \mathbf{x}_{n+1} to the root is proportional to the square of the distance of the previous estimate \mathbf{x}_n to the root. However, it comes at the cost of computing a Hessian matrix \mathbf{H} of second-order partial derivatives.

If we apply Newton's method to the nonlinear least squares problem, we can derive more specific formulations for gradient vector \mathbf{g} and Hessian matrix \mathbf{H} :

$$g_i(\mathbf{x}) = 2 \sum_{m=1}^M r_m(\mathbf{x}) \frac{\partial r_m(\mathbf{x})}{\partial x_i}$$

$$H_{ij}(\mathbf{x}) = 2 \sum_{m=1}^M \left(\frac{\partial r_m(\mathbf{x})}{\partial x_j} \frac{\partial r_m(\mathbf{x})}{\partial x_i} + r_m(\mathbf{x}) \frac{\partial^2 r_m(\mathbf{x})}{\partial x_i \partial x_j} \right)$$

Only the second term in the Hessian matrix computation involves expensive second-order partial derivatives. The *Gauss-Newton algorithm* follows the Newton's method, but completely *ignores the second term*, which is a valid approximation as long as the second term is considerably smaller compared to the first term:

$$\left| r_m(\mathbf{x}) \frac{\partial^2 r_m(\mathbf{x})}{\partial x_i \partial x_j} \right| \ll \left| \frac{\partial r_m(\mathbf{x})}{\partial x_j} \frac{\partial r_m(\mathbf{x})}{\partial x_i} \right|$$

We can rewrite the residual functions r_m into a vectorized version $\mathbf{r} = (r_1, \dots, r_M)^T$ and introduce a Jacobian matrix $\mathbf{J} \in \mathbb{R}^{M \times N}$:

$$J_{mi} = \frac{\partial r_m(\mathbf{x})}{\partial x_i}$$

This simplifies the equations for gradient \mathbf{g} and approximate Hessian $\tilde{\mathbf{H}}$, without considering the second-order derivative term, into the following Gauss-Newton formulation:

$$\mathbf{g} = 2\mathbf{J}^T \mathbf{r}$$

$$\tilde{\mathbf{H}} = 2\mathbf{J}^T \mathbf{J}$$

The update step in the Gauss-Newton algorithm then takes the following vectorized form:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - (\mathbf{J}^T(\mathbf{x}_n)\mathbf{J}(\mathbf{x}_n))^{-1}\mathbf{J}^T(\mathbf{x}_n)\mathbf{r}(\mathbf{x}_n)$$

One important difference with the Newton's algorithm is that there is no need for second-order partial derivatives, which results in significant computation savings. In certain optimization problems, the Jacobian matrix \mathbf{J} has a *block-wise sparse structure*, with a low number of non-zero sub-matrix blocks. We further exploit that and develop specialized sparse matrix-matrix and matrix-vector operations. Furthermore, instead of computing an expensive inverse of the matrix $(\mathbf{J}^T\mathbf{J})^{-1}$, a linear system is solved at each iteration to compute the parameter increment $\Delta\mathbf{x}_n = \mathbf{x}_{n+1} - \mathbf{x}_n$:

$$\mathbf{J}^T(\mathbf{x}_n)\mathbf{J}(\mathbf{x}_n)\Delta\mathbf{x}_n = -\mathbf{J}^T(\mathbf{x}_n)\mathbf{r}(\mathbf{x}_n)$$

Finally, you can observe that when the algorithm converges to the root, then the Jacobian matrix value is $\mathbf{J}(\mathbf{x}_n) = 0$, and the linear system above is ill-conditioned. To resolve the numerical instability, a small positive factor $\lambda > 0$ is added to diagonal elements of $\mathbf{J}^T\mathbf{J}$, and the following linear system is then solved in each Gauss-Newton iteration:

$$(\mathbf{J}^T(\mathbf{x}_n)\mathbf{J}(\mathbf{x}_n) + \lambda\mathbf{I})\Delta\mathbf{x}_n = -\mathbf{J}^T(\mathbf{x}_n)\mathbf{r}(\mathbf{x}_n) \quad (2.1)$$

While we usually keep the factor λ constant, it could also be beneficial to adapt it iteratively, resulting in a *Levenberg-Marquardt algorithm*.

2.2.3 Conjugate Gradient Descent

As described in the previous section, a major step of the Gauss-Newton algorithm is solving a linear system 2.1. More generally, for a matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ and vector $\mathbf{b} \in \mathbb{R}^N$, we would like solve for (increment) vector $\mathbf{x} \in \mathbb{R}^N$ that satisfies the linear system:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

To execute the algorithm fast, we need a very efficient linear system solver. While there exist many *direct linear solvers*, i.e. methods that provide an explicit solution for \mathbf{x} , it is hard to parallelize these solvers and run efficiently on a GPU. On the other hand, there exist an *iterative linear solver* that is highly parallelizable, called *conjugate gradient descent*.

Before going into more detail about the conjugate gradient descent, we have to mention an additional condition that needs to hold in order to use this linear system solver – the matrix \mathbf{A} has to be symmetric and *positive-definite*, i.e. for all non-zero $\mathbf{x} \in \mathbb{R}^N$ it follows:

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$$

In our case, $\mathbf{A} = \mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}$ with $\lambda > 0$, which is symmetric, i.e. $\mathbf{A}^T = \mathbf{A}$, and it's easy to show that the above condition holds for non-zero $\mathbf{x} \in \mathbb{R}^N$:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = \mathbf{x}^T (\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}) \mathbf{x} = (\mathbf{J} \mathbf{x})^T (\mathbf{J} \mathbf{x}) + \lambda (\mathbf{x}^T \mathbf{x}) = \|\mathbf{J} \mathbf{x}\|^2 + \lambda \|\mathbf{x}\|^2 > 0$$

Therefore, \mathbf{A} is also positive-definite in each iteration of the Gauss-Newton algorithm.

The main idea behind conjugate gradient descent is to iteratively find *conjugate directions* – these are vectors $\mathbf{p}_k \in \mathbb{R}^N$ that are orthogonal wrt. to matrix \mathbf{A} , i.e. $\mathbf{p}_k^T \mathbf{A} \mathbf{p}_l = 0$ for different $\mathbf{p}_k, \mathbf{p}_l$. The linear system solution \mathbf{x} can then be expressed as a linear combination of these conjugate directions. While we could extract all N conjugate directions, it turns out that it's often enough to only consider smaller number K of most significant vectors \mathbf{p}_k .

Algorithm 1 Conjugate Gradient Descent

```

1:  $\mathbf{d}_0 = \mathbf{b} - \mathbf{A} \mathbf{x}_0$ 
2:  $\mathbf{p}_0 = \mathbf{d}_0$ 
3: for  $k = 0, \dots, K - 1$  do
4:    $\alpha_k = \frac{\mathbf{d}_k^T \mathbf{d}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$ 
5:    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ 
6:    $\mathbf{d}_{k+1} = \mathbf{d}_k - \alpha_k \mathbf{A} \mathbf{p}_k$ 
7:   if  $\|\mathbf{d}_{k+1}\|$  is small then return  $\mathbf{x}_{k+1}$ 
8:    $\beta_k = \frac{\mathbf{d}_{k+1}^T \mathbf{d}_{k+1}}{\mathbf{d}_k^T \mathbf{d}_k}$ 
9:    $\mathbf{p}_{k+1} = \mathbf{d}_{k+1} + \beta_k \mathbf{p}_k$ 
10: return  $\mathbf{x}_K$ 

```

The algorithm is summarized in Alg. 1. We initialize the solution with $\mathbf{x}_0 = 0$, and set the initial conjugate direction to $\mathbf{p}_0 = \mathbf{b} - \mathbf{A} \mathbf{x}_0$, which represents the system residual \mathbf{d}_0 of the current solution \mathbf{x}_0 . The solution is iteratively updated for at most K iterations. In each iteration, we find an updated solution \mathbf{x}_{k+1} and a new conjugate direction \mathbf{p}_{k+1} . The detailed derivation of the algorithm can be found in [19]. If the system residual \mathbf{d}_k is small enough at any iteration, we can stop the algorithm early. To improve algorithm's convergence, we can make use of *preconditioning* by multiplying the linear system with a matrix \mathbf{C}^{-1} and instead solve the following system:

$$\mathbf{C}^{-1} \mathbf{A} \mathbf{x} = \mathbf{C}^{-1} \mathbf{b}$$

The preconditioner \mathbf{C}^{-1} should be symmetric and positive-definite, and the matrix $\mathbf{C}^{-1} \mathbf{A}$ should have a smaller *condition number* compared to \mathbf{A} . In the case of Gauss-Newton algorithm we often use the inverse of the diagonal values of \mathbf{A} as a preconditioner, i.e. $\mathbf{C} = \text{diag}(\mathbf{A})$.

As can be observed, the update steps consist of simple linear algebra operations, requiring only vector operations and application of matrix \mathbf{A} to different vectors. At each iteration, only the latest values of vectors \mathbf{x}_k , \mathbf{d}_k and \mathbf{p}_k need to be stored. This

allows for a very efficient and parallelizable algorithm implementation on the GPU. If matrix \mathbf{A} is sparse (or block-wise sparse), we can use an efficient sparse matrix-vector multiplication. Often it is expensive to explicitly compute the matrix $\mathbf{A} = \mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}$, it is faster to instead apply two matrix-vector multiplications, first with \mathbf{J} and second with \mathbf{J}^T . For some optimization problems it can even be beneficial to not even compute the transpose matrix \mathbf{J}^T , but instead re-generate $\mathbf{J}^T \mathbf{J}$ elements on-the-fly for each matrix-vector multiplication, as is described in [20].

2.2.4 Differentiable Optimization

In the previous sections we described algorithms for efficient solving of nonlinear least squares optimization problems, and many problems in computer vision fit well to this general formulation. However, the problem’s energy formulation and optimization solver parameters can have a large impact on the algorithm’s performance. For example, while Gauss-Newton offers fast convergence, often requiring only a few iterations, the converged point can also be a local minimum, especially when dealing with a highly non-convex energy. Improving the optimization energy landscape to be more convex-like is a crucial step when developing the computer vision algorithms. Furthermore, the convergence speed of an iterative conjugate gradient descent depends on the condition number of the linear system, and getting a better problem-specific preconditioner can make a big difference in terms of runtime.

While these problem-specific properties could be fine-tuned manually on a few problem examples, this can often lead to sub-optimal algorithms and overfitting. We can instead use a larger training corpus with many examples of the inputs and solutions for our problem, and optimize for optimal optimization problem and algorithm parameters automatically in a *data-driven manner*. For example, we might want to figure out the optimal conjugate gradient preconditioner for the problem of non-rigidly tracking the motion between two RGB-D frames, to improve the algorithm’s convergence on this very specific problem. As is proposed in [21], we can run a non-rigid alignment on a dataset of RGB-D frame pairs to acquire problem-specific set of linear systems, and solve these linear systems by running conjugate gradient for many iterations without preconditioner. Afterwards we train a network predicting an optimal preconditioner that solves the linear problem in much fewer iterations.

However, to make the data-driven optimization of certain energy or algorithm properties possible, the algorithm has to be *differentiable*. There exist many general frameworks that support a variety of linear algebra operations and offer *automatic differentiation*, such as PyTorch [22]. That makes it much easier to develop differentiable modules. Our optimization algorithms consist of a gradient vector \mathbf{g} and a Jacobian matrix \mathbf{J} construction, followed by relatively simple linear algebra operations. To make them differentiable, we can just port all required operations to a differentiable framework. This offers the flexibility of adapting any parts of the algorithm, if necessary.

Sometimes there could be benefits of doing the analytic derivative manually, without relying on the automatic differentiation. In cases where the algorithm properties don’t change during the iterations, e.g. the conjugate gradient descent doesn’t have any

iteration-dependent parameters, computing the analytic derivative of the linear solve operation can result in considerably faster computation and also a more numerically stable solution. More concretely, we solve the following linear system as part of our algorithm:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

After using the system’s solution \mathbf{x} in a computation of a certain loss \mathcal{L} , we would like to compute the partial derivatives of \mathcal{L} with respect to matrix \mathbf{A} and vector \mathbf{b} . Following the analytic derivative formulation in [23] (and described in more detail in App. D.2), the derivatives can be computed as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \mathbf{A}^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{x}} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \left(\mathbf{A}^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{x}} \right) \mathbf{x}^T = -\frac{\partial \mathcal{L}}{\partial \mathbf{b}} \mathbf{x}^T$$

This partial derivative formulation is not dependent on the number of iterations. Given a solution \mathbf{x} from the forward linear solve and its partial derivative $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$ wrt. the loss function \mathcal{L} , only a single additional linear system needs to be solved to directly compute the partial derivatives analytically, without requiring to backpropagate the gradients through conjugate gradient iterations.

2.3 Non-rigid Tracking

Assuming we know the shape of a deformable object, the task of non-rigid tracking is to *track the deformations* of the object through time, i.e. estimate the dense motion of the object. The observations come in a form of a video recording the moving object. While we could work with only an *RGB video* as input, most algorithms in this thesis rely on additional *depth video* input as well, making the tracking problem better constrained. Since affordable RGB-D sensors, such as Microsoft Kinect, are widely available, and already present on many smartphones, requiring a depth sensor doesn’t really limit the applications too much. We are mostly interested in the setting with a *single sensor*, however, the algorithms presented in this section can easily be applied to the *multi-view capture* setting, as we also showcase in some of our works.

2.3.1 Deformation Graph

The object shape is usually given in a form of a mesh. There are many ways to represent deformations of the object, ranging from very coarse representations to fine and detailed motion modeling. One extreme are *skeletons*, a sparse set of manually-designed joints and connecting bones for specific object classes, such as humans. They allow only a few degrees of freedom, also called articulations, and define dense motion as *interpolation* of joints’ motion. On the other end is dense motion estimation, known as *scene flow*, where motion of each mesh vertex is represented with a different flow vector, supporting a large variety of deformations. We decided to follow a generalized concept of *deformation*

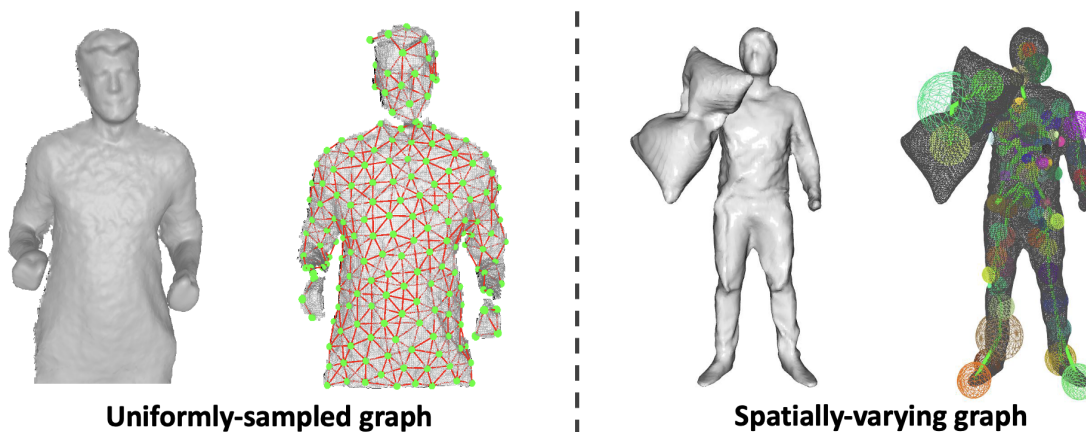


Figure 2.2: Deformation graph construction. Given a shape geometry, we define the deformation graph nodes and edges either by using uniform sampling and connecting nearest-neighbors (*left*), or by optimizing over a spatially-varying node positions, influence and connectivity, optimal for the entire video sequence (*right*).

graphs, where the motion representation can approach both extremes, depending on the density of the graph.

A deformation graph is a collection of *nodes* and *edges*, visualized in Fig. 2.2. Graph nodes define the degrees of freedom for the motion representation, and we estimate a rotation $\mathbf{R}_i \in \mathbb{R}^{3 \times 3}$ and translation $\mathbf{t}_i \in \mathbb{R}^3$ for each node $v_i \in \mathcal{V}$. This results in a set of transformations $\mathcal{T} = \{(\mathbf{R}_i, \mathbf{t}_i), i = 1, \dots, N\}$ that are optimized for every video frame, modeling the deformations of the object. We additionally note down the node’s *canonical position*, i.e. position in the mesh template, as $\mathbf{v}_i \in \mathbb{R}^3$. Graph edges $\mathcal{E} = \{e_k = (v_i, v_j)\}$ denote the connectivity between different nodes, and can enforce additional regularization, encouraging motion of neighboring nodes to be similar.

Graph construction. The deformation graph can be constructed by a *uniform sampling* of nodes over the template mesh, ensuring that every mesh vertex has at least one node in its ϵ -neighborhood (left in Fig. 2.2). A simple greedy algorithm is employed that incrementally samples new graph nodes until the entire mesh is covered. The graph edges can be defined as connections between node v_i with its nearest M spatial neighbors (commonly setting $M = 8$), and are *directional* – node v_i can be connected to node v_j , while node v_j doesn’t need to be connected to node v_i . We could use *Euclidean distance* for computing nearest neighbors, however more accurate connectivity is given by *geodesic distance*, considering the nearest paths between nodes on the surface. Instead of using a fixed ϵ -neighborhood, we can optimize for optimal influence radius $r_i \in \mathbb{R}$ for each node, depending on the motion observed in the video sequence (right in Fig. 2.2). Similarly, the connectivity between nodes the nodes can also be optimized, instead of defined purely on the distance in canonical template space. We will provide more details about optimizing a *spatially-varying graph* in the Chapter 5 about *neural deformation graphs*.

2.3.2 Motion Interpolation

There does not need to be a deformation node for every vertex in the mesh. Instead, we interpolate the motion of each vertex \mathbf{x} using the deformation estimates of its nearest K nodes, denoted as neighborhood $\mathcal{N}_{\mathbf{x}}$ of vertex \mathbf{x} (we usually use $K = 4$). For motion interpolation we follow the Embedded Deformation [24] formulation:

$$\mathbf{Q}(\mathbf{x}) = \frac{1}{W} \sum_{v_i \in \mathcal{N}_{\mathbf{x}}} w_i(\mathbf{x})(\mathbf{R}_i(\mathbf{x} - \mathbf{v}_i) + \mathbf{v}_i + \mathbf{t}_i)$$

The interpolation weights $w_i(\mathbf{x})$ are also called *skinning weights*, they model the influence of node v_i on the point \mathbf{x} . The factor W is defined as the sum over all skinning weights, i.e. $W = \sum_{v_i \in \mathcal{N}_{\mathbf{x}}} w_i(\mathbf{x})$, ensuring that the interpolation weights sum up to 1. The skinning weights could be defined in different ways, we use an exponential kernel that decays the node’s influence depending on the distance of point \mathbf{x} to the node position \mathbf{v}_i :

$$w_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{v}_i\|_2^2}{r_i^2}\right) \quad (2.2)$$

Here the radius r_i denotes the node-specific influence in the spatially-varying graph, and is set to a constant value $r_i = \epsilon$ if we use a uniformly-sampled graph.

2.3.3 Motion Regularization

The goal of non-rigid tracking is fitting to the video observations, which translates to estimating the node deformations that optimally align the template mesh to every RGB-D frame. However, there is often noise present in the input depth maps, and global correspondences extracted from RGB-D frames can also be inaccurate. To be able to fit to the RGB-D frames and at the same time be robust to noise present in observations, additional motion regularization is introduced. A common assumption introduced in [25] is that the non-rigid motion is locally *as-rigid-as-possible* (ARAP), which can be formulated as:

$$E_{\text{reg}}(\mathcal{T}) = \sum_{(v_i, v_j) \in \mathcal{E}} w_{ij} \|\mathbf{R}_i(\mathbf{v}_j - \mathbf{v}_i) + \mathbf{v}_i + \mathbf{t}_i - (\mathbf{v}_j + \mathbf{t}_j)\|_2^2$$

The edge weights w_{ij} can simply be set 1, i.e. $w_{ij} = 1$, or they can, similarly to the skinning weights, reflect the influence of node v_i to the node v_j :

$$w_{ij} = \exp\left(-\frac{\|\mathbf{v}_j - \mathbf{v}_i\|_2^2}{r_i^2}\right)$$

In the ARAP formulation we model the local motion relative to each node v_i with a rotation \mathbf{R}_i , which is often represented with 3-dimensional *axis-angle* parameters $\boldsymbol{\omega}_i \in \mathbb{R}^3$ [26]. An alternative *embedded deformation* (ED) formulation [24] allows for additional affine scaling as well, and instead uses affine matrices $\mathbf{A}_i \in \mathbb{R}^{3 \times 3}$ to model the nodes’

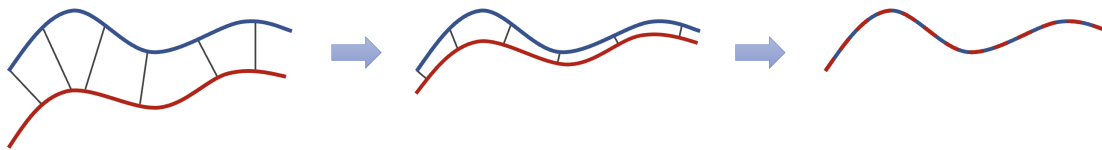


Figure 2.3: Iterative closest points. The template mesh (*red*) is iteratively aligned to the current point cloud (*blue*) by finding (new) closest point matches at each iteration.

deformation. However, to limit the degrees of freedom, an additional energy term is added that encourages the affine matrices $\mathbf{A}_i \in \mathbb{R}^{3 \times 3}$ to be close to rotation matrices. If we decompose each affine matrix into its columns, i.e. $\mathbf{A}_i = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$, then different columns are supposed to be orthogonal to each other and the length of each column should be equal to 1:

$$E_{\text{rot}}(\mathcal{T}) = \sum_{v_i \in \mathcal{V}} ((\mathbf{a}_1 \cdot \mathbf{a}_2)^2 + (\mathbf{a}_1 \cdot \mathbf{a}_3)^2 + (\mathbf{a}_2 \cdot \mathbf{a}_3)^2 + (\mathbf{a}_1 \cdot \mathbf{a}_1 - 1)^2 + (\mathbf{a}_2 \cdot \mathbf{a}_2 - 1)^2 + (\mathbf{a}_3 \cdot \mathbf{a}_3 - 1)^2)$$

In this case, the complete motion regularization energy is reformulated into the following:

$$E_{\text{reg}}(\mathcal{T}) = \sum_{(v_i, v_j) \in \mathcal{E}} w_{ij} \|\mathbf{A}_i(\mathbf{v}_j - \mathbf{v}_i) + \mathbf{v}_i + \mathbf{t}_i - (\mathbf{v}_j + \mathbf{t}_j)\|_2^2 + E_{\text{rot}}(\mathcal{T})$$

2.3.4 Iterative Closest Points

A core part of the non-rigid tracking algorithms are the *data terms*, i.e. energy terms that measure the quality of fitting the template mesh to the input observations. When we have a depth map as input, a very important data term are *iterative closest points* (ICP) [27]. Let's assume we have successfully aligned the template mesh to the previous frame. To non-rigidly track the current frame, we backproject the depth map at the current frame into 3D, resulting in a 3D point cloud. The template mesh is matched to the current point cloud by finding nearest point-to-point correspondences between the mesh and the point cloud. We denote the set of correspondences as $\mathcal{C} = \{(\mathbf{p}, \mathbf{c})\}$, with \mathbf{p} as the point on the template mesh and \mathbf{c} its nearest Euclidean neighbor in the current point cloud. The goal is to minimize the point-to-point distance between the matches:

$$E_{\text{point2point}}(\mathcal{T}) = \sum_{(\mathbf{p}, \mathbf{c}) \in \mathcal{C}} \|\mathbf{Q}(\mathbf{p}) - \mathbf{c}\|_2^2$$

The point-to-point matches are visualized in Fig. 2.3. It can be observed that the initial point-to-point correspondences are not matching the corresponding surface points exactly. That is why the nearest point-to-point matches are *iteratively* refined at each optimization iteration, resulting in better matches after initial non-rigid alignment. Furthermore, we also take special care to *filter outlier correspondences*. The point-to-point matches are kept only if they don't exceed the predefined maximum point-to-point dis-

tance, maximum angle difference between the current point cloud normals and template mesh normals, and the maximum difference between the current view direction and the mesh normals (the latter removes unstable correspondences at the boundaries).

There is another data term that is usually used in combination with point-to-point energy. We additionally compute the template mesh normal $\mathbf{n}_{\mathbf{p}} \in \mathbb{R}^3$ at every correspondence point \mathbf{p} (using the latest surface deformations), and minimize the point-to-point distance projected along the surface normal. The resulting term is called point-to-plane energy, and it experiences a more robust alignment performance also in the case of less accurate matches:

$$E_{\text{point2plane}}(\mathcal{T}) = \sum_{(\mathbf{p}, \mathbf{c}) \in \mathcal{C}} (\mathbf{n}_{\mathbf{p}}^T (\mathbf{Q}(\mathbf{p}) - \mathbf{c}))^2$$

The total ICP energy consists of both point-to-point and point-to-plane data terms, and we balance the contributions of both terms (by setting the $\lambda_{\text{point2point}} = 0.1$ and $\lambda_{\text{point2plane}} = 1.0$):

$$E_{\text{icp}}(\mathcal{T}) = \lambda_{\text{point2point}} E_{\text{point2point}}(\mathcal{T}) + \lambda_{\text{point2plane}} E_{\text{point2plane}}(\mathcal{T})$$

Efficient projective ICP. Since we compute new ICP correspondences at each optimization iteration, we need a very efficient ICP matching implementation for real-time tracking performance. While there exist acceleration structures for computing nearest point matches, such as a *k-d tree*, their performance is not fast enough. Instead, we approximate the ICP matches using *projective correspondences*, i.e. we project each mesh point to the depth map and assign the queried depth pixel as the match. This can be done very efficiently using the graphics rendering pipeline, and the projective matches are further refined by checking all depth pixels in a local 2D pixel window around the projection.

2.3.5 Global Correspondences

While ICP correspondences are often very effective, they can only recover non-rigid deformations accurately if frame-to-frame motion is not too large. In the case of fast motion, or if certain object parts are occluded for a while, and then observed again later, the frame-to-frame deformation change can become arbitrarily large. For robust non-rigid tracking we therefore additionally require more *global non-rigid correspondences*. Assuming we can extract global matches between the template mesh and the current frame, resulting in a set of global correspondences $\tilde{\mathcal{C}} = \{(\tilde{\mathbf{p}}, \tilde{\mathbf{c}})\}$, then the data term is actually formulated very similar to the ICP data term:

$$E_{\text{gc}}(\mathcal{T}) = \sum_{(\tilde{\mathbf{p}}, \tilde{\mathbf{c}}) \in \tilde{\mathcal{C}}} w_{\tilde{\mathbf{c}}}^2 \|\mathbf{Q}(\tilde{\mathbf{p}}) - \tilde{\mathbf{c}}\|_2^2$$

Beside the point-to-point distance, we use an additional *confidence weight* $w_{\tilde{\mathbf{c}}}$ that can decrease the influence of global matches with lower confidence.

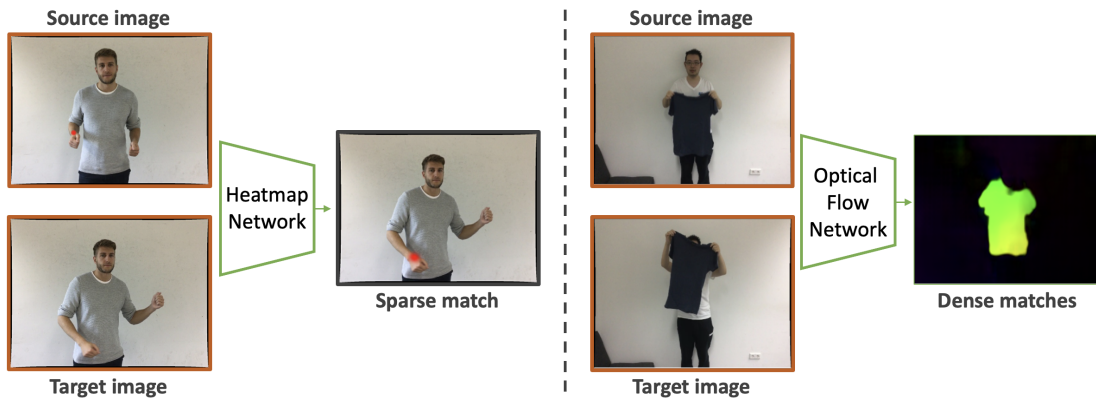


Figure 2.4: Global correspondences. We can either estimate sparse correspondences for selected pixels via heatmap prediction (*left*), or densely predict correspondences for all pixels using an optical flow network (*right*).

There exist many approaches for finding global non-rigid correspondences, and in this thesis we propose a few novel approaches as well. In Fig. 2.4 we illustrate two common choices: we can predict a match independently for each selected pixel, resulting in sparse correspondences [28]–[32], or we directly predict dense pixel correspondences using an optical flow network [33], [34]. The matching methods can operate on 3D point clouds [30], [32], RGB-D frames [28], or only 2D RGB images [29], [31], [33], [34], however we can always formulate the point-to-point distance by lifting the matches into 3D using depth maps, if necessary. If the method additionally provides the confidence measures for different matches, we use these values as confidence weights, otherwise we use the default constant value of $w_{\tilde{z}} = 1$.

2.3.6 Optimization Energy

To non-rigidly align the template mesh to each input frame of the video, we jointly optimize both the data and regularization terms, described in the previous sections. Importantly, we only optimize the graph nodes’ transformations \mathcal{T} , we keep the mesh shape fixed. The total non-rigid tracking energy takes the following form:

$$E(\mathcal{T}) = \lambda_{\text{reg}} E_{\text{reg}}(\mathcal{T}) + \lambda_{\text{icp}} E_{\text{icp}}(\mathcal{T}) + \lambda_{\text{gc}} E_{\text{gc}}(\mathcal{T})$$

If you carefully examine all energy terms, you can notice that each term follows the non-linear least squares formulation. That means that we can apply the previously described Gauss-Newton algorithm to minimize the optimization energy. This enables fast convergence and highly efficient implementation of solver iterations, resulting in real-time non-rigid tracking performance.

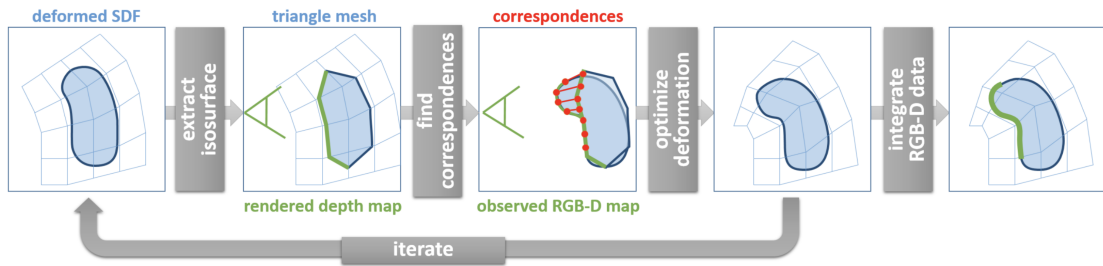


Figure 2.5: Incremental non-rigid reconstruction. At each frame we extract the latest mesh from the grid-based SDF representation, compute correspondences with RGB-D frame and optimize the deformations, and finally integrate the newly observed RGB-D data to complete the SDF representation (figure is from [35]).

2.4 Non-rigid Reconstruction

Non-rigid tracking plays a core part in modeling deformable objects. However, it assumes a shape geometry is already estimated, and tracks the deformations given the template mesh. In many applications, it’s hard to obtain object shape prior to capturing the video of its motion, and we need to reconstruct the object’s shape at the same time as track its motion. This task is known as *non-rigid reconstruction*. In this section we will describe existing non-rigid reconstruction approaches, starting with *non-rigid volumetric fusion*, which can be applied to any deformable object and supports real-time performance. It doesn’t make use of any data-driven priors though, which could help with more robust reconstruction performance in underconstrained settings. Therefore, we also look into *neural shape representations*, which provide differentiable modules that can efficiently learn reconstruction priors from 3D shape datasets. We are especially interested in shape reconstruction from a video, that is why we go into more detail about *learning shape fusion* from video observations.

2.4.1 Non-rigid Volumetric Fusion

Instead of assuming a complete shape reconstruction is given, we can use a partial shape reconstruction for non-rigid tracking, and incrementally update the shape as new RGB-D observations are available. This idea was introduced in DynamicFusion [4], a real-time non-rigid reconstruction approach, consisting of several important modules illustrated in Fig. 2.5. Since we have no shape template at the start of the recording, the initial shape is generated directly from the initial depth map, resulting in a partial mesh. Finding correspondences between the partial shape mesh and the observed RGB-D map, and optimizing the deformations, are core modules of non-rigid tracking, described in detail in the previous section. Finally, integrating the newly observed RGB-D data using the *non-rigid volumetric fusion* is what makes it possible to gradually complete the shape reconstruction. These steps are repeated for every frame in the video sequence, and

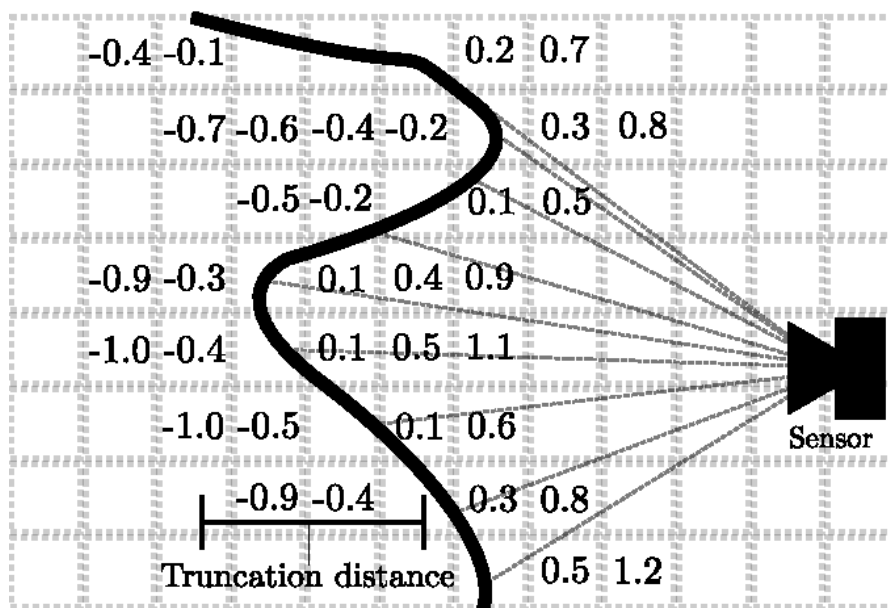


Figure 2.6: Volumetric fusion. We project a dense grid of voxels to the depth map, and compute projective depth distances, positive in front of the surface and negative behind it (figure is from [36]).

at the end of the video we have as complete shape reconstruction as possible, with its estimated deformations throughout the sequence.

Volumetric fusion. We start with a static volumetric fusion of depth frames, as proposed in [37] and further extended in [2], since it’s an important building block of non-rigid volumetric fusion. The shape surface is represented by a *signed distance field* (SDF), with values stored in a dense voxel grid. Beside SDF values we also store a *weight* value for each voxel that keeps track of all already fused observations, and is initialized with zero. For each input frame i , every voxel is projected to the depth camera view, and the voxel’s depth value d_v in current view is compared with the queried depth value d_m from the depth map. The distance between the values, $\delta_i = d_m - d_v$, is known as the *projective distance*. Given the voxel’s existing SDF D_{i-1} and weight W_{i-1} values from the grid, the new observation is fused as:

$$D_i = \frac{D_{i-1}W_{i-1} + \delta_i w_i}{W_{i-1} + w_i}$$

$$W_i = W_{i-1} + w_i$$

The observation weight w_i could be set in a way to reflect the quality of the observation, e.g. depending on the view direction, but we usually just keep it constant as $w_i = 1$. An example of fused SDF is visualized in Fig. 2.6. In the region behind the surface, we

only fuse depth observations in a smaller region, up to a predefined *truncation distance*, since otherwise we could corrupt a different surface occluded in the current view. Furthermore, we also cap the maximum positive distance values $\delta_i = \min(d_m - d_v, \delta_{\max})$, since projective distances are only an approximation of the true signed distances with larger errors further away from the surface.

Non-rigid shape warping. Extending the volumetric fusion to non-rigid setting is actually quite straight-forward. The dense SDF grid is fused in the *canonical frame*, i.e. the initial frame of the video. After tracking a newly observed RGB-D frame using a partial shape, we similarly want to integrate new observations into the canonical grid. Instead of directly projecting the voxel position $\mathbf{v} \in \mathbb{R}^3$ to the current camera view, we apply the estimated warping from canonical to current frame and project the deformed voxel position $Q(\mathbf{v})$. We can only warp voxels that are close to the surface, since that is where the graph nodes are defined, up to *one-ring neighborhood* around the surface (usually set to 2ϵ , with ϵ being the coverage radius used for graph construction). Beside this additional voxel filtering and warping, the fusion process proceeds in the same way as in the static setting. After every non-rigid fusion step, we extract a more complete mesh via Marching cubes [18], which is then non-rigidly aligned to the next frame.

Dealing with topology changes. The shape is reconstructed in a canonical space, i.e. the first frame of the sequence. During video capture, a *topology change* can occur between the first frame and later frames, e.g. two shape parts tightly in contact in canonical space become disconnected, or more concretely, a person takes off his coat. To properly reconstruct the shape and model the motion, we would need to extract independent surfaces for the coat and the underlying human body. However, due to the limited SDF grid resolution, this is often very hard. One simple change to the incremental fusion can make it possible to overwrite problematic surface regions, where topology changes occur and tracking fails, with the latest geometry updates – we can update the fusion weights in a windowed manner, i.e. $W_i = \min(W_{i-1} + w_i, W_{\max})$. This resets the canonical shape to keep up with the latest state, but unfortunately cannot recover deformation tracking throughout the whole video anymore. In Chapter 5 we will present an alternative shape representation that doesn't assume a fixed canonical space in a (specific) video frame, and can model topology changes better.

Extending deformation graph. After updating the shape geometry, we also need to update the corresponding deformation graph. Similarly to the initial graph construction, where we proceed in a greedy fashion and sample new graph nodes until the entire mesh is covered, we can add new nodes after the mesh gets more complete. If *geodesic distances*, i.e. distances on the mesh, are used to determine the graph connectivity, then graph edges also need to be updated in surface regions that were updated in the latest iteration. Lastly, in case of topology changes when we over-write the older version of the surface, the outdated graph nodes, i.e. nodes too far away from the latest mesh surface, need to be removed.

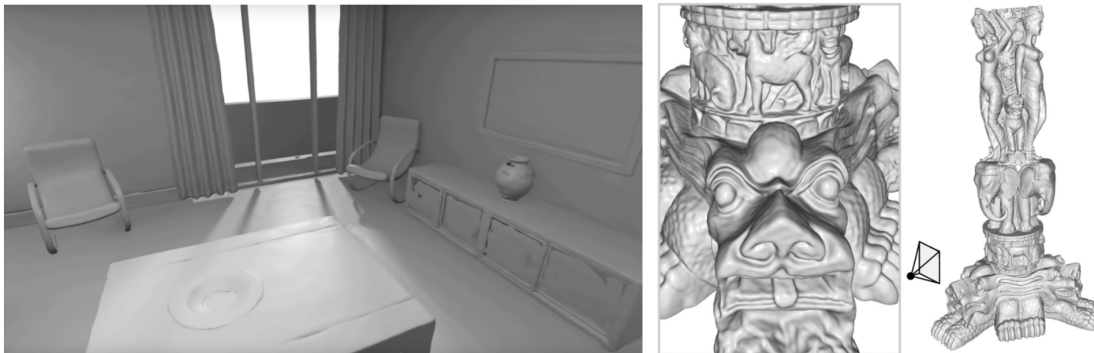


Figure 2.7: MLP-based reconstructions. Some examples of SIREN [39] reconstructions, where an MLP stores the SDF value for any continuously queried point.

2.4.2 Neural Shape Representations

The non-rigid volumetric fusion of RGB-D frames is a general non-rigid reconstruction method that can be applied to any deformable shape, but it doesn't rely on any priors for reconstruction. In case the object is not completely captured, e.g. certain object parts are never observed, the resulting reconstruction will always be partial. On the other hand, there exist reconstruction approaches that learn shape priors from datasets of 3D shapes, which enables them to predict complete reconstruction also in underconstrained settings, even from a single image. Having a more complete shape reconstruction also makes non-rigid tracking more robust, which is another reason for exploring data-driven priors for shape reconstruction. Finally, if we want to develop a non-rigid reconstruction approach that takes only an RGB video as input, without any depth information, then shape reconstruction priors are a necessity for reliable non-rigid reconstruction performance in that extremely underconstrained scenario [38].

Implicit MLP-based representation. When trying to learn data-driven priors, *neural networks* offer a very effective way to represent a large training corpus for a variety of tasks, including shape reconstruction. It turns out that a *multi-layer perceptron* (MLP), a sequence of linear layers and nonlinear activations, is very suitable for surface modeling [40]–[43]. Taking as input a 3D position, the MLP directly outputs an SDF or occupancy value at that location. This *continuous representation* doesn't suffer from grid discretization, and is very good at compressing the geometry information, requiring only a few MB for a detailed scene. In Fig. 2.7 we visualize some reconstruction results using SIREN [39] architecture, where *sine* activations are used.

Part-based multi-MLP representation. While an MLP requires much less memory compared to dense grids, each SDF evaluation requires a forward pass through the network. For larger scenes or more detailed reconstructions, the MLP has to become quite large to retain all the geometry details, resulting in slow SDF queries. Alternatively,

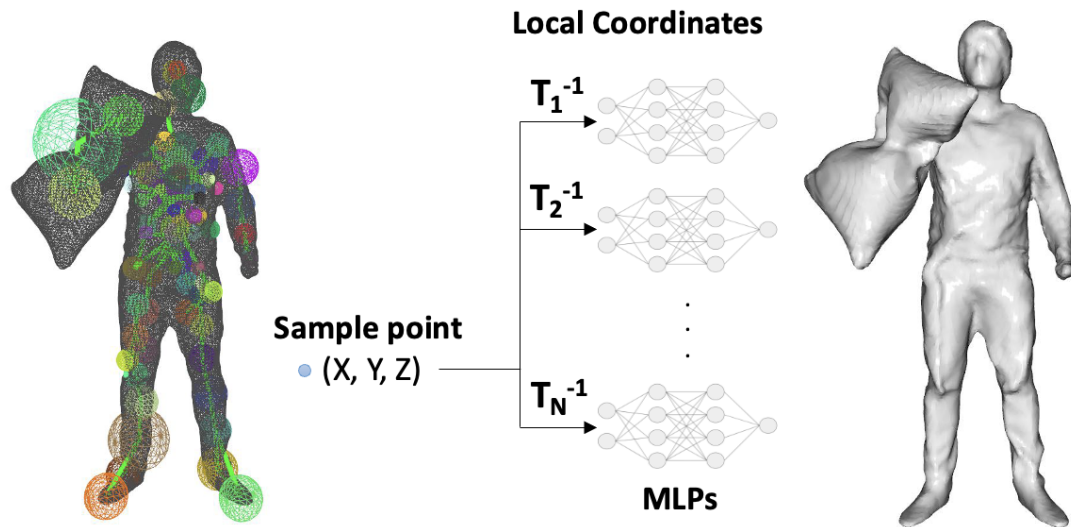


Figure 2.8: Multi-MLP object representation. Each MLP stores a corresponding transformation, which maps from local MLP-specific space to the global world space. A point is transformed to every local space, evaluated by a small MLP, and the MLP contributions are aggregated via a (weighted) sum.

we can use a set of smaller MLPs, with each MLP specialized to model a certain part of the object or scene. We can represent parts of static objects, but it can also be used to model dynamic objects, such as humans [44], [45], or even general deformable objects, as we will describe in more detail in Chapter 5. When it comes to deformable objects, each MLP has a corresponding transformation that maps from a local MLP-specific space to the global world space. That effectively eliminates the need for a global canonical space, each part can be reconstructed in its own local space, which offers more robustness in handling topology changes.

Shape generalization. MLP-based shape representations have its benefits in terms of reconstruction fidelity and surface compression. However, they are still optimized from scratch for every single object, and don't generalize at all across different objects. Our motivation for using neural networks was to find a differentiable representation that can be used for learning shape reconstruction priors. It turns out that it's relatively easy to extend the MLP representations to generalize across shapes. A simple example is DeepSDF [40], where an additional shape code is added as input to the MLP, beside the 3D position. A shared MLP is then trained on a large-scale dataset of objects (e.g. [46], [47]), and only the shape code is changing between different objects. When reconstructing a novel object at test-time, we can directly optimize the shape code, while keeping the MLP parameters frozen, preserving all shape priors learned during training. A code is often replaced by a coarse grid of features, which improves the

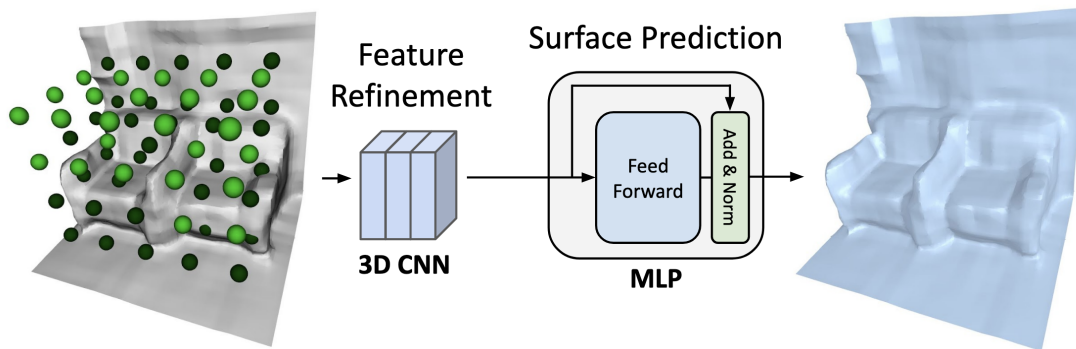


Figure 2.9: Feature-based MLP reconstruction. Shape is reconstructed using an (object-specific) feature grid and a (shared) MLP decoder. Additional networks such as a 3D CNN are often employed to improve generalization.

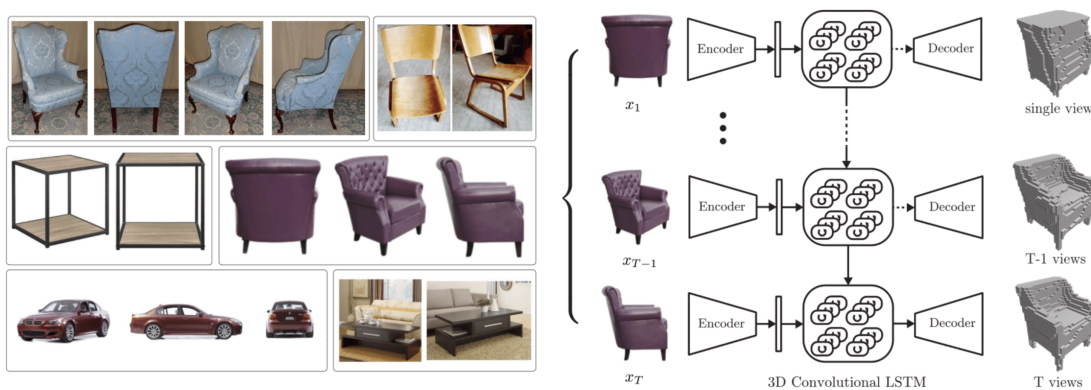


Figure 2.10: Overview of 3D-R2N2 [49]. A multi-view reconstruction approach that encodes RGB frames into global codes that are fused across frames via an LSTM network, without considering input camera poses.

representation power and makes it possible to model shape with very different geometry. A common architecture is illustrated in Fig. 2.9, where an (object-specific) feature grid is additionally refined by a (shared) 3D CNN before applying a (shared) MLP decoder for surface prediction. Another option, beside using code- or feature-conditioned MLPs, is to use a specialized network, called a *hyper-network*, to generate object-specific MLP parameters [48]. While it’s an interesting alternative, it often requires longer training times for the same reconstruction quality.

2.4.3 Learning Shape Fusion

The neural shape representations presented in the previous section are well-suited for learning reconstruction priors, and they can be used for shape prediction from a variety of different inputs. For example, there exist quite some approaches that predict object

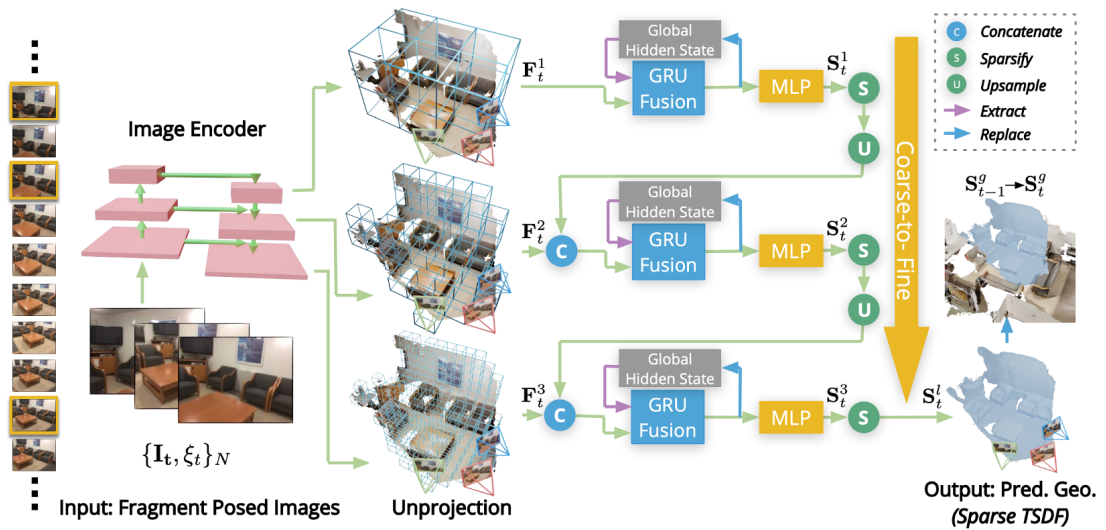


Figure 2.11: Overview of NeuralRecon [55]. A 3D reconstruction approach that encodes multi-view RGB frames into coarse-to-fine feature grids, aligned using camera poses, and fused across time using GRU units.

geometry from a *single image*, either using only an RGB image [50], [51], a depth image [41], [52], [53], or an RGB-D image [54]. However, single image shape prediction is a challenging task, and therefore requires extensive amounts of training data to build strong reconstruction priors. This is why existing approaches mostly showcase results on humans or limited object classes. In our case, we always get a video as input, and we want to use all the available information from input frames for shape reconstruction. We are interested in exploring RGB videos as inputs, without any depth sensor, since this is the scenario where non-rigid reconstruction without shape priors is extremely challenging.

Beside a generalizable MLP architecture for shape reconstruction, there is usually another component that is used to generate codes or feature grids for a specific input – an *encoder network*. This is often a convolutional network, and in the case of RGB images, it takes each image as input and predict either a single global feature, or pixel-aligned features using a 2D U-net architecture [56]. The main difference with single-view prediction approaches is that we want to fuse these features across video frames. There are different ways how to formulate fusion with a network. Recurrent neural networks (RNNs), such as long-short-term memory units (LSTMs) or gated recurrent units (GRUs), can be used to fuse sequence data. One of the early approaches for object reconstruction from images, 3D-R2N2 [49], uses a convolutional LSTM to fuse the global frame codes into a shared representation. The fused feature grid is then decoded using a 3D CNN to produce a coarse shape reconstruction, as visualized in Fig. 2.10.

The 3D-R2N2 [49] method doesn't take any additional motion information as input, such as a camera viewpoint or object deformations. This has an advantage that the ap-

proach can perform shape reconstruction independently from object tracking. However, it also means that the network has to resolve the camera motion implicitly, which is a hard task and might require a lot of training samples for high-quality reconstructions. Similarly to how we incrementally reconstruct and track the deformable objects in the RGB-D setting, we could explore ways to develop a similar iterative strategy when given only an RGB video. Therefore, we could assume that motion tracking is given as an additional input for shape reconstruction from an RGB video. There have been a variety of object and scene reconstruction approaches that take advantage of camera poses being given as input [55], [57]–[61]. They take the encoded pixel-aligned features and unproject them along the pixel rays into 3D feature grids, using the camera transformations to warp the rays into a shared world coordinate system. In Fig. 2.11 we show an overview of one of the methods, NeuralRecon [55], which uses GRU units to further fuse these already aligned features across time. Since the pixel rays are correctly aligned in 3D, a smaller fusion network can be used, resulting in real-time 3D reconstruction performance. Additionally, having aligned features as input makes it an easier task for the surface decoder, which showcases impressive reconstruction results on large-scale scenes.

Part II

Non-rigid Reconstruction using Data-driven Priors

3 Learning Non-rigid Reconstruction

Abstract of paper. Applying data-driven approaches to non-rigid 3D reconstruction has been difficult, which we believe can be attributed to the lack of a large-scale training corpus. Unfortunately, existing methods fail for important cases such as highly non-rigid deformations. We first address this problem of lack of data by introducing a novel semi-supervised strategy to obtain dense inter-frame correspondences from a sparse set of annotations. This way, we obtain a large dataset of 400 scenes, over 390,000 RGB-D frames, and 5,533 densely aligned frame pairs; in addition, we provide a test set along with several metrics for evaluation. Based on this corpus, we introduce a data-driven non-rigid feature matching approach, which we integrate into an optimization-based reconstruction pipeline. Here, we propose a new neural network that operates on RGB-D frames, while maintaining robustness under large non-rigid deformations and producing accurate predictions. Our approach significantly outperforms existing non-rigid reconstruction methods that do not use learned data terms, as well as learning-based approaches that only use self-supervision.

3.1 Introduction

Non-rigid 3D reconstruction, i.e., the dense, space-time coherent capture of non-rigidly deforming surfaces in full temporal correspondence, is key towards obtaining 3D abstractions of the moving real world. The wide availability of commodity RGB-D sensors, such as the Microsoft Kinect or Intel Realsense, has led to tremendous progress on static scene reconstruction methods. However, robust and high-quality reconstruction of non-rigidly moving scenes with one depth camera is still challenging. Applications for real-time non-rigid reconstruction range from augmented (AR) and virtual reality (VR) up to building realistic 3D holograms for fully immersive teleconferencing systems. The seminal DynamicFusion [4] approach was the first to show dynamic non-rigid reconstruction in real-time. Extensions primarily differ in the used energy formulation. Some methods use hand-crafted data terms based on dense geometry [4], [62], [63], dense color and geometry [64], [65], and sparse feature constraints [35]. Other approaches leverage multi-camera RGB-D setups [66], [67] for higher robustness. However, there are very few reconstruction methods that use learning-based data terms for general real-world scenes rather than specific scenarios [68], and that are trained to be robust under real-world appearance variation and difficult motions. One reason for this is the lack of a large-scale training corpus. One recent approach [69] proposes self-supervision for ground truth generation, i.e., they employ DynamicFusion [4] for reconstruction and train a non-rigid correspondence descriptor on the computed inter-frame correspondences. However, we show

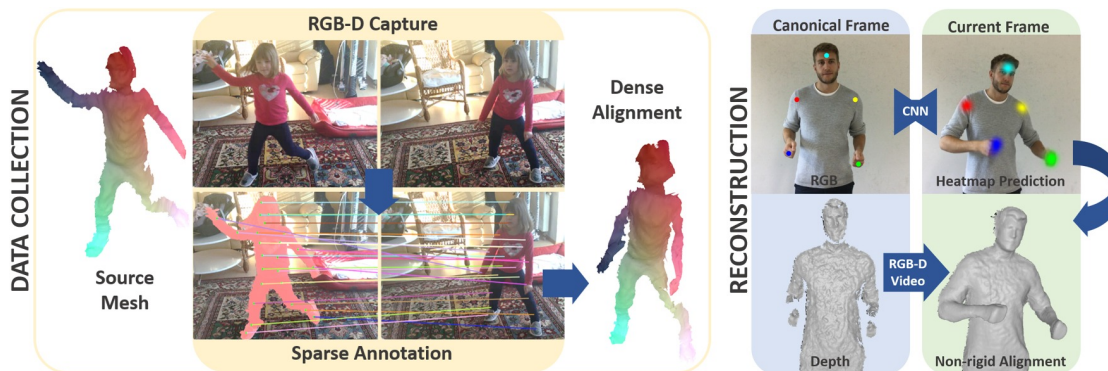


Figure 3.1: Approach overview. We propose a semi-supervised strategy combining self-supervision with sparse annotations to build a large-scale RGB-D dataset of non-rigidly deforming scenes (400 scenes, 390,000 frames, 5,533 densely aligned frame pairs). With this data, we propose a new method for non-rigid matching, which we integrate into a non-rigid reconstruction approach.

that existing non-rigid reconstruction methods are not robust enough to handle realistic non-rigid sequences; therefore this tracking-based approach does not scale to training data generation for real world scenes. Unfortunately, this means that self-supervision exactly fails for many challenging scenarios such as highly non-rigid deformations and fast scene motion. By design, this approach trained with self-supervision cannot be better than the employed tracker. We propose to employ semi-supervised training data by combining self-supervision with sparse user annotations to obtain dense inter-frame correspondences. The annotated sparse point correspondences guide non-rigid reconstruction; this allows us to handle even challenging motions. The result is a large dataset of 400 scenes, over 390,000 RGB-D frames, and 5,533 densely aligned frame pairs. Based on this novel training corpus, we develop a new non-rigid correspondence matching approach (see Sec. 3.3) that finds accurate matches between RGB-D frames and is robust to difficult real world deformations. We further propose a re-weighting scheme that gives more weight to corner cases and challenging deformations during training. Given a keypoint in a source frame, our approach predicts a probability heatmap of the corresponding location in the target frame. Finally, we integrate our learned data term into a non-rigid reconstruction pipeline that combines learned heatmap matches with a dense RGB-D reconstruction objective. In addition, we introduce a new benchmark and metric for evaluating RGB-D based non-rigid 3D correspondence matching and reconstruction. We extensively compare our new data-driven approach to existing hand-crafted features. We also integrate the learned features into a non-rigid reconstruction framework, leading to significant improvement over state of the art. In sum, our contributions are:

- A semi-supervised labeling approach for dense non-rigid correspondence learning, resulting in a dataset featuring 400 annotated dynamic RGB-D sequences and 5,533 densely aligned frame pairs.



Figure 3.2: Example frames of the collected dataset. Our large-scale dataset contains a large variety of dynamic sequences with segmentation masks and point correspondences between different RGB-D frames.

- A novel data-driven non-rigid correspondence matching strategy that leads to more robust correspondence estimation compared to the state-of-the-art hand-crafted and learned descriptors, especially in the case of extreme deformations.
- A non-rigid reconstruction approach for general scenes that combines learned and geometric data-terms and handles significantly faster and more complex motions than the state-of-the-art.

3.2 Related Work

Our approach is related to several research areas, such as volumetric 3D scene reconstruction, non-rigid object tracking, and learned correspondence matching. We focus our discussion on the most related RGB-D based techniques. For a detailed discussion, we refer to the recent survey [70].

Volumetric Scene Reconstruction. Reconstructing static environments with a single RGB-D sensor has had a long history in vision and graphics, including KinectFusion [2], which employs a uniform voxel grid to represent the scene as a truncated signed distance function (TSDF) [37], as well as many extensions to large-scale scenes [71]–[74]. These techniques track the 6-DoF camera motion by solving a geometric model-to-frame alignment problem using a fast data-parallel variant of the point-to-plane Iterative Closest Point (ICP) algorithm [27]. Globally consistent reconstruction based on Bundle Adjustment [75], [76] was for a long time only possible offline; data-parallel solvers now enable real-time frame rates [77]. An alternative to TSDFs are point-based scene representations [78]–[80]. Recent techniques also employ non-rigid registration to robustly handle loop closures [81], [82].

Non-Rigid Reconstruction. The reconstruction of general non-rigidly deforming objects based on real-time scan data has a long tradition [83]. One class of methods uses pre-defined templates, e.g., human templates, to capture pose and time-varying shape of clothed humans from RGB-D [84] or stereo camera data [85]. First template-less approaches had slow offline runtimes and only worked for slow and simple motions. The approach of [86] solves a global optimization problem to reconstruct the canonical shape of a non-rigidly deforming object given an RGB-D video sequence as input, but does not recover the time-dependent non-rigid motion across the entire sequence. The first approach to demonstrate truly dynamic reconstruction of non-rigid deformation and rest shape in real-time was DynamicFusion [4]. Since this seminal work, many extensions have been proposed. VolumeDeform [35] improves tracking quality based on sparse feature alignment. In addition, they parameterize the deformation field based on a dense volumetric grid instead of a sparse deformation graph. The KillingFusion [62] and SobolevFusion [63] approaches allow for topology changes, but do not recover dense space-time correspondence along the complete input sequence. Other approaches jointly optimize for geometry, albedo, and motion [64] to obtain higher robustness and better quality. The approach of Wang et al. [65] employs global optimization to minimize surface tracking errors. In contrast to these methods using a single RGB-D camera, other techniques use multiple color [87], [88] or depth cameras [66], [67], [89], [90], which enables high-quality reconstruction at the cost of more complex hardware. We propose a new learning-based correspondence matching and reconstruction approach that outperforms existing techniques.

Learning Rigid Correspondence Matching. Historically, correspondence matching for the task of rigid registration has been based on hand-crafted geometry descriptors [30], [91]–[94]. If color information is available in addition to depth, SIFT [29] or SURF [31] can be used to establish a sparse set of feature matches between RGB-D frames. More recently, 2D descriptors for feature matching in static scenes have been learned directly from large-scale training corpora [95]–[99]. The Matchnet [97] approach employs end-to-end training of a CNN to extract and match patch-based features in 2D image data. Descriptors for the rigid registration of static scenes can be learned and matched

directly in 3D space with the 3DMatch [32] architecture. Visual descriptors for dense correspondence estimation can be learned in a self-supervised manner by employing a dense reconstruction approach to automatically label correspondences in RGB-D recordings [69]. Descriptor learning and matching for static scenes has been well-studied, but is lacking in the challenging non-rigid scenario. While class-specific dense matching of non-rigid scenes has been learned for specific object classes [68], [100], none of these techniques can handle arbitrary deforming non-rigid objects. We believe one reason for this is the lack of a large-scale training corpus. In this work, we propose such a corpus and demonstrate how non-rigid matching between depth images can be learned end-to-end.

RGB-D Datasets. While we have seen a plethora of RGB-D datasets for static scenes, such as NYU [101], SUN RGB-D [102], and ScanNet [17], the largest of which have thousands of scans, non-rigid RGB-D datasets remain in their infancy. While these datasets can be used to pretrain networks for the task of non-rigid correspondence matching, they do not capture the invariants that are useful for the much harder non-rigid setting, and thus lead to sub-par accuracy. Current non-rigid reconstruction datasets are far too small and often limited to specific scene types [87], [89], which is not sufficient to provide the required training data for supervised learning. The datasets that provide real-world depth recordings [35], [103] do not come with ground truth reconstructions, which makes objectively benchmarking different approaches challenging. Other datasets that are commonly used for evaluation do not provide real-world depth data, e.g., [87], [88]. In this work, we introduce the first large-scale dataset for non-rigid matching based on semi-supervised labeling and provide a benchmark enabling objective comparison of different approaches.

3.3 Data-driven Non-Rigid Matching

Our goal is to find matches between a source and a target RGB-D frame. To this end, we propose a network architecture for RGB-D matching based on a Siamese network [104] with two towers. Input to the network are two local patches of size 224×224 pixels each (with 3 color channels and 3-dimensional points in camera coordinate space). We assume that the source patch is centered at a feature point.

Heatmap. The goal is to predict a probability heatmap \mathcal{H} in the target frame that gives the likelihood of the location of the source point. First, we compute a *sigmoid-heatmap*:

$$\mathcal{H}_{\text{sg}} = \sigma_{\text{sg}}(\mathbf{H}(D_{\text{out}}))$$

It is computed based on a sigmoid activation σ_{sg} to map responses to $[0, 1]$. Here, D_{out} is the output feature map of the last layer of the decoder and \mathbf{H} is a convolutional layer converting feature space into heatmap values. This is equivalent to independent binary

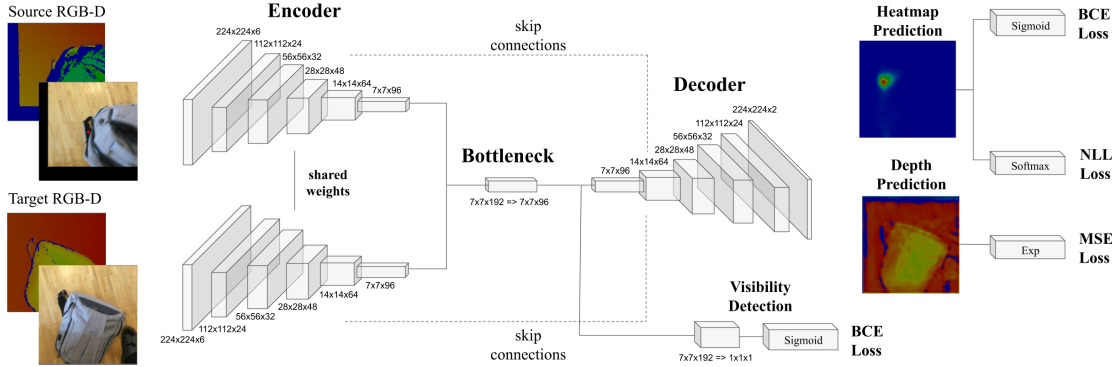


Figure 3.3: Network architecture. We devise an end-to-end architecture for RGB-D matching based on a Siamese network to find matches between a source and a target frame. Our network is based on two towers that share the encoder and have a decoder that predicts two probability heatmaps in the target frame that encode the likelihood of the location of the source point. Our network also predicts a depth value for the matched point and a visibility score that measures if the source point is visible in target frame.

classification problems per pixel. Second, we also compute a *softmax-heatmap*:

$$\mathcal{H}_{sm} = \sigma_{sm}(\mathbf{H}(D_{out}))$$

Here we use a softmax activation σ_{sm} to make the complete heatmap a probability distribution, i.e., it sums to one. As ground truth for the heatmap prediction we could take an image with zero values everywhere except at the ground truth pixel position that we set to one. To prevent the trained network from predicting only zero values, we apply a Gaussian kernel $G_{x_{gt}}$ around the ground truth pixel. It sets the ground truth pixel's value to one and decays the neighboring pixel values to zero with standard deviation of 7 pixels, resulting in the ground truth heatmap \mathcal{H}_{gt} . We also add larger weight to pixels close to the ground truth pixel, defining the pixel weight as $w_{\mathcal{H}}(x) = 1 + 10 \cdot G_{x_{gt}}(x)$. The heatmap loss is then computed as:

$$\mathcal{L}_{\mathcal{H}} = \sum_i \Phi_{bce}(w_{\mathcal{H}}(\mathcal{H}_{sg} - \mathcal{H}_{gt})) + \lambda_{nll} \sum_i \Phi_{nll}(w_{\mathcal{H}}(\mathcal{H}_{sm} - \mathcal{H}_{gt}))$$

Here, $\Phi_{bce}(\bullet)$ denotes the binary cross entropy loss and $\Phi_{nll}(\bullet)$ the negative log-likelihood loss, and we empirically determined a weight $\lambda_{nll} = 10$. From these two probability heatmaps, a single one is computed as $\mathcal{H} = \mathcal{H}_{sg} \otimes \mathcal{H}_{sm}$, where \otimes is the Hadamard product.

Depth. In addition to heatmap prediction, our network also predicts the matched point's depth value in the target camera's coordinate system. Inspired by [105], we predict the depth densely, predicting the same depth value for every pixel in the output

image:

$$\mathcal{D} = \exp(\mathbf{D}(D_{\text{out}}))$$

Here, \mathbf{D} is a convolutional layer converting feature space into depth values, and the exponential is applied to guarantee positive depth predictions. Ground truth for depth prediction \mathcal{D}_{gt} is the depth of the ground truth match, repeated for the whole image. Since we want to encourage depth prediction to focus on the matched pixel, we again use pixel weighting, this time in the form of $w_{\mathcal{D}}(x) = G_{x_{\text{gt}}}(x)$, setting the center pixel’s weight to 1 and decaying the weights to 0. Using the weighted version of mean squared error $\Phi_{mse}(\bullet)$ we employ the following loss for depth prediction:

$$\mathcal{L}_{\mathcal{D}} = \lambda_{\text{d}} \sum_i \Phi_{mse}(w_{\mathcal{D}}(\mathcal{D} - \mathcal{D}_{\text{gt}}))$$

Visibility. Furthermore, we also predict a visibility score $\in [0, 1]$ that measures whether the source point is visible (high value) or occluded (low value) in the target frame:

$$\mathcal{V} = \sigma_{\text{sg}}(\mathbf{V}(B_{\text{out}}))$$

Here, B_{out} is the output feature map of the bottleneck layer, \mathbf{V} is a convolutional layer, and σ_{sg} a sigmoid activation. The visibility loss takes the following form:

$$\mathcal{L}_{\mathcal{V}} = \sum_i \Phi_{bce}(\mathcal{V} - \mathcal{V}_{\text{gt}})$$

In the end, we train the network using a weighted combination of all presented loss functions:

$$\mathcal{L} = \mathcal{L}_{\mathcal{H}} + \lambda_{\mathcal{D}} \mathcal{L}_{\mathcal{D}} + \lambda_{\mathcal{V}} \mathcal{L}_{\mathcal{V}}$$

In all experiments we use the constant and empirically determined weights $\lambda_{\mathcal{D}} = 100$ and $\lambda_{\mathcal{V}} = 1$. An overview of the network architecture is given in Fig. 3.3. We implemented our non-rigid matching approach in PyTorch [22] and we train it using stochastic gradient descent with batch size of 32, momentum of 0.9 and learning rate of 0.01. For regularization, we use a weight decay of 0.0005. The learning rate is divided by 10 every 30k iteration steps. We first train the network for heatmap and depth prediction for 100k iterations. Afterwards, we train only the visibility detection layers for another 100k iterations, keeping the weights in the encoder and bottleneck layers fixed. Different on-the-fly data augmentation techniques are applied, such as random 2D rotation, translation, and horizontal flip.

3.4 Non-Rigid Reconstruction Pipeline

We integrate the learned non-rigid matching algorithm into a non-rigid RGB-D reconstruction framework that efficiently tracks dense, space-time coherent, non-rigid deformations on the GPU and also provides an efficient volumetric fusion backend. A canon-

ical model of the scene is reconstructed from data, in parallel to tracking the non-rigid deformations, and stored based on a truncated signed distance field (TSDF) represented by a uniform voxel grid. New observations are fused into the grid based on an exponentially moving average. The non-rigid scene motion is tracked based on the following tracking energy:

$$E_{\text{total}}(\mathcal{T}) = E_{\text{data}}(\mathcal{T}) + \lambda_{\text{learned}} E_{\text{learned}}(\mathcal{T}) + \lambda_{\text{reg}} E_{\text{reg}}(\mathcal{T})$$

The weights $\lambda_{\text{learned}} = 1$ and $\lambda_{\text{reg}} = 1$ are empirically determined and balance the different terms.

3.4.1 Deformation Model

To parameterize scene motion, similar to [24], we employ a coarse deformation graph \mathcal{G} with K deformation nodes $\mathbf{v}_i \in \mathbb{R}^3$. The graph models the deformation of space based on the convex combination of local per-node transformations that are parameterized by rotation parameters $\boldsymbol{\omega}_i \in \mathbb{R}^3$ in Lie algebra space and a translation vector $\mathbf{t}_i \in \mathbb{R}^3$. In total, this leads to about 2k free variables to describe scene motion that we jointly refer to as \mathcal{T} . This enables us to decouple the number of free variables from the complexity of the reconstructed scene. The deformation nodes are connected based on proximity, for details we refer to the original embedded deformation paper [24].

3.4.2 Optimization Terms

For data term $E_{\text{data}}(\mathcal{T})$, similar to [4], [35], we employ dense point-to-point and point-to-plane alignment constraints between the input depth map and the current reconstruction. For regularizer E_{reg} , we employ the as-rigid-as-possible (ARAP) constraint [25] to enforce locally rigid motion. In addition, we integrate a sparse feature alignment term based on our learned correspondences (see Sec. 3.3). For each node \mathbf{v}_i of the current deformation graph, we predict a probability heatmap \mathcal{H}_i that gives the likelihood of its 2D uv-position in the current input depth map, using the initial depth map as the reference frame. Furthermore, we also back-project the pixel with the maximum heatmap response into a 3D point $\mathbf{p}_i \in \mathbb{R}^3$, using its depth. We aim to align the graph nodes with the maximum in the corresponding heatmap using the following alignment constraint:

$$E_{\text{learned}}(\mathcal{T}) = \sum_{v_i \in \mathcal{G}} (1 - \mathcal{H}_i(\pi(\mathbf{v}_i + \mathbf{t}_i)))^2 + \lambda_{\text{point}} \sum_{v_i \in \mathcal{G}} (\mathbf{v}_i + \mathbf{t}_i - \mathbf{p}_i)^2$$

Here, $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the projection from 3D camera space to 2D screen space. The heatmap \mathcal{H}_i is normalized to a maximum of 1. We empirically set $\lambda_{\text{point}} = 10$. In order to handle outliers, especially in the case of occluded correspondences, we make use of the predicted visibility score and the predicted depth value of the match. We filter out all heatmap correspondences with visibility score < 0.5 . We compare the predicted depth with the queried depth from the target frame’s depth map at the pixel with the

maximum heatmap response and invalidate any correspondence with a depth difference > 0.15 meters.

3.4.3 Energy Optimization

We efficiently tackle the underlying optimization problem using a data-parallel Gauss-Newton solver to find the deformation graph \mathcal{G}^* that best explains the data:

$$\mathcal{G}^* = \arg \min E_{\text{total}}(\mathcal{G}) .$$

In the Gauss-Newton solver, we solve the underlying sequence of linear problems using data-parallel preconditioned conjugate gradient (PCG).

3.5 Semi-supervised Data Acquisition

In the following, we provide the details of our semi-supervised non-rigid data collection process that is used for training the non-rigid matching and for the evaluation of non-rigid reconstruction algorithms. The high-level overview of the data acquisition pipeline is shown in Fig. 3.1.

3.5.1 Data Acquisition

In order to obtain RGB-D scans of non-rigidly moving objects, we use a Structure Sensor mounted on an iPad. The depth stream is recorded at a resolution of 640×480 and 30 frames per second; the RGB stream is captured with the iPad camera at a resolution of 1296×968 pixels that is calibrated with respect to the range sensor. Regarding the scanning instructions, we follow the ScanNet [17] pipeline. However, in our case, we focus on scenes with one up to several non-rigidly moving objects in addition to a static background. In total, we recorded 400 scenes with over 390,000 RGB-D frames.

3.5.2 Data Annotation

We crowd sourced sparse ground truth correspondence annotations and segmentation masks for our novel data set. To this end, we employed a web-based annotation tool. The annotation was divided into two tasks. Firstly, we select up to 10 frames per sequence. All dynamic objects that are found in these frames are given unique instance ids (the same instance in the whole sequence) and their masks are annotated in each frame. To accelerate mask segmentation, we use a hierarchy of superpixels as candidate brush sizes. Secondly, among the annotated frames up to 10 frame pairs are selected, and the sparse correspondences between all dynamic objects are annotated. Expert annotators were instructed to annotate correspondences uniformly over the complete object, labeling about 20 point matches per frame pair. Furthermore, in parts of the source image that are occluded in the target image occlusion points were uniformly selected to collect data samples for visibility detection. The dynamic object segmentation task takes on average

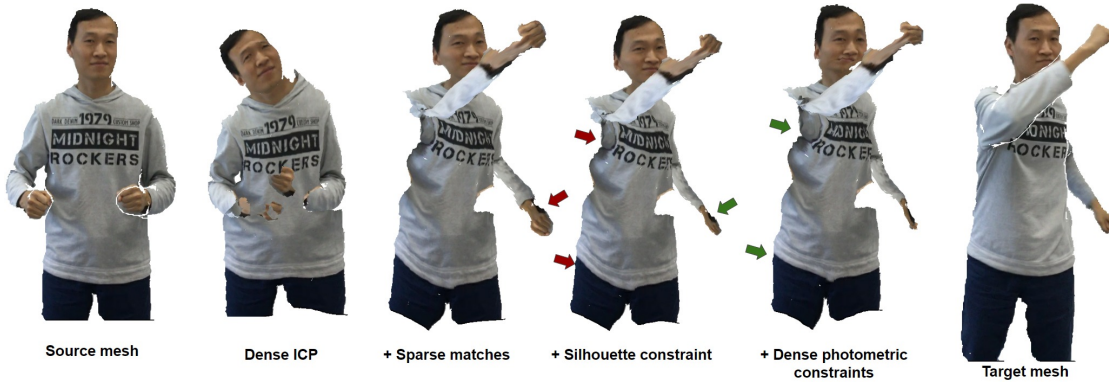


Figure 3.4: Dense alignment constraints. We qualitatively show the effect of different optimization constraints on the dense alignment of the given RGB-D frame pair.

about 1 min per frame, while the correspondence labeling task takes on average about 2 min per frame. In total, our dataset contains 4,479 object masks, 149,228 sparse matches and 63,512 point occlusions.

3.5.3 Dense Data Alignment

Using the annotated object masks and the sparse matches, dense non-rigid alignment of the frame pairs is performed. We follow a similar approach as for non-rigid reconstruction (see Sec. 3.4), based on the sparse deformation graph of [24]. The deformation graph is defined on the source frame’s depth map, covering only the dynamic object by using the source object mask. The final energy function that is optimized to align the source RGB-D frame to the target RGB-D frame is:

$$E_{\text{total}}(\mathcal{T}) = E_{\text{data}}(\mathcal{T}) + \lambda_{\text{photo}}E_{\text{photo}}(\mathcal{T}) + \lambda_{\text{silh}}E_{\text{silh}}(\mathcal{T}) + \lambda_{\text{sparse}}E_{\text{sparse}}(\mathcal{T}) + \lambda_{\text{reg}}E_{\text{reg}}(\mathcal{T})$$

Here, $E_{\text{photo}}(\mathcal{T})$ encourages the color gradient values from the source frame to match the target frame, E_{silh} penalizes deformation of the object outside of the target frame’s object mask, and E_{sparse} enforces annotated sparse matches to be satisfied. The weights $\lambda_{\text{photo}} = 0.001$, $\lambda_{\text{silh}} = 0.0001$, $\lambda_{\text{sparse}} = 100.0$ and $\lambda_{\text{reg}} = 10.0$ have been empirically determined. Details about the different optimization terms and a qualitative comparison of their effects can be found Fig. 3.4. In order to cope with simple apparent topology changes that are very common while capturing natural non-rigid motion, such as the hand touching the body in one frame and moving away in another frame, we execute non-rigid alignment in both directions and compute the final non-rigid alignment using forward-backward motion interpolation, similar to [106]. At the end, a quick manual review step is performed, in which, if necessary, any incorrectly aligned mesh parts are removed. The review step takes about 30 seconds per frame. Examples of dense alignment results and the employed review interface can be found in the accompanying video. Overall, our dataset includes 5,533 densely aligned frame pairs.

Method	2D-err	3D-err	2D-acc	3D-acc
SIFT [29]	138.40	0.552	16.20	14.08
SURF [31]	125.72	0.476	22.13	19.82
SHOT [93]	105.34	0.342	13.43	11.51
FPFH [30]	109.49	0.393	10.85	9.43
3Dmatch [32]	68.98	0.273	30.50	25.33
GPC [107]	65.04	0.231	31.93	28.16
FlowNet-2.0 [108]	27.32	0.118	68.68	63.67
Ours-12.5%	78.82	0.268	27.17	23.28
Ours-25.0%	58.28	0.197	40.32	35.81
Ours-50.0%	45.43	0.156	50.70	46.57
Ours-Rigid	57.87	0.270	40.93	35.64
Ours-SelfSupervised	33.34	0.121	60.87	55.70
Ours-Sparse	31.42	0.106	58.53	52.24
Ours-NoWeighting	23.72	0.083	73.13	68.46
Ours	19.56	0.073	77.60	72.48

Table 3.1: Quantitative matching comparison. We outperform baseline matching methods by a considerable margin. 2D/3D errors are average pixel/point errors, and 2D/3D accuracy is the percentage of pixels/points with distance of at most 20 px/0.05 m.

3.6 Experiments

We provide a train-val-test split with 340 sequences in the training set, 30 in the test set, and 30 in the validation set. We made sure that there is no overlap of captured environments between training and validation/test scenes.

3.6.1 Non-Rigid Matching Evaluation

For a given set of pixels (and corresponding 3D points) in the source image, the task is to find the corresponding pixel (and 3D point) in the target image, as visualized in Fig. 3.5. We evaluate the average 2D pixel and 3D point error (in meters), and compute the matching accuracy (ratio of matches closer than 20 pixels or 0.05 meters from the ground truth correspondences). We compare our non-rigid matching approach to several hand-crafted feature matching strategies, that are based on depth or color based descriptors, and to the learned 3Dmatch [32] descriptor, see Tab. 3.1. Specifically, we compare to the hand-crafted geometry descriptors, such as Unique Signatures of Histograms (SHOT) [93] and the Fast Point Feature Histograms (FPFH) [30]. We also compare to color-based descriptors, e.g., SIFT [29] and SURF [31], that can be used to establish a sparse set of matches between RGB-D frames. Finally, we train a learned descriptor from [32], patch-based random forest matcher from [107] and optical flow prediction network from [108] on our training sequences. Our method consistently outperforms all the baselines.

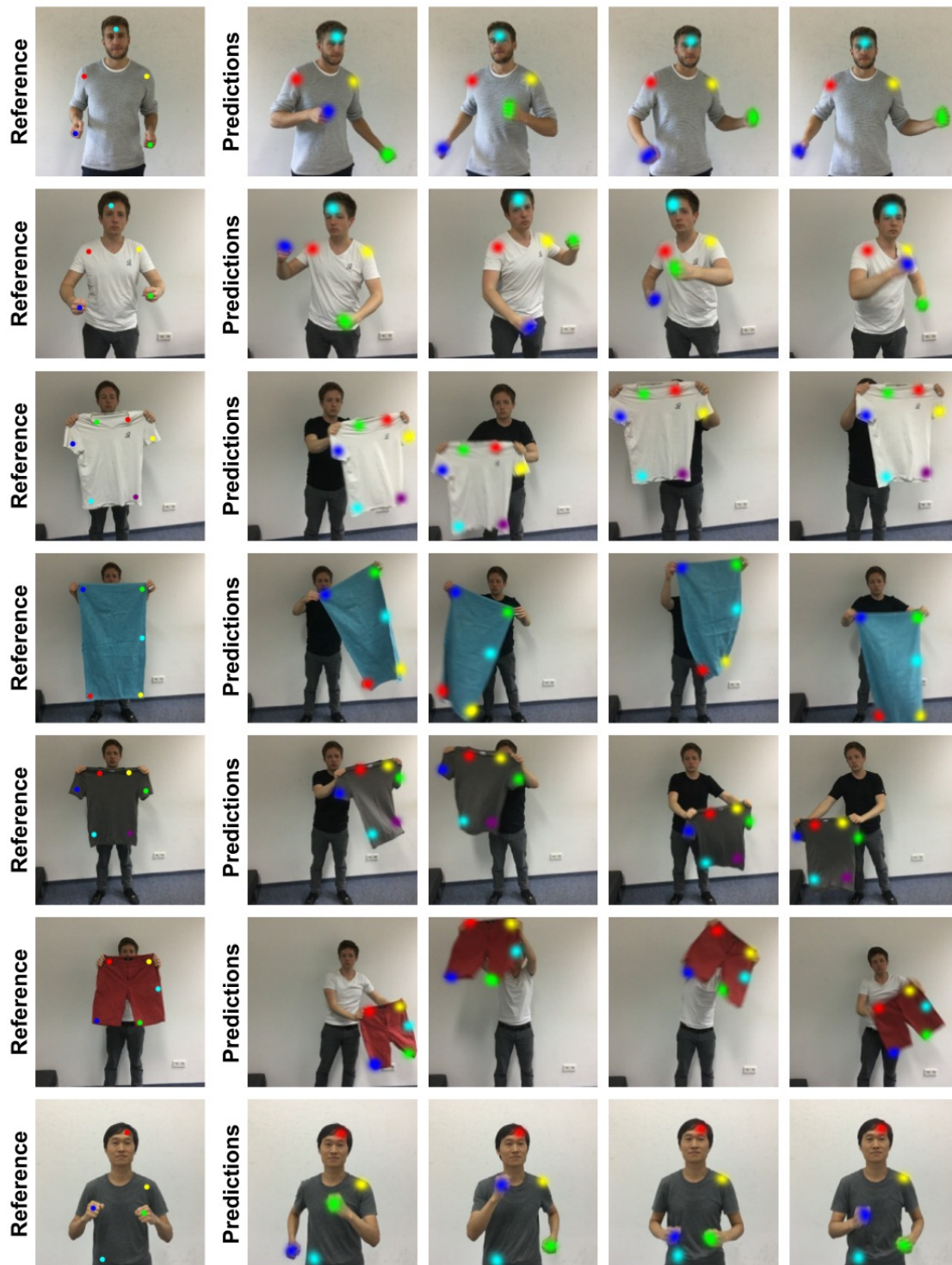


Figure 3.5: Qualitative heatmap prediction results. Our matching approach works well even for highly non-rigid motions.

Method	Def. error (cm)	Geo. error (cm)
DynamicFusion re-impl. [4]	6.31	1.08
VolumeDeform [35]	21.27	7.78
DynamicFusion [4] + 3Dmatch [32]	6.64	1.59
Ours-Rigid	12.21	2.30
Ours-Sparse	8.24	0.77
Ours-SelfSupervised	5.47	0.54
Ours-Base	3.94	0.43
Ours-Occlusion	3.70	0.42
Ours-Occlusion+Depth	3.28	0.41

Table 3.2: Quantitative reconstruction comparison. Our learned correspondences significantly improve both tracking and reconstruction quality compared to the state-of-the-art approaches. We also provide ablation studies on training data type and different network parts.

3.6.2 Non-Rigid Reconstruction Results

We integrated our learned matching strategy into a non-rigid reconstruction pipeline. Our learned data term significantly improves reconstruction quality, both qualitatively and quantitatively. To be able to perform a quantitative comparison on our test sequences, we used our re-implementation of [4], and the code or results provided from the authors of [35], [62]–[64]. We also replaced our data-driven correspondence matching module with the descriptor learning network from [32], trained on our data, and used it in combination with 3D Harris keypoints. The quantitative evaluation is shown in Tab. 3.2. The evaluation metrics measure deformation error (a 3D distance between the annotated and computed correspondence positions) and geometry error (comparing depth values inside the object mask to the reconstructed geometry). Deformation error is the more important metric, since it also measures tangential drift within the surface. To be able to know which dynamic object to reconstruct if multiple are present, we always provide the initial ground truth segmentation mask of the selected object. All approaches in Tab. 3.2 were evaluated on all 30 test sequences to provide a comparison on different kinds of objects and deformable motions. [64] provided results on two challenging test sequences, their average deformation and geometry error are 21.05 cm and 14.87 cm respectively, while our approach achieves average errors of 3.63 cm and 0.48 cm. Our approach outperforms the state of the art by a large margin. The methods [62] and [63] do not compute explicit point correspondences from the canonical frame to other frames, so we could not evaluate these approaches quantitatively; we provide qualitative comparison on our sequences in Fig. 3.8. We also show qualitative comparisons with our re-implementation of DynamicFusion [4] in Fig. 3.6 and with the state-of-the-art approach of [64] in Fig. 3.7. Our learned correspondences enable us to handle faster object motion as well as challenging planar motion, where even photometric cues fail, for instance due to uniform object color, resulting in high-quality reconstructions in Fig. 3.9.

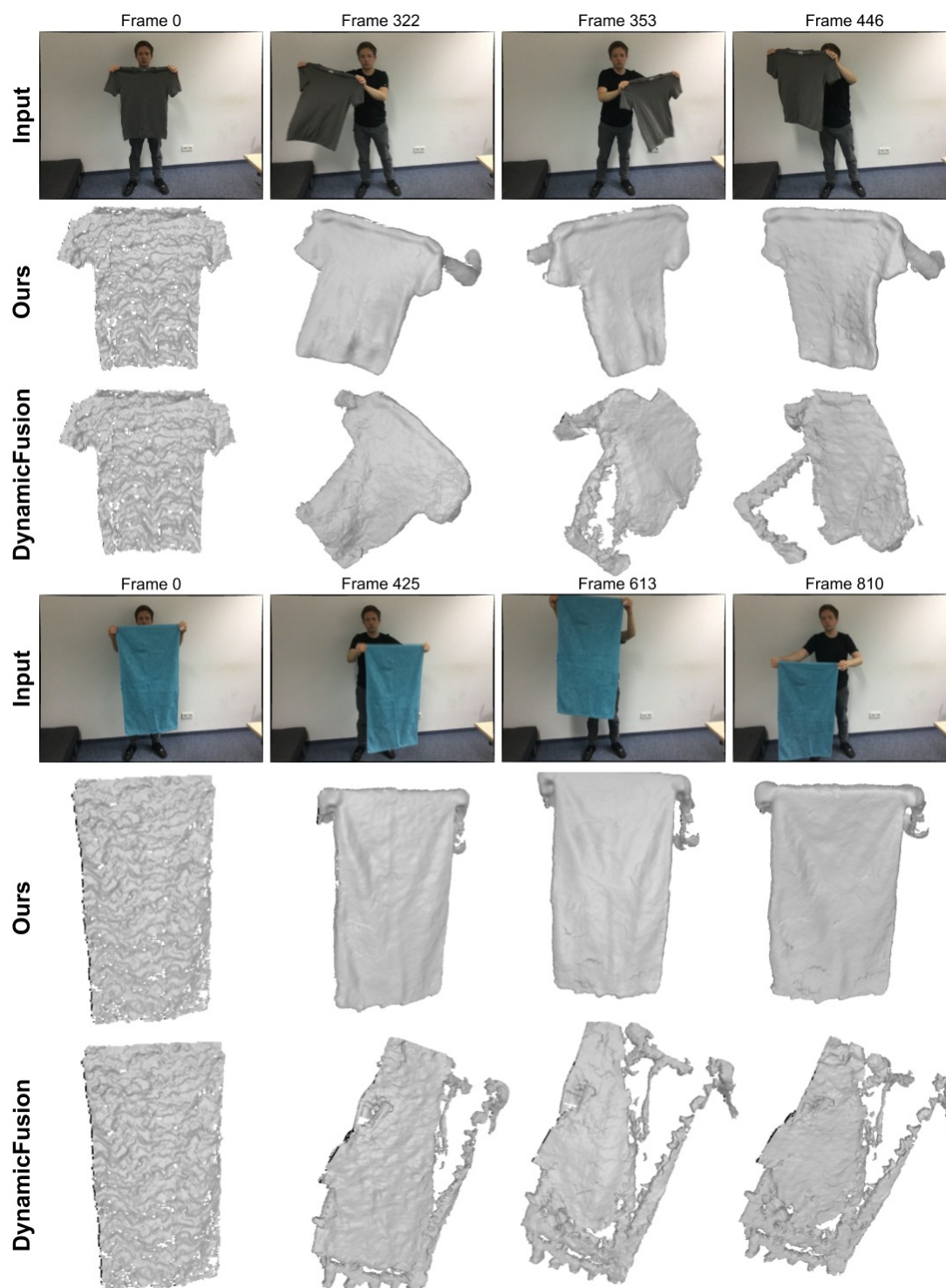


Figure 3.6: Qualitative comparison to DynamicFusion [4]. We use our re-implementation of the framework.

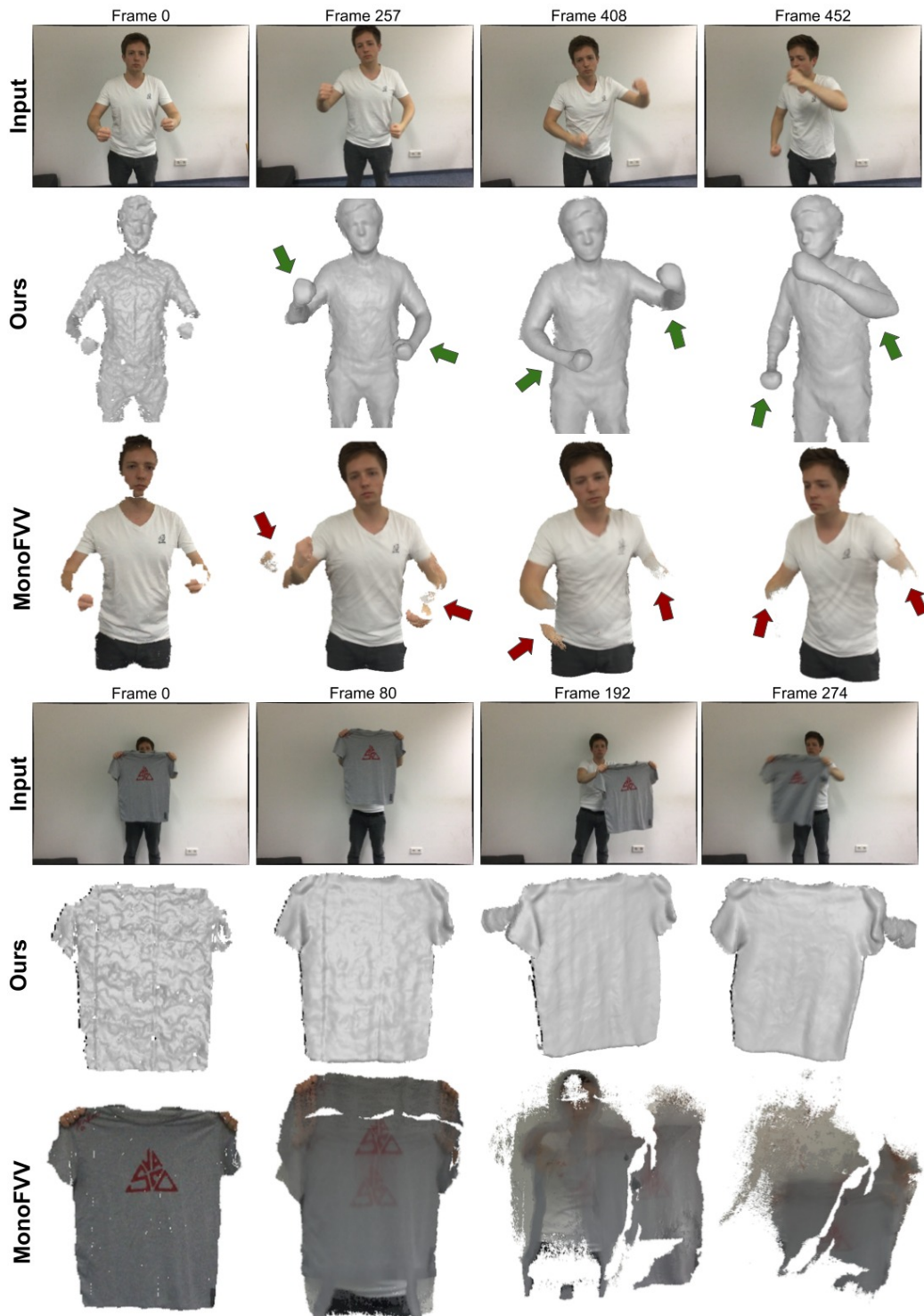


Figure 3.7: Qualitative comparison to MonoFVV [64]. Reconstruction results were kindly provided by the authors.

Part II. Non-rigid Reconstruction using Data-driven Priors

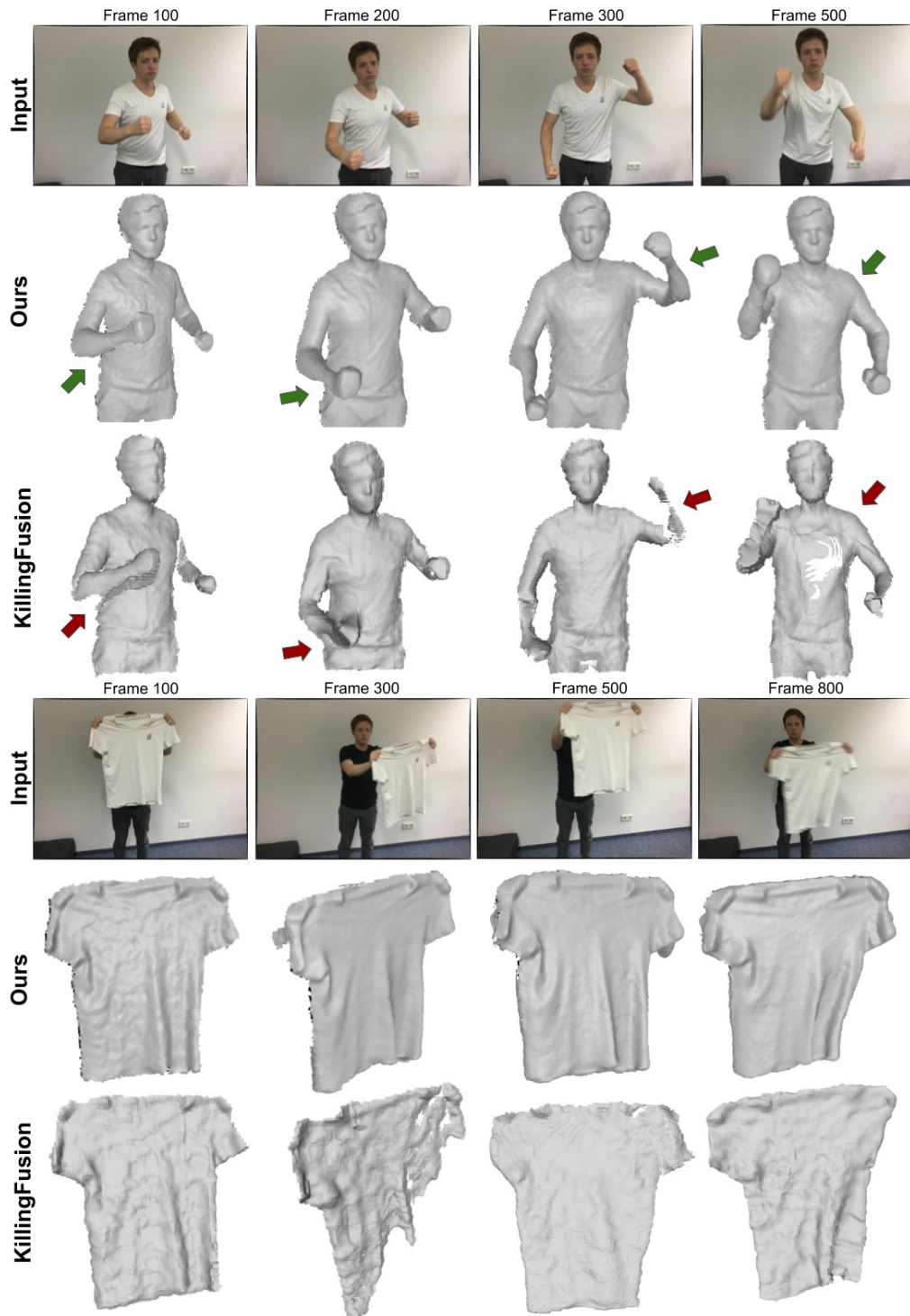


Figure 3.8: Qualitative comparison to KillingFusion [62]. Reconstruction results were kindly provided by the authors.

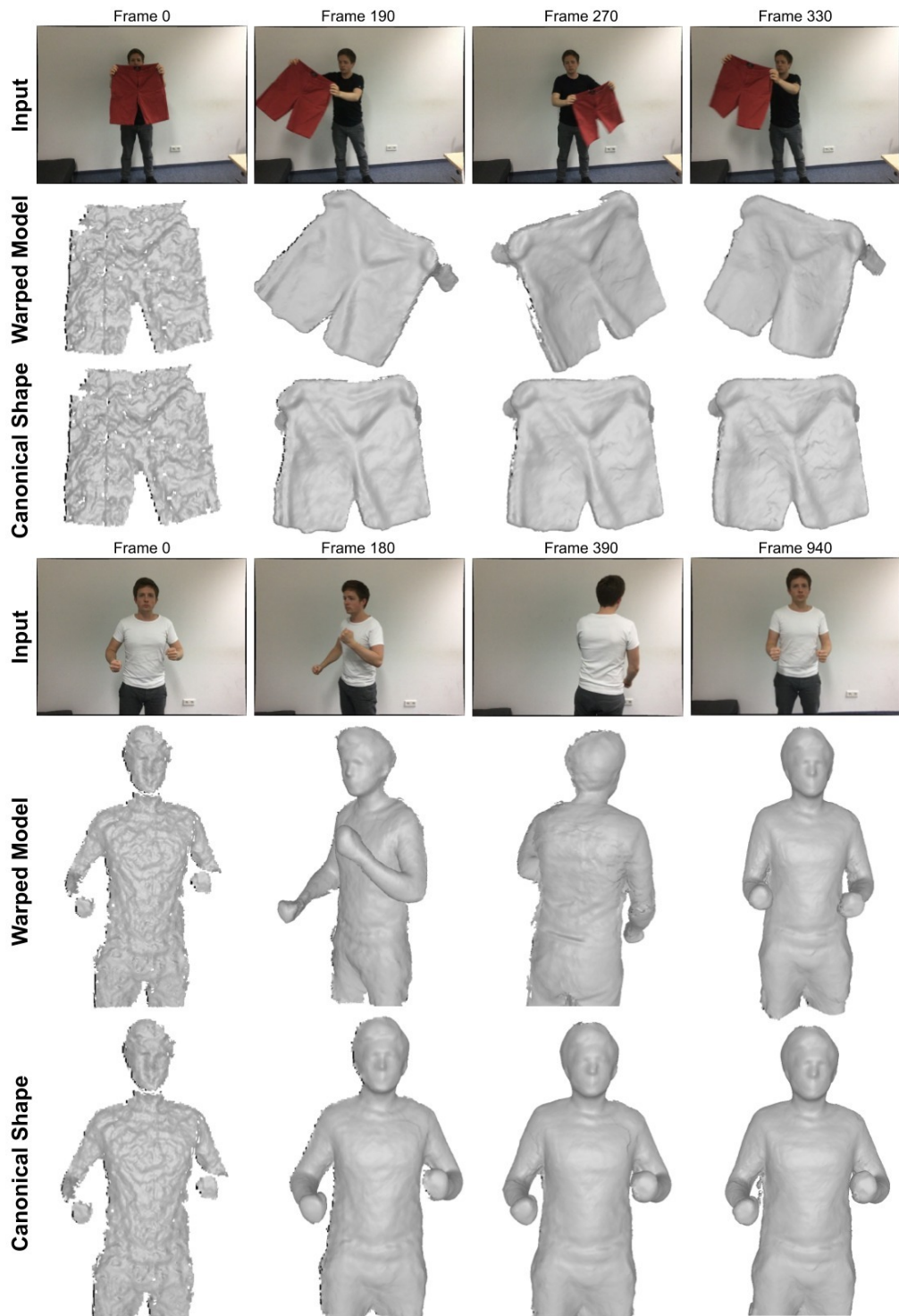


Figure 3.9: Reconstruction visualizations. Our approach obtains high-quality warped model (in the current frame) and canonical shape (in the initial frame).

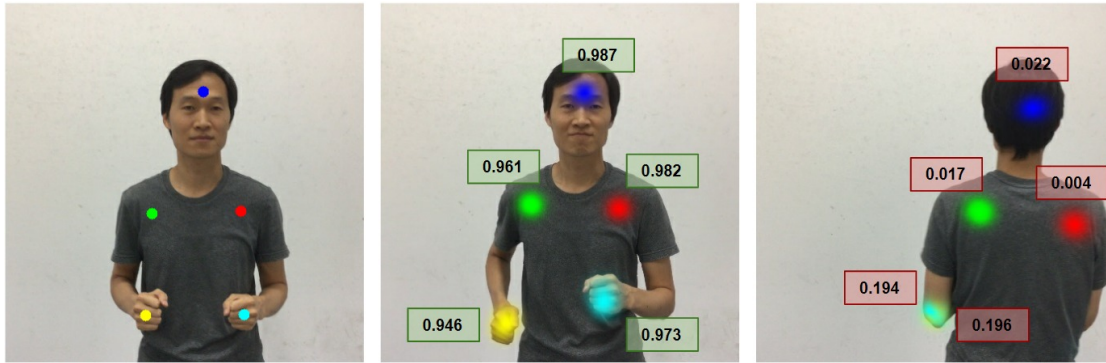


Figure 3.10: Visibility detection. The visibility score is in the range $[0, 1]$, it is high for visible correspondences and low for occluded parts. We filter out all correspondences with visibility score less than 0.50.

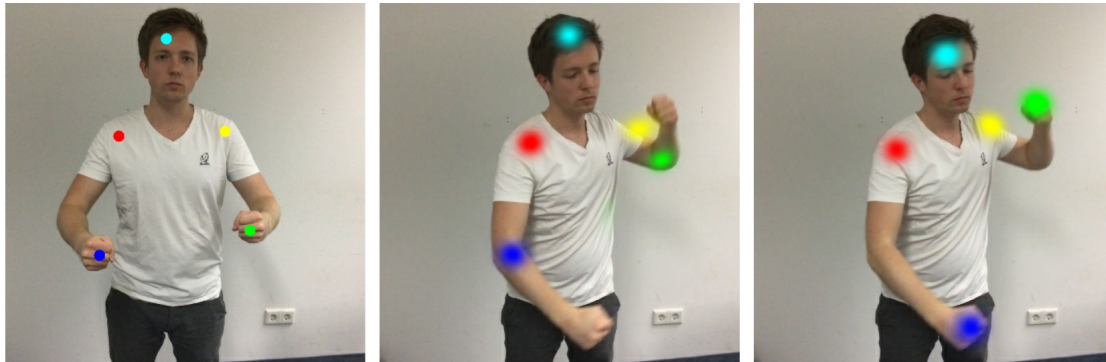


Figure 3.11: Ablation on data generation. Comparison of correspondence prediction for reference frame (left) using self-supervised training data (middle) and semi-supervised dense data (right).

3.6.3 Ablation Study

We evaluated different components of our network and their effect on the reconstruction quality, see Tab. 3.2. Since some sequences include motions in which large parts of the reconstructed object are occluded, as can be observed in Fig. 3.10, using visibility detection for correspondence pruning makes our method more robust. Furthermore, since depth measurements and heatmap predictions can sometimes both be noisy, adding correspondence filtering with depth prediction further improves the reconstruction results.

3.6.4 Data Generation Evaluation

To show the importance of our semi-supervised strategy for constructing the training corpus, we evaluate how different training corpora influence the performance of data-driven reconstruction methods. Aside from our training data, which has been generated using dense semi-supervised frame alignment, we used a publicly available rigid dataset

of indoor scenes (from [17]), self-supervised alignment of our sequences (as in [69]), and only manually annotated sparse samples from our dataset. We provide comparison on both non-rigid matching in Tab. 3.1 and non-rigid reconstruction in Tab. 3.2. Using only rigid data does not generalize to non-rigid sequences. While sparse matches already improve network performance, there is not enough data for reliable correspondence prediction on every part of the observed object. In addition, annotated sparse matches are usually matches on image parts that are easy for humans to match, and there are much less accurate correspondences in areas with uniform color. In the self-supervised setting, the network gets dense correspondence information, which improves the method’s reconstruction performance compared to using only sparse features. However, without semi-supervised densely aligned frame pairs, we can only generate matches for the simple deformations, where the DynamicFusion approach can successfully track the motion. Therefore, the performance of the network trained only on self-supervised data degrades considerably on more extreme deformations, as can be seen in Fig. 3.11. Dense alignment of too far away frames is needed for accurate network prediction in the case of extreme deformations. Since the majority of the densely aligned matches are still moving rigidly, it turned out to be beneficial to sample more deformable samples during training. In order to estimate which parts of the scene are more deformable, we employed sparsely annotated matches and ran RANSAC in combination with a Procrustes algorithm to estimate the average rigid pose. The more the motion of each sampled match differs from the average rigid motion, the more often we sample it during network training using a multinomial distribution of the non-rigid displacement weights. This strategy improved the network performance, as is shown in Tab. 3.1, compared to training on non-weighted samples. Finally, we demonstrate how much data is needed to achieve robust correspondence prediction performance; using less training data considerably degrades matching accuracy, as summarized in Tab. 3.1, where we trained networks using only 12.5%, 25.0%, and 50.0% of the training data.

3.6.5 Limitations

While learned correspondences make tracking of fast motion more robust, there is still room for improvement when reconstructing dynamic objects. One pressing issue is that background clutter might be accidentally fused with the object when the object is close to the background. In this case, the reconstructed shape would slowly grow and we might also start reconstructing the background. This can cause wrong deformation graph connectivity and lead to tracking failures. A potential future avenue is to subtract and ignore the background; e.g., we could use our annotated object masks to develop a data-driven method.

3.7 Conclusion

We have proposed a neural network architecture for matching correspondences in non-rigid sequences that operates on RGB-D frames and demonstrated that our learned

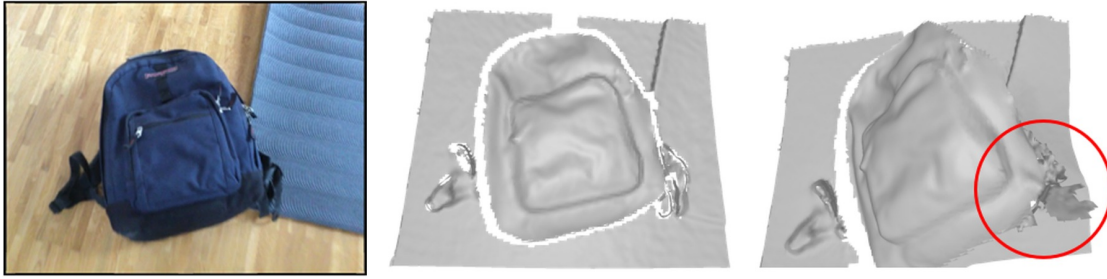


Figure 3.12: Limitations. Integration of background can cause wrong deformation graph connectivity, which can lead to non-rigid tracking failure.

descriptors outperform existing hand-crafted ones. In addition, we introduced the first large-scale dataset that is composed of 400 scenes, over 390,000 RGB-D frames, and 5,533 densely aligned frame pairs. The dataset is obtained with a semi-supervised strategy by combining self-supervision with sparse annotations to obtain dense inter-frame correspondences. We also provide a test set along with several metrics for evaluating non-rigid matching and non-rigid reconstruction. We believe that our dataset is a first step towards enabling learning-based non-rigid matching and our benchmark will help to quantitatively and objectively compare different approaches.

4 Neural Non-rigid Tracking

Abstract of paper. We introduce a novel, end-to-end learnable, differentiable non-rigid tracker that enables state-of-the-art non-rigid reconstruction by a learned robust optimization. Given two input RGB-D frames of a non-rigidly moving object, we employ a convolutional neural network to predict dense correspondences and their confidences. These correspondences are used as constraints in an as-rigid-as-possible (ARAP) optimization problem. By enabling gradient back-propagation through the weighted non-linear least squares solver, we are able to learn correspondences and confidences in an end-to-end manner such that they are optimal for the task of non-rigid tracking. Under this formulation, correspondence confidences can be learned via self-supervision, informing a learned robust optimization, where outliers and wrong correspondences are automatically down-weighted to enable effective tracking. Compared to state-of-the-art approaches, our algorithm shows improved reconstruction performance, while simultaneously achieving $85\times$ faster correspondence prediction than comparable deep-learning based methods.

4.1 Introduction

The capture and reconstruction of real-world environments is a core problem in computer vision, enabling numerous VR/AR applications. While there has been significant progress in reconstructing static scenes, tracking and reconstruction of dynamic objects remains a challenge. Non-rigid reconstruction focuses on dynamic objects, without assuming any explicit shape priors, such as human or face parametric models. Commodity RGB-D sensors, such as Microsoft’s Kinect or Intel’s Realsense, provide a cost-effective way to acquire both color and depth video of dynamic motion. Using a large number of RGB-D sensors can lead to an accurate non-rigid reconstruction, as shown by [66]. Our work focuses on non-rigid reconstruction from a single RGB-D camera, thus eliminating the need for specialized multi-camera setups.

The seminal DynamicFusion by [4] introduced a non-rigid tracking and mapping pipeline that uses depth input for real-time non-rigid reconstruction from a single RGB-D camera. Various approaches have expanded upon this framework by incorporating sparse color correspondences [35] or dense photometric optimization [64]. DeepDeform [13] presented a learned correspondence prediction, enabling significantly more robust tracking of fast motion and re-localization. Unfortunately, the computational cost of the correspondence prediction network (~ 2 seconds per frame for a relatively small number of non-rigid correspondences) inhibits real-time performance.

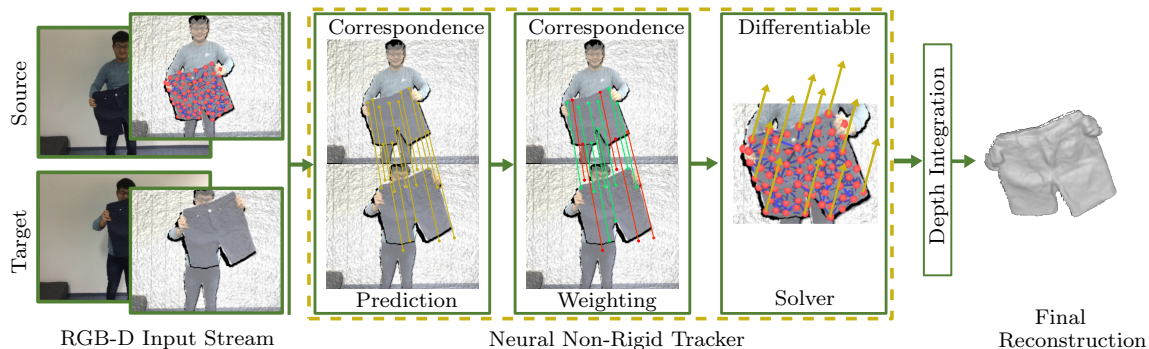


Figure 4.1: Neural Non-Rigid Tracking. Based on RGB-D input data of a source and a target frame, our learned non-rigid tracker estimates the non-rigid deformations to align the source to the target frame. We propose an end-to-end approach, enabling correspondences and their importance weights to be informed by the non-rigid solver. Similar to robust optimization, this provides robust tracking, and the resulting deformation field can then be used to integrate the depth observations in a canonical volumetric 3D grid that implicitly represents the surface of the object (final reconstruction).

Simultaneously, work on learned optical flow has shown dense correspondence prediction at real-time rates [34]. However, directly replacing the non-rigid correspondence predictions from [13] with these optical flow predictions does not produce accurate enough correspondences for comparable non-rigid reconstruction performance. In our work, we propose a neural non-rigid tracker, i.e., an end-to-end differentiable non-rigid tracking pipeline which combines the advantages of classical deformation-graph-based reconstruction pipelines [4], [35] with novel learned components. Our end-to-end approach enables learning outlier rejection in a self-supervised manner, which guides a robust optimization to mitigate the effect of inaccurate correspondences or major occlusions present in single RGB-D camera scenarios.

Specifically, we cast the non-rigid tracking problem as an as-rigid-as-possible (ARAP) optimization problem, defined on correspondences between points in a source and a target frame. A differentiable Gauss-Newton solver allows us to obtain gradients that enable training a neural network to predict an importance weight for every correspondence in a completely self-supervised manner, similar to robust optimization. The end-to-end training significantly improves non-rigid tracking performance. Using our neural tracker in a non-rigid reconstruction framework results in $85\times$ faster correspondence prediction and improved reconstruction performance compared to the state of the art.

In summary, we propose a novel neural non-rigid tracking approach with two key contributions:

- an end-to-end differentiable Gauss-Newton solver, which provides gradients to better inform a correspondence prediction network used for non-rigid tracking of two frames;

- a self-supervised approach for learned correspondence weighting, which is informed by our differentiable solver and enables efficient, robust outlier rejection, thus, improving non-rigid reconstruction performance compared to the state of the art.

4.2 Related Work

Non-rigid Reconstruction. Reconstruction of deformable objects using a single RGB-D camera is an important research area in computer vision. State-of-the-art methods rely on deformation graphs [24], [109] that enable robust and temporally consistent 3D motion estimation. While earlier approaches required an object template, such graph-based tracking has been extended to simultaneous tracking and reconstruction approaches [4], [86]. These works used depth fitting optimization objectives in the form of iterative closest points, or continuous depth fitting in [62], [63]. Rather than relying solely on depth information, recent works have incorporated SIFT features [35], dense photometric fitting [64], and sparse learned correspondence [13].

Correspondence Prediction for Non-rigid Tracking. In non-rigid tracking, correspondences must be established between the two frames we want to align. While methods such as DynamicFusion [4] rely on projective correspondences, recent methods leverage learned correspondences [13]. DeepDeform [13] relies on sparse predicted correspondences, trained on an annotated dataset of deforming objects. Since prediction is done independently for each correspondence, this results in a high compute cost, compared to dense predictions of state-of-the-art optical flow networks. Optical flow [33], [34], [108], [110] and scene flow [111]–[114] methods achieve promising results in predicting dense correspondences between two frames, with some approaches not even requiring direct supervision [115]–[117]. In our proposed neural non-rigid tracking approach, we build upon PWC-Net [34] for dense correspondence prediction to inform our non-rigid deformation energy formulation. Since our approach allows for end-to-end training, our 2D correspondence prediction finds correspondences better suited for non-rigid tracking.

Differentiable Optimization. Differentiable optimizers have been explored for various tasks, including image alignment [118], rigid pose estimation [119], [120], multi-frame direct bundle-adjustment [121], and rigid scan-to-CAD alignment [122]. In addition to achieving higher accuracy, an end-to-end differentiable optimization approach also offers the possibility to optimize run-time, as demonstrated by learning efficient preconditioning methods in [21], [123], [124]. Unlike Li et al. [21], which employs an image-based tracker (with descriptors defined on nodes in a pixel-aligned graph), our approach works on general graphs and learns to robustify correspondence prediction for non-rigid tracking by learning self-supervised correspondence confidences.

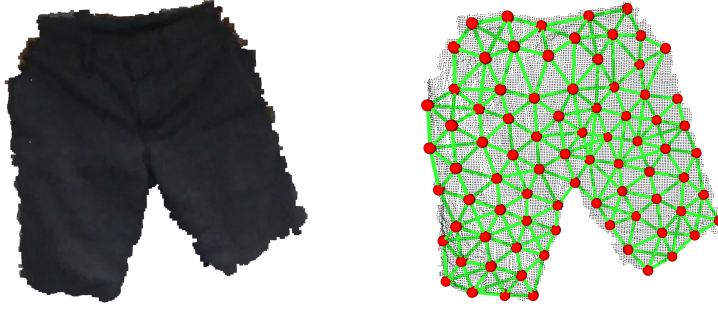


Figure 4.2: Deformation graph construction. Given an object in the source RGB-D frame, we define a deformation graph \mathcal{G} over the former. Nodes \mathcal{V} (red spheres) are uniformly subsampled over the source RGB-D frame. Edges \mathcal{E} (green lines) are computed between nodes based on geodesic connectivity among the latter.

4.3 Non-Rigid Reconstruction Notation

Non-rigid alignment is a crucial part of non-rigid reconstruction pipelines. In the single RGB-D camera setup, we are given a pair of source and target RGB-D frames $\{(\mathcal{I}_s, \mathcal{P}_s), (\mathcal{I}_t, \mathcal{P}_t)\}$, where $\mathcal{I}_* \in \mathbb{R}^{H \times W \times 3}$ is an RGB image and $\mathcal{P}_* \in \mathbb{R}^{H \times W \times 3}$ a 3D point image. The goal is to estimate a warp field $Q : \mathbb{R}^3 \mapsto \mathbb{R}^3$ that transforms \mathcal{P}_s into the target frame. Note that we define the 3D point image \mathcal{P}_s as the result of back-projecting every pixel $\mathbf{u} \in \Pi_s \subset \mathbb{R}^2$ into the camera coordinate system with given camera intrinsic parameters. To this end, we define the inverse of the perspective projection to back-project a pixel \mathbf{u} given the pixel's depth $d_{\mathbf{u}}$ and the intrinsic camera parameters \mathbf{c} :

$$\pi_{\mathbf{c}}^{-1} : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^3, \quad (\mathbf{u}, d_{\mathbf{u}}) \mapsto \pi_{\mathbf{c}}^{-1}(\mathbf{u}, d_{\mathbf{u}}) = \mathbf{p} \quad (4.1)$$

To maintain robustness against noise in the depth maps, state-of-the-art approaches define an embedded deformation graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ over the *source* RGB-D frame, where \mathcal{V} is the set of graph nodes defined by their 3D coordinates $\mathbf{v}_i \in \mathbb{R}^3$ and \mathcal{E} the set of edges between nodes, as described in [24] and illustrated in Fig. 4.2. Thus, for every node in \mathcal{G} , a global translation vector $\mathbf{t}_{\mathbf{v}_i} \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R}_{\mathbf{v}_i} \in \mathbb{R}^{3 \times 3}$, must be estimated in the alignment process. We parameterize rotations with a 3-dimensional axis-angle vector $\boldsymbol{\omega} \in \mathbb{R}^3$. We use the exponential map $\exp : \mathfrak{so}(3) \rightarrow \text{SO}(3)$, $\hat{\boldsymbol{\omega}} \mapsto e^{\hat{\boldsymbol{\omega}}} = \mathbf{R}$ to convert from axis-angle to matrix rotation form, where the $\hat{\cdot}$ -operator creates a 3×3 skew-symmetric matrix from a 3-dimensional vector. The resulting graph motion is denoted by $\mathcal{T} = (\boldsymbol{\omega}_{\mathbf{v}_1}, \mathbf{t}_{\mathbf{v}_1}, \dots, \boldsymbol{\omega}_{\mathbf{v}_N}, \mathbf{t}_{\mathbf{v}_N}) \in \mathbb{R}^{N \times 6}$ for a graph with N nodes.

Dense motion can then be computed by interpolating the nodes' motion \mathcal{T} by means of a warping function Q . When applied to a 3D point $\mathbf{p} \in \mathbb{R}^3$, it produces the point's deformed position

$$Q(\mathbf{p}, \mathcal{T}) = \sum_{\mathbf{v}_i \in \mathcal{V}} \alpha_{\mathbf{v}_i} (e^{\hat{\boldsymbol{\omega}}_{\mathbf{v}_i}} (\mathbf{p} - \mathbf{v}_i) + \mathbf{v}_i + \mathbf{t}_{\mathbf{v}_i}) \quad (4.2)$$

The weights $\alpha_{\mathbf{v}_i} \in \mathbb{R}$, also known as *skinning* weights, measure the influence of each node on the current point \mathbf{p} and are computed as defined in Eq. 2.2.

Axis-angle optimization. The axis-angle rotation representation has singularities for larger angles, i.e., two different vectors $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$ can represent the same rotation (for example keeping the same axis and increasing the angle by 2π results in identical rotation). To avoid singularities, we decompose the rotation matrix into $e^{\hat{\boldsymbol{\omega}}_{\mathbf{v}_i}} = e^{\hat{\boldsymbol{\epsilon}}_{\mathbf{v}_i}} \mathbf{R}_{\mathbf{v}_i}$ with $\boldsymbol{\epsilon}_{\mathbf{v}_i} = 0$, therefore optimizing only delta rotations that have rather small rotation angles.

4.4 Neural Non-rigid Tracking

Given a pair of source and target RGB-D frames $(\mathcal{Z}_s, \mathcal{Z}_t)$, where $\mathcal{Z}_* = (\mathcal{I}_* | \mathcal{P}_*) \in \mathbb{R}^{H \times W \times 6}$ is the concatenation of an RGB and a 3D point image as defined in Section 4.3, we aim to find a function Θ that estimates the motion \mathcal{T} of a deformation graph \mathcal{G} with N nodes (given by their 3D coordinates \mathcal{V}) defined over the source RGB-D frame. This implicitly defines source-to-target dense 3D motion (see Figure 4.3). Formally, we have:

$$\Theta : \mathbb{R}^{H \times W \times 6} \times \mathbb{R}^{H \times W \times 6} \times \mathbb{R}^{N \times 3} \rightarrow \mathbb{R}^{N \times 6}, \quad (\mathcal{Z}_s, \mathcal{Z}_t, \mathcal{V}) \mapsto \Theta(\mathcal{Z}_s, \mathcal{Z}_t, \mathcal{V}) = \mathcal{T} \quad (4.3)$$

To estimate \mathcal{T} , we first establish dense 2D correspondences between the source and target frame using a deep neural network Φ . These correspondences, denoted as \mathcal{C} , are used to construct the data term in our non-rigid alignment optimization. Since the presence of outlier correspondence predictions has a strong negative impact on the performance of non-rigid tracking, we introduce a weighting function Ψ , inspired by robust optimization, to down-weight inaccurate predictions. Function Ψ outputs importance weights \mathcal{W} and is learned in a self-supervised manner. Finally, both correspondence predictions \mathcal{C} and importance weights \mathcal{W} are input to a differentiable, non-rigid alignment optimization module Ω . By optimizing the non-rigid alignment energy (see Section 4.4.3), the differentiable optimizer Ω estimates the deformation graph parameters \mathcal{T} that define the motion from source to target frame:

$$\mathcal{T} = \Theta(\mathcal{Z}_s, \mathcal{Z}_t, \mathcal{V}) = \Omega(\Phi(\cdot), \Psi(\cdot), \mathcal{V}) = \Omega(\mathcal{C}, \mathcal{W}, \mathcal{V}) \quad (4.4)$$

In the following, we define the dense correspondence predictor Φ , the importance weighting Ψ and the optimizer Ω , and describe a fully differentiable approach for optimizing Φ and Ψ such that we can estimate dense correspondences with importance weights best suited for non-rigid tracking.

4.4.1 Dense Correspondence Prediction

The dense correspondence prediction function Φ takes as input a pair of source and target RGB images $(\mathcal{I}_s, \mathcal{I}_t)$, and for each source pixel location $\mathbf{u} \in \Pi_s \subset \mathbb{R}^2$ it outputs a corresponding pixel location in the target image \mathcal{I}_t , which we denote by $\mathbf{c}_{\mathbf{u}} \in \Pi_t \subset \mathbb{R}^2$.

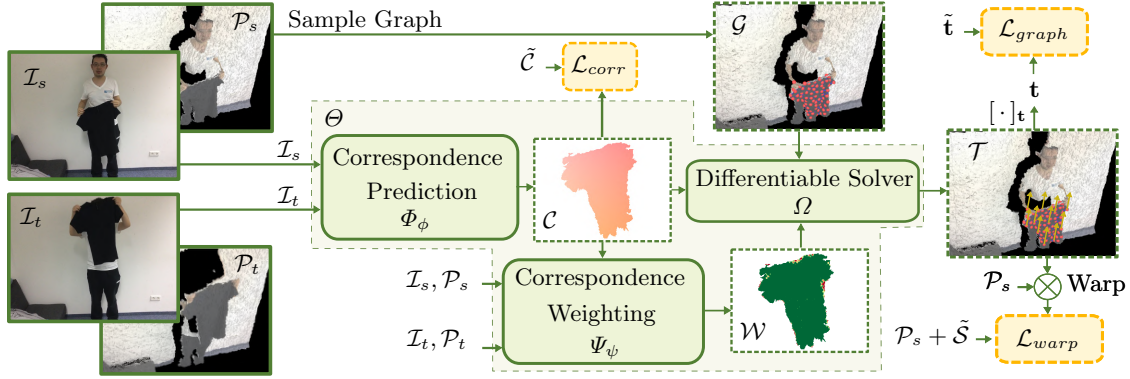


Figure 4.3: Overview of neural non-rigid tracker. Given a pair of source and target images, \mathcal{I}_s and \mathcal{I}_t , a dense correspondence map \mathcal{C} between the frames is estimated via a convolutional neural network Φ . Importance weights \mathcal{W} for these correspondences are computed through a function Ψ . Together with a graph \mathcal{G} defined over the source RGB-D frame \mathcal{P}_s , both \mathcal{C} and \mathcal{W} are input to a differentiable solver Ω . The solver outputs the graph motion \mathcal{T} , i.e., the non-rigid alignment between source and target frames. Our approach is optimized end-to-end, with losses on the final alignment using $\mathcal{L}_{\text{graph}}$ and $\mathcal{L}_{\text{warp}}$, and an intermediate loss on the correspondence map $\mathcal{L}_{\text{corr}}$.

Formally, Φ is defined as

$$\Phi : \mathbb{R}^{H \times W \times 3} \times \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{H \times W \times 2}, \quad (\mathcal{I}_s, \mathcal{I}_t) \mapsto \Phi(\mathcal{I}_s, \mathcal{I}_t) = \mathcal{C} \quad (4.5)$$

where \mathcal{C} is the resulting dense correspondence map. The function Φ is represented by a deep neural network that leverages the architecture of a state-of-the-art optical flow estimator [34].

Coarse-to-fine optimization. The design of our correspondence prediction function Φ follows the PWC-Net [34] architecture that predicts the correspondences in a coarse-to-fine manner. Initially the correspondences are predicted at a coarse resolution of 10×7 px, and then refined to a resolution of 20×14 px, etc. In total, there are $L = 5$ levels in the correspondence hierarchy, and the finest level predictions are used in the differentiable non-rigid optimization. At every level l , we bilinearly downsample the groundtruth correspondences to a coarser resolution, resulting in $\tilde{\mathcal{C}}^l$. Similarly, the ground-truth mask is downsampled to a coarser version $\tilde{M}^{\mathcal{C}^l}$, to avoid propagating gradients through invalid pixels. For each training sample $(\mathcal{I}_s, \mathcal{I}_t)$ and every level l , we therefore compute ground-truth correspondences $\tilde{\mathcal{C}}^l$ and the ground-truth mask $\tilde{M}^{\mathcal{C}^l}$.

4.4.2 Correspondence Importance Weights

For each source pixel $\mathbf{u} \in \Pi_s \subset \mathbb{R}^2$ and its correspondence $\mathbf{c}_u \in \Pi_t \subset \mathbb{R}^2$, we additionally predict an importance weight $w_u \in (0, 1)$ by means of the weighting function Ψ . The

latter takes as input the source RGB-D image \mathcal{Z}_s , the corresponding sampled target frame values \mathcal{Z}'_t , and intermediate features from the correspondence network Φ , and outputs weights for the correspondences between source and target. Note that \mathcal{Z}'_t is the result of bilinearly sampling [125] the target image \mathcal{Z}_t at the predicted correspondence locations \mathcal{C} . The last layer of features \mathcal{H} of the correspondence network Φ , with dimension $D = 565$, are used to inform Ψ . The weighting function is thus defined as

$$\Psi : \mathbb{R}^{H \times W \times 6} \times \mathbb{R}^{H \times W \times 6} \times \mathbb{R}^{H \times W \times D} \rightarrow \mathbb{R}^{H \times W \times 1}, \quad (\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H}) \mapsto \Psi(\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H}) = \mathcal{W} \quad (4.6)$$

4.4.3 Differentiable Optimizer

We introduce a differentiable optimizer Ω to estimate the deformation graph parameters \mathcal{T} , given the correspondence map \mathcal{C} , importance weights \mathcal{W} , and N graph nodes \mathcal{V} :

$$\Omega : \mathbb{R}^{H \times W \times 2} \times \mathbb{R}^{H \times W \times 1} \times \mathbb{R}^{N \times 3} \rightarrow \mathbb{R}^{N \times 6}, \quad (\mathcal{C}, \mathcal{W}, \mathcal{V}) \mapsto \Omega(\mathcal{C}, \mathcal{W}, \mathcal{V}) = \mathcal{T} \quad (4.7)$$

with \mathcal{C} and \mathcal{W} estimated by functions Φ (Eq. 4.5) and Ψ (Eq. 4.6), respectively. Using the predicted dense correspondence map \mathcal{C} , we establish the data term for the non-rigid tracking optimization. Specifically, we use a 2D data term that operates in image space and a depth data term that leverages the depth information of the input frames. In addition to the data terms, we employ an As-Rigid-As-Possible regularizer [25] to encourage node deformations to be locally rigid, enabling robust deformation estimates even in the presence of noisy input cues. Note that the resulting optimizer module Ω is fully differentiable, but contains no learnable parameters. In summary, we formulate non-rigid tracking as the following nonlinear optimization problem:

$$\arg \min_{\mathcal{T}} (\lambda_{2D} E_{2D}(\mathcal{T}) + \lambda_{\text{depth}} E_{\text{depth}}(\mathcal{T}) + \lambda_{\text{reg}} E_{\text{reg}}(\mathcal{T})) \quad (4.8)$$

2D reprojection term. Given the outputs of the dense correspondence predictor and weighting function, $\Phi(\mathcal{I}_s, \mathcal{I}_t)$ and $\Psi(\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H})$, respectively, we query for every pixel \mathbf{u} in the source frame its correspondence $\mathbf{c}_{\mathbf{u}}$ and weight $w_{\mathbf{u}}$ to build the following energy term:

$$E_{2D}(\mathcal{T}) = \sum_{\mathbf{u} \in \mathcal{I}_s} w_{\mathbf{u}}^2 \|\pi_{\mathbf{c}}(\mathbf{Q}(\mathbf{p}_{\mathbf{u}}, \mathcal{T})) - \mathbf{c}_{\mathbf{u}}\|_2^2 \quad (4.9)$$

where $\pi_{\mathbf{c}} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$, $\mathbf{p} \mapsto \pi_{\mathbf{c}}(\mathbf{p})$ is a perspective projection with intrinsic parameters \mathbf{c} and $\mathbf{p}_{\mathbf{u}} = \pi_{\mathbf{c}}^{-1}(\mathbf{u}, d_{\mathbf{u}})$ as defined in Eq. 4.1. Each pixel is back-projected to 3D, deformed using the current graph motion estimate as described in Eq. 4.2 and projected onto the target image plane. The projected deformed location is compared to the predicted correspondence $\mathbf{c}_{\mathbf{u}}$.

Depth term. The depth term leverages the depth cues of the source and target images. Specifically, it compares the z components of a warped source point, i.e., $[\mathbf{Q}(\mathbf{p}_{\mathbf{u}}, \mathcal{T})]_z$,

and a target point sampled at the corresponding location \mathbf{c}_u using bilinear interpolation:

$$E_{\text{depth}}(\mathcal{T}) = \sum_{\mathbf{u} \in \Pi_s} w_{\mathbf{u}}^2 ([Q(\mathbf{p}_u, \mathcal{T})]_z - [P_t(\mathbf{c}_u)]_z)^2 \quad (4.10)$$

Regularization term. We encourage the deformation of neighboring nodes in the deformation graph to be locally rigid. Each node $\mathbf{v}_i \in \mathcal{V}$ has at most $K = 8$ neighbors in the set of edges \mathcal{E} , computed as nearest nodes using geodesic distances. The regularization term follows [25]:

$$E_{\text{reg}}(\mathcal{T}) = \sum_{(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E}} \left\| e^{\hat{\omega}_{\mathbf{v}_i}} (\mathbf{v}_j - \mathbf{v}_i) + \mathbf{v}_i + \mathbf{t}_{\mathbf{v}_i} - (\mathbf{v}_j + \mathbf{t}_{\mathbf{v}_j}) \right\|_2^2 \quad (4.11)$$

Equation 4.8 is minimized using the Gauss-Newton algorithm, as described in Algorithm 2. In the following, we denote the number of correspondences by $|\mathcal{C}|$ and the number of graph edges by $|\mathcal{E}|$. Moreover, we transform all energy terms into a residual vector $\mathbf{r} \in \mathbb{R}^{3|\mathcal{C}|+3|\mathcal{E}|}$. For every graph node, we compute partial derivatives with respect to translation and rotation parameters, constructing a Jacobian matrix $\mathbf{J} \in \mathbb{R}^{(3|\mathcal{C}|+3|\mathcal{E}|) \times 6N}$, where N is the number of nodes in the set of vertices \mathcal{V} . Analytic formulas for partial derivatives are described in App. D.

Initially, the deformation parameters are initialized to $\mathcal{T}_0 = \mathbf{0}$, corresponding to zero translation and identity rotations. In each iteration n , the residual vector \mathbf{r}_n and the Jacobian matrix \mathbf{J}_n are computed using the current estimate \mathcal{T}_n , and the following linear system is solved (using LU decomposition) to compute an increment $\Delta\mathcal{T}$:

$$\mathbf{J}_n^T \mathbf{J}_n \Delta\mathcal{T} = -\mathbf{J}_n^T \mathbf{r}_n \quad (4.12)$$

At the end of every iteration, the motion estimate \mathcal{T} is updated as $\mathcal{T}_{n+1} = \mathcal{T}_n + \Delta\mathcal{T}$. Most operations are matrix-matrix or matrix-vector multiplications, which are trivially differentiable. Derivatives of the linear system solve operation are computed analytically, as described in [23] and detailed in App. D. We use *max_iter* = 3 Gauss-Newton iterations, which encourages the correspondence prediction and weight functions, Φ and Ψ , respectively, to make predictions such that convergence in 3 iterations is possible. In our experiments we use $(\lambda_{2D}, \lambda_{\text{depth}}, \lambda_{\text{reg}}) = (0.001, 1, 1)$.

4.4.4 End-to-end Optimization

Given a dataset of samples $\mathcal{X}_{s,t} = \{[\mathcal{I}_s | \mathcal{P}_s], [\mathcal{I}_t | \mathcal{P}_t], \mathcal{V}\}$, our goal is to find the parameters ϕ and ψ of Φ_ϕ and Ψ_ψ , respectively, so as to estimate the motion \mathcal{T} of a deformation graph \mathcal{G} defined over the source RGB-D frame. This can be formulated as a differentiable optimization problem (allowing for back-propagation) with the following objective:

$$\arg \min_{\phi, \psi} \sum_{\mathcal{X}_{s,t}} \lambda_{\text{corr}} \mathcal{L}_{\text{corr}}(\phi) + \lambda_{\text{graph}} \mathcal{L}_{\text{graph}}(\phi, \psi) + \lambda_{\text{warp}} \mathcal{L}_{\text{warp}}(\phi, \psi) \quad (4.13)$$

Algorithm 2 Gauss-Newton Optimization

```

1:  $\mathcal{C} \leftarrow \Phi(\mathcal{I}_s, \mathcal{I}_t)$  ▷ Estimate correspondences
2:  $\mathcal{W} \leftarrow \Psi(\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H})$  ▷ Estimate importance weights
3: function SOLVER( $\mathcal{C}, \mathcal{W}, \mathcal{V}$ )
4:    $\mathcal{T} \leftarrow \mathbf{0}$ 
5:   for  $n \leftarrow 0$  to max_iter do
6:      $\mathbf{J}, \mathbf{r} \leftarrow \text{ComputeJacobianAndResidual}(\mathcal{V}, \mathcal{T}, \mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{C}, \mathcal{W})$ 
7:      $\Delta\mathcal{T} \leftarrow \text{LUdecomposition}(\mathbf{J}^T \mathbf{J} \Delta\mathcal{T} = -\mathbf{J}^T \mathbf{r})$  ▷ Solve linear system
8:      $\mathcal{T} \leftarrow \mathcal{T} + \Delta\mathcal{T}$  ▷ Apply increment
9:   return  $\mathcal{T}$ 
    
```

Correspondence loss. We use a robust q -norm as in [34] to enforce closeness of correspondence predictions to ground-truth. At every hierarchy level l the correspondence loss has the following form:

$$\mathcal{L}_{\text{corr}}^l(\phi) = \tilde{M}^{\mathcal{C}^l} (|\Phi_\phi^l(\mathcal{I}_s, \mathcal{I}_t) - \tilde{\mathcal{C}}^l| + \epsilon)^q \quad (4.14)$$

Operator $|\cdot|$ denotes the ℓ_1 norm, $q < 1$ (we set it to $q = 0.4$) and ϵ is a small constant. Ground-truth correspondences at specific level l are denoted by $\tilde{\mathcal{C}}^l$. Since valid ground truth for all pixels is not available, we employ a ground-truth mask $\tilde{M}^{\mathcal{C}^l}$ to avoid propagating gradients through invalid pixels.

Graph loss. We impose an l_2 -loss on node translations \mathbf{t} (ground-truth rotations are not available):

$$\mathcal{L}_{\text{graph}}(\phi, \psi) = \tilde{M}^{\mathcal{V}} \left\| \underbrace{[\Omega(\Phi_\phi(\mathcal{I}_s, \mathcal{I}_t), \Psi_\psi(\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H}), \mathcal{V}))]_{\mathbf{t}}}_{\mathcal{T}} - \tilde{\mathbf{t}} \right\|_2^2 \quad (4.15)$$

where $[\cdot]_{\mathbf{t}} : \mathbb{R}^{N \times 6} \rightarrow \mathbb{R}^{N \times 3}$, $\mathcal{T} \mapsto [\mathcal{T}]_{\mathbf{t}} = \mathbf{t}$ extracts the translation part from the graph motion \mathcal{T} . Node translation ground-truth is denoted by $\tilde{\mathbf{t}}$ and $\tilde{M}^{\mathcal{V}}$ masks out invalid nodes. Mask $\tilde{M}^{\mathcal{V}}$ is needed for numerically stable optimization, as described in more detail below.

Warp loss. We have found that it is beneficial to use the estimated graph deformation \mathcal{T} to deform the dense source point cloud \mathcal{P}_s and enforce the result to be close to the source point cloud when deformed with the ground-truth scene flow $\tilde{\mathcal{S}}$:

$$\mathcal{L}_{\text{warp}}(\phi, \psi) = \tilde{M}^{\mathcal{S}} \left\| \mathbf{Q} \left(\mathcal{P}_s, \underbrace{\Omega(\Phi_\phi(\mathcal{I}_s, \mathcal{I}_t), \Psi_\psi(\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H}), \mathcal{V}))}_{\mathcal{T}} \right) - (\mathcal{P}_s + \tilde{\mathcal{S}}) \right\|_2^2 \quad (4.16)$$

Here, we extend the warping operation \mathbf{Q} (Eq. 4.2) to operate on the dense point cloud \mathcal{P}_s element-wise, and define $\tilde{M}^{\mathcal{S}}$ to mask out invalid points. Note that since we sample graph



Figure 4.4: Appearance reconstruction. We compute shape textures by aggregating color images over 100 frames of motion into a voxel grid.

nodes on depth maps, the graph loss is in practice a subset of the warp loss. However, we found it to be a more general notation to disentangle them (e.g., for scenarios where graph nodes are not sampled on the RGB-D frame).

Numerically stable optimization. The non-rigid tracking optimization objective includes data (correspondence and depth) terms and a regularization (ARAP) term. If we only use regularization term, the optimization problem becomes ill-posed, since any rigid transformation of all graph nodes has no effect on the regularization term. The edge set \mathcal{E} of the deformation graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is computed by connecting each graph node with $K = 8$ nearest nodes, using geodesic distances on the depth map mesh as a metric. This can lead to multiple disconnected graph components, i.e., different node clusters. To ensure the optimization problem is well-defined, we ensure that we have a minimum number of correspondences in each node cluster. In order to satisfy memory limits, we do not use all pixel correspondences \mathcal{C} at training time, but instead randomly sample 10k correspondences. In our experiments we filter out all node clusters with less than 2000 correspondences. This filtering has to be reflected in the loss computation. Thus, we define the mask matrix $\tilde{M}^{\mathcal{V}}$ to have zeros for nodes from filtered clusters, which prevents gradient back-propagation through invalidated graph nodes.

4.4.5 Neural Non-rigid Tracking for 3D Reconstruction

We introduce our differentiable tracking module into the non-rigid reconstruction framework of [4]. In addition to the dense depth ICP correspondences employed in the original method, which help towards local deformation refinement, we employ a keyframe-based tracking objective. Without loss of generality, every 50th frame of the sequence is chosen as a keyframe, to which we establish dense correspondences including the respective weights. We apply a conservative filtering of the predicted correspondences based on the predicted correspondence weights using a fixed threshold $\delta = 0.35$ and re-weight the correspondences based on *cycle consistency*, as detailed below. Using the correspondence predictions and correspondence weights of valid keyframes ($> 50\%$ valid correspondences), the non-rigid tracking optimization problem is solved. The resulting deformation field is used to integrate the depth frame into the canonical volume of the object. We refer to the original reconstruction paper [4] for details regarding the fusion process. On top of geometry, we can also recover shape appearance, as visualized in Fig. 4.4.

Cycle-consistency filtering. We apply correspondence reweighting based on cycle consistencies. Specifically, we enforce bi-directional consistency and multi-keyframe consistency. *Bi-directional consistency* enables us to detect self-occlusions between a keyframe and current frame. Correspondences are predicted in both directions keyframe-to-frame and frame-to-keyframe. If following the correspondence in forward keyframe-to-frame and afterwards in backward frame-to-keyframe direction results in a 3D error larger than 0.20 m, we reject the correspondence. For *multi-keyframe consistency*, multiple keyframe-to-frame predictions are estimated that correspond to the same 3D point in the canonical volume and the mean prediction value is computed. If any of the predictions is more than 0.15 m away from the mean value, we reject all correspondences for a given 3D canonical point.

4.5 Experiments

In the following, we evaluate our method quantitatively and qualitatively on both non-rigid tracking and non-rigid reconstruction. To this end, we use the DeepDeform dataset [13] for training, with the given 340-30-30 train-val-test split of RGB-D sequences. Both non-rigid tracking and reconstruction are evaluated on the hidden test set of the DeepDeform benchmark.

4.5.1 Training Scheme

The non-rigid tracking module has been implemented using the PyTorch library [22] and trained using stochastic gradient descent with momentum 0.9 and learning rate 10^{-5} . We use an Intel Xeon 6240 Processor and an Nvidia RTX 2080Ti GPU. The parameters of the dense correspondence prediction network ϕ are initialized with a PWC-Net model pre-trained on FlyingChairs [33] and FlyingThings3D [126]. We use a 10-factor learning

Model	EPE 3D (mm)	Graph Error 3D (mm)
Φ_c	44.05	67.25
Φ_{c+g}	39.12	57.34
Φ_{c+g+w}	36.96	54.24
$\Phi_c + \Psi_{\text{supervised}}$	28.95	36.77
$\Phi_{c+g+w} + \Psi_{\text{supervised}}$	27.42	34.68
$\Phi_{c+g+w} + \Psi_{\text{self-supervised}}$	26.29	31.00

Table 4.1: Quantitative non-rigid tracking evaluation. We evaluate non-rigid tracking on the DeepDeform dataset [13], showing the benefit of end-to-end differentiable optimizer losses and self-supervised correspondence weighting. We denote correspondence prediction as Φ_c , Φ_{c+g} and Φ_{c+g+w} , depending on which losses $\mathcal{L}_{\text{corr}}$, $\mathcal{L}_{\text{graph}}$, $\mathcal{L}_{\text{warp}}$ are used, and correspondence weighting as $\Psi_{\text{supervised}}$ and $\Psi_{\text{self-supervised}}$, either using an additional supervised loss or not.

rate decay every 10k iterations, requiring about 30k iterations in total for convergence, with a batch size of 4. For optimal performance, we first optimize the correspondence predictor Φ_ϕ with $(\lambda_{\text{corr}}, \lambda_{\text{graph}}, \lambda_{\text{warp}}) = (5, 5, 5)$, without the weighting function Ψ_ψ . Afterwards, we optimize the weighting function parameters ψ with $(\lambda_{\text{corr}}, \lambda_{\text{graph}}, \lambda_{\text{warp}}) = (0, 1000, 1000)$, while keeping ϕ fixed. Finally, we fine-tune both ϕ and ψ together, with $(\lambda_{\text{corr}}, \lambda_{\text{graph}}, \lambda_{\text{warp}}) = (5, 5, 5)$.

Supervised Weight Network Baseline. As a baseline, we introduce a model where we supervise the optimization of the weighting function Ψ_ψ . The ground-truth correspondence weighting $\tilde{W} \in \mathbb{R}^{H \times W \times 1}$ for this supervision is generated by comparing current correspondence predictions \mathcal{C} against the ground-truth correspondences $\tilde{\mathcal{C}}$. We compare 3D distances between correspondences, using the target depth map \mathcal{D}_t to query corresponding depth values. A pixel in the ground-truth weighting \tilde{W} is assigned a 1 or 0 depending on the correspondence error. Optimal performance was achieved by assigning 1 to correspondences that are at most 0.1 m away from groundtruth, and 0 to correspondences that are at least 0.3 m away from groundtruth, without propagating any gradient through remaining correspondence weights. Binary cross-entropy loss is used to optimize Ψ in this supervised setting.

4.5.2 Non-rigid Tracking Evaluation

For any frame pair $\mathcal{X}_{s,t}$ in the DeepDeform dataset [13], we define a deformation graph \mathcal{G} by uniformly sampling graph nodes \mathcal{V} over the source object in the RGB-D frame, given a segmentation mask of the former. Graph node connectivity \mathcal{E} is computed using geodesic distances on a triangular mesh defined over the source depth map. As a pre-processing step, we filter out any frame pairs where more than 30% of the source object is occluded in the target frame. In Table 4.1 non-rigid tracking performance is evaluated by the mean translation error over node translations \mathbf{t} (Graph Error 3D), where the latter

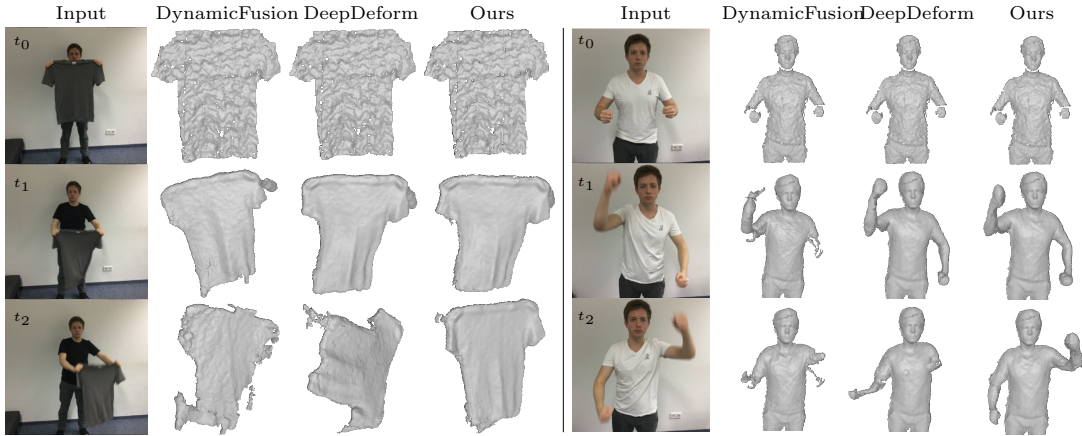


Figure 4.5: Qualitative comparison with DynamicFusion [4] and DeepDeform [13]. We compare reconstruction results on test sequences from [13]. The rows show different time steps of the sequence.

are compared to ground-truth with an l_2 metric. In addition, we evaluate the dense end-point-error (EPE 3D) between the source point cloud deformed with the estimated graph motion, $Q(\mathcal{P}_s, \mathcal{T})$, and the source point cloud deformed with the ground-truth scene flow, $\mathcal{P}_s + \hat{\mathcal{S}}$. To support reproducibility, we report the mean error metrics of multiple experiments, running every setting 3 times.

We show that using graph and warp losses, $\mathcal{L}_{\text{graph}}$ and $\mathcal{L}_{\text{warp}}$, and differentiating through the non-rigid optimizer considerably improves both EPE 3D and Graph Error 3D compared to only using the correspondence loss $\mathcal{L}_{\text{corr}}$. Adding self-supervised correspondence weighting further decreases the errors by a large margin. Supervised outlier rejection with binary cross-entropy loss does bring an improvement compared to models that do not optimize for the weighting function Ψ_ψ . However, optimizing Ψ_ψ in a *self-supervised* manner clearly outperforms the former supervised setup. This is due to the fact that, in the self-supervised scenario, gradients that flow from $\mathcal{L}_{\text{graph}}$ and $\mathcal{L}_{\text{warp}}$ through the differentiable solver Ω can better inform the optimization of Ψ_ψ by minimizing the end-to-end alignment losses. We further experimented with different design choices for our solver.

ARAP edge re-weighting. In non-rigid tracking, it is possible to weight ARAP terms for every graph edge differently, depending on the distance between the nodes. In our method, we sample nodes uniformly on the surface, thus, all edges have similar length (7.13 ± 1.38 cm). Hence, edge re-weighting changes EPE 3D only marginally: 0.8% lower EPE 3D and 1.7% lower Graph Error 3D.

Nearest-neighbor vs. bilinear depth sampling. When querying depth after predicting 2D correspondences, we found bilinear sampling to perform better, with 5.8% lower EPE 3D and 6.29% lower Graph Error 3D compared to nearest-neighbor sampling.

Method	Deformation error (mm)	Geometry error (mm)
DynamicFusion [4]	61.79	10.78
VolumeDeform [35]	208.41	74.85
DeepDeform [13]	31.52	4.16
Ours (Φ_c)	54.85	5.92
Ours (Φ_{c+g+w})	53.27	5.84
Ours ($\Phi_c + \Psi_{\text{supervised}}$)	40.21	5.39
Ours ($\Phi_{c+g+w} + \Psi_{\text{self-supervised}}$)	28.72	4.03

Table 4.2: Quantitative non-rigid reconstruction evaluation. Our method achieves state-of-the-art non-rigid reconstruction results on the DeepDeform benchmark [13]. Both our end-to-end differentiable optimizer and the self-supervised correspondence weighting are necessary for optimal performance. Not only does our approach achieve lower deformation and geometry error compared to state of the art, our correspondence prediction is about $85\times$ faster.

Influence of graph density. We sample graph nodes with 5 cm node coverage, which fits well with our setup of 11 GiB for training. Using coarser graphs, with 10 cm and 15 cm node coverage resulted in poorer performance: 5.49% and 8.33% higher EPE 3D, as well as 7.34% and 28.46% higher Graph Error 3D, respectively. In turn, the memory footprint on the GPU during training (with batch size 4) decreases with node coverage: 10 513 MiB, 6153 MiB and 5931 MiB for 5 cm, 10 cm and 15 cm node coverage, respectively.

Number of optimization steps. We empirically found 3 solver iterations to be the best compromise between performance and computational cost. Unrolling 3 solver iterations instead of 1 / 2, results in 24.9% / 0.16% lower EPE 3D and 22.7% / 0.23% lower Graph Error 3D. Using 4 iterations only improves slightly with respect to 3 (0.04% lower EPE 3D, 0.03% lower Graph Error 3D). More than 4 does not change performance notably.

4.5.3 Non-rigid Reconstruction Evaluation

We evaluate the performance of our non-rigid reconstruction approach on the DeepDeform benchmark [13] (see Table 4.2). The evaluation metrics measure *deformation error*, a 3D end-point-error between tracked and annotated correspondences, and *geometry error*, which compares reconstructed shapes with ground truth depth maps inside annotated foreground object masks. Our approach performs about 8.9% better than the state-of-the-art non-rigid reconstruction approach of [13] on the deformation metric. While our approach consistently shows better performance on both metrics, we also significantly lower the per-frame runtime to 27 ms per keyframe, in contrast to [13], which requires 2299 ms. Thus, our approach can also be used with multiple keyframes at interactive frames rates, e.g., 90 ms for 5 keyframes and 199 ms for 10 keyframes.

To show the influence of the different learned components of our method, we perform an ablation study by disabling either of our two main components: the end-to-end differentiable optimizer or the self-supervised correspondence weighting. As can be seen, our end-to-end trained method with self-supervised correspondence weighting demonstrates the best performance. Qualitatively, we show this in Fig. 4.5. In contrast to DynamicFusion [4] and DeepDeform [13], our method is notably more robust in fast motion scenarios. Fig. 4.6 shows a comparison to [64]. As can be seen, our method better handles non-rigid movements with fast motions (t-shirt) and occlusions (arm). In Fig. 4.7, we compare our approach to the reconstruction results of [62]. The reconstruction of our method leads to more complete and smooth meshes. Additionally, we show qualitative reconstruction results of our method on VolumeDeform [35] sequences in Fig. 4.8. Our method can robustly reconstruct these RGB-D sequences, despite the fact that a Kinect sensor was used to record them, whereas our training data was obtained using a Structure IO sensor. This shows that our network predictions can generalize to a different structured-light sensor input.

4.6 Conclusion

We propose Neural Non-Rigid Tracking, a differentiable non-rigid tracking approach that allows learning the correspondence prediction and weighting of traditional tracking pipelines in an end-to-end manner. The differentiable formulation of the entire tracking pipeline enables back-propagation to the learnable components, guided by a loss on the tracking performance. This not only achieves notably improved tracking error in comparison to state-of-the-art tracking approaches, but also leads to better reconstructions, when integrated into a reconstruction framework like DynamicFusion [4]. We hope that this work inspires further research in the direction of neural non-rigid tracking and believe that it is a stepping stone towards fully differentiable non-rigid reconstruction.

Part II. Non-rigid Reconstruction using Data-driven Priors

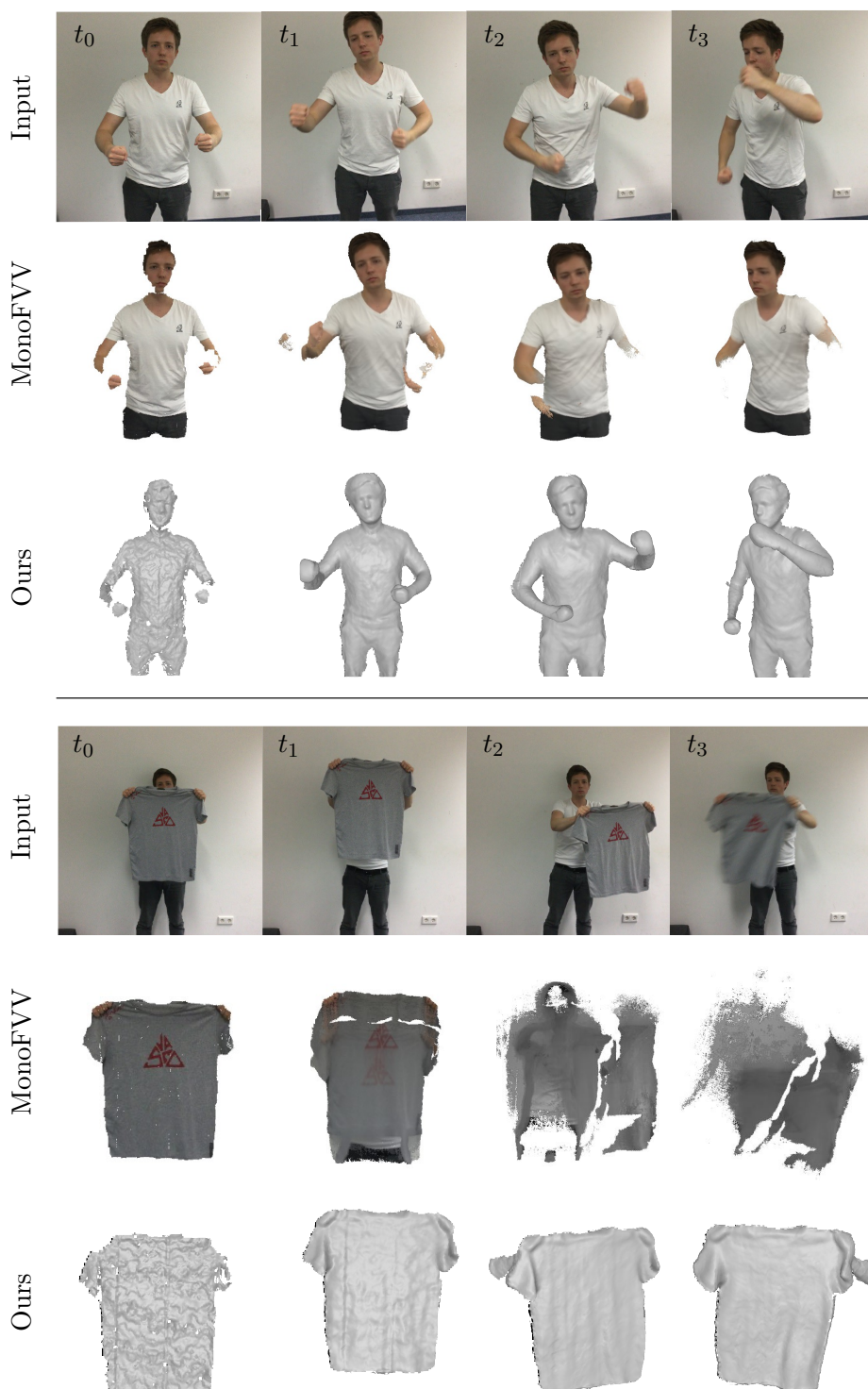


Figure 4.6: Qualitative comparison with MonoFVV [64]. The results on test sequences from [13] were kindly provided by the authors.

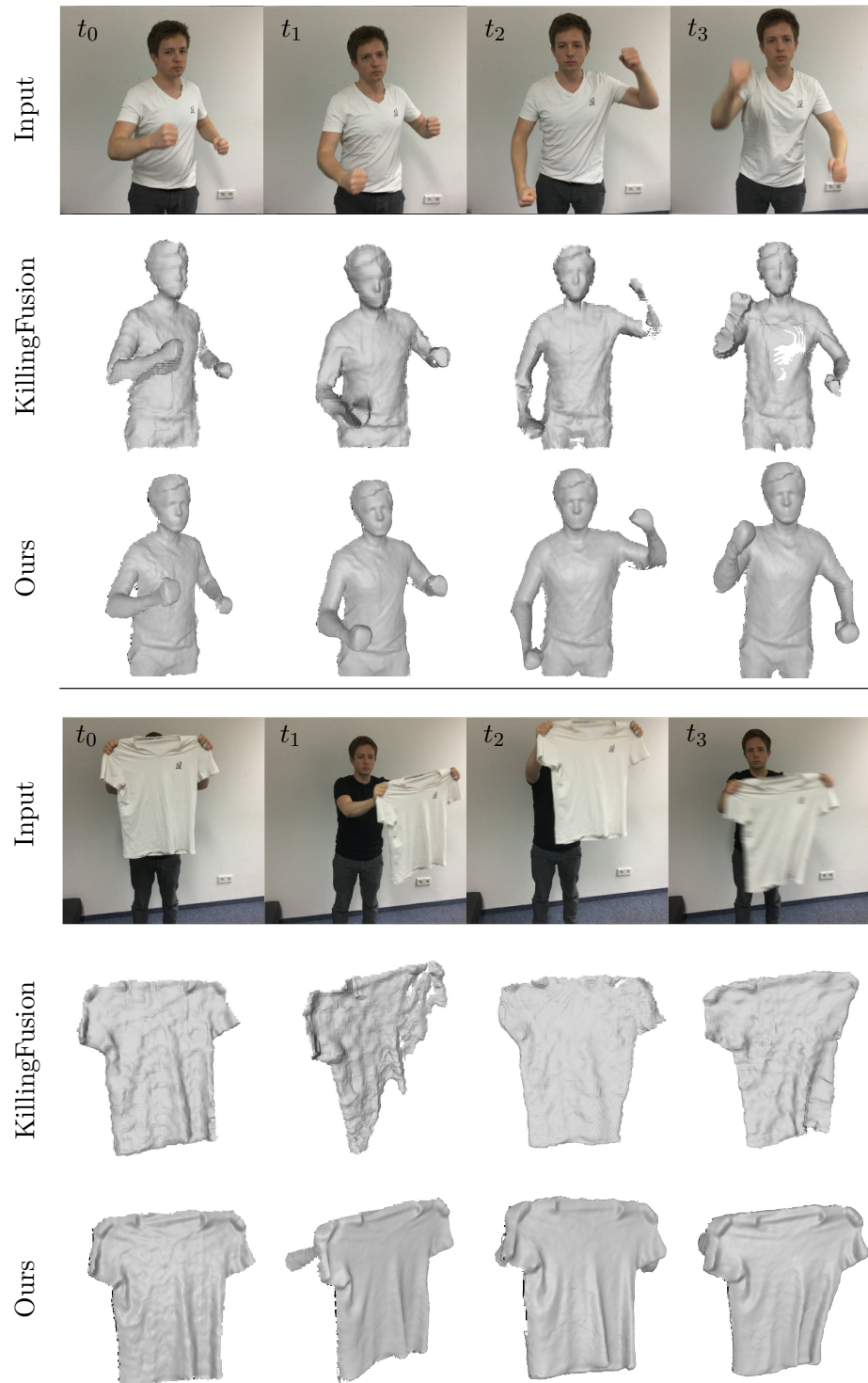


Figure 4.7: Qualitative comparison with KillingFusion [62]. The results on test sequences from [13] were kindly provided by the authors.

Part II. Non-rigid Reconstruction using Data-driven Priors

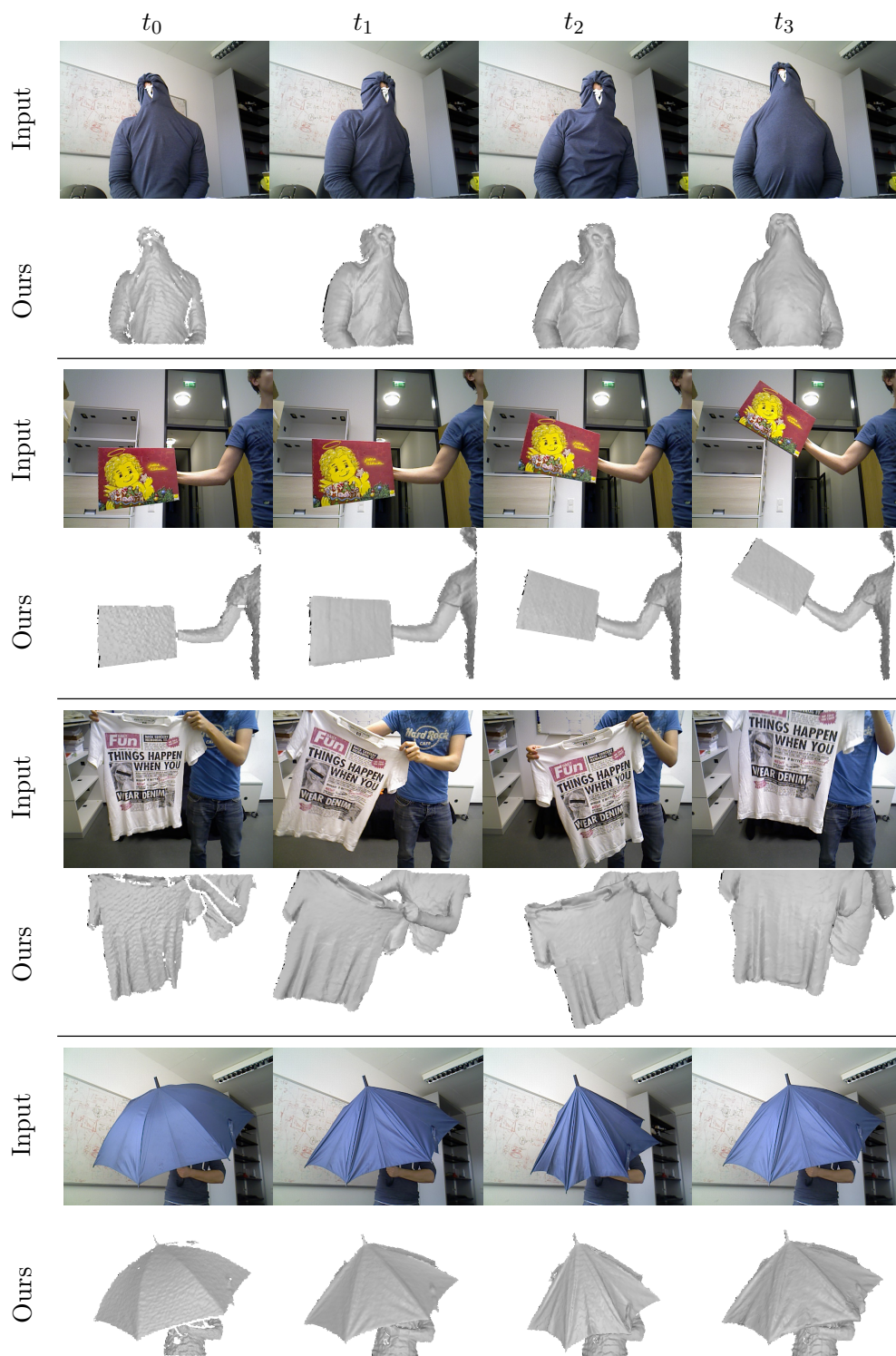


Figure 4.8: **Generalization to a different sensor.** We show qualitative reconstruction results on VolumeDeform [35] sequences, captured by a Microsoft Kinect sensor.

5 Neural Deformation Graphs

Abstract of paper. We introduce Neural Deformation Graphs for globally-consistent deformation tracking and 3D reconstruction of non-rigid objects. Specifically, we implicitly model a deformation graph via a deep neural network. This neural deformation graph does not rely on any object-specific structure and, thus, can be applied to general non-rigid deformation tracking. Our method globally optimizes this neural graph on a given sequence of depth camera observations of a non-rigidly moving object. Based on explicit viewpoint consistency as well as inter-frame graph and surface consistency constraints, the underlying network is trained in a self-supervised fashion. We additionally optimize for the geometry of the object with an implicit deformable multi-MLP shape representation. Our approach does not assume sequential input data, thus enabling robust tracking of fast motions or even temporally disconnected recordings. Our experiments demonstrate that our Neural Deformation Graphs outperform state-of-the-art non-rigid reconstruction approaches both qualitatively and quantitatively, with 64% improved reconstruction and 54% improved deformation tracking performance.

5.1 Introduction

Capturing non-rigidly deforming surfaces is essential towards reconstructing and understanding real-world environments, which are often highly dynamic. While impressive advances have been made in reconstructing static 3D scenes [37], [127], dynamic tracking and reconstruction remains very challenging. A plethora of domain-specific dynamic tracking methods has been developed (e.g., human bodies, faces, hands), leveraging strong domain shape and motion priors for robust tracking [6]–[8], [128]. However, real-world environments encompass a vast diversity of deformable objects – including people with clothing or animals – making domain specific shape priors often intractable for general deformable reconstruction; in this work, we thus focus on general non-rigid 3D reconstruction without shape or motion priors for general object tracking and reconstruction.

A seminal work in non-rigid 3D reconstruction is DynamicFusion [4], which was the first approach to demonstrate real-time dense reconstruction of dynamic scenes using just a single RGB-D sensor. DynamicFusion showed promising results towards dynamic reconstruction, but still struggles in many real-world scenarios, which typically include strong deformations and fast frame-to-frame motion, due to its low-level, local correspondence association step. In particular, the incremental construction of a deformation graph is prone to error aggregation and can lead to tracking failures. Recently, data-driven methods based on deep learning have been introduced [13], [14], [21] that learn

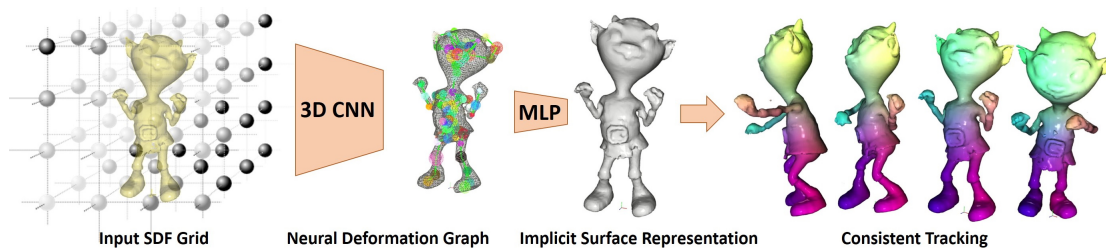


Figure 5.1: Neural Deformation Graphs. Given range input data, represented as a signed distance field, our method predicts globally-consistent deformation graph that is used to reconstruct the non-rigidly deforming surface of an object. The surface of the object is represented as a set of implicit functions centered around the deformation graph nodes. Our global optimization provides consistent surface and deformation prediction, enabling robust tracking of an observed input sequence and even multiple disjoint captures of the same object (as we do not assume sequential input data).

priors of non-rigidly deforming objects from dense flow annotations. These approaches leverage a similar incremental deformation graph construction as DynamicFusion, but learn to establish more robust tracking via more sophisticated correspondence optimization based on data-driven priors. However, despite more robust correspondences, these methods still operate on a frame-by-frame basis, thus, aggregate tracking errors and are unable to recover if tracking fails. In order to address these shortcomings without assuming data-driven priors, we propose a globally-consistent neural deformation graph which allows for non-rigid reconstruction from commodity sensor observations, represented as signed distance fields (see Fig. 5.1). The neural deformation graph gives access to the per frame deformation graph nodes and stores the global graph connectivity. To robustly optimize for consistent deformations over fast motions, we introduce viewpoint consistency (independently for every frame) as well as graph and surface consistency constraints (between pairs of frames). Our viewpoint consistency loss measures the consistency of graph node position predictions w.r.t. rotation augmentation. The graph and surface consistency losses encourage deformations to be modeled in our Neural Deformation Graph such that local graph edge distances are preserved between frames and the deformed surface geometry of a source frame aligns well with the geometry of the target frame. Additionally, our approach does not assume temporally close frames, thus making it easily applicable to low FPS settings or the combination of independently captured depth recordings.

Since there exists no general canonical pose (like a T-pose of a human [6]) that fits all deformable objects, we avoid modeling it explicitly. Instead, we propose to employ a set of implicit functions that are centered around the deformation graph nodes. Specifically, we model local signed distance functions (SDFs) using multi-layer perceptrons (MLPs) that can be deformed to fit any frame, without requiring an explicit canonical pose. The global shape is evaluated by the integration of these local MLP predictions.

To summarize, our technical contributions are:

- a globally-optimized deformation graph that is able to handle deformations present in all frames of an unstructured dataset or a sequence of an object;
- a combination of per-frame viewpoint consistency and frame-to-frame graph and surface consistency for robust tracking of fast deformations;
- an implicit deformable multi-MLP shape representation anchored on the scene-specific deformation graph.

5.2 Related Work

Our approach is leveraging a low dimensional deformation graph to model the non-rigid deformations of an object, while the actual surface is represented by an implicit function by means of a multi-layer perceptron (MLP). We will discuss the most related approaches in these two fields.

Non-rigid Reconstruction. Non-rigid reconstruction is a highly active research field, in particular using commodity RGB-D sensors such as the Kinect. The seminal work DynamicFusion of Newcombe et al. [4] tracks deformable motion and reconstructs the object’s shape in an incremental fashion, i.e., frame-by-frame. While this approach relies on local depth correspondences, follow-up methods additionally use sparse SIFT features [35], dense color tracking [64] or dense SDF alignment [62], [63]. These methods show impressive results, but often struggle with fast frame-to-frame motion given their use of hand-crafted correspondences. Bozic et al. [13] introduced an annotated dataset of non-rigid motions that allows to train data-driven non-rigid reconstruction methods with learned correspondences [13], [14], [21]. While learned correspondences improve tracking performance, the approaches are still inherently limited by the employed frame-to-frame tracking paradigm, i.e., tracking errors accumulate over time, and if tracking is lost it is unable to recover. Tracking robustness can also be improved without any learned priors by using multi-view input data [129], [130] (setups with more than 50 cameras) and high-speed cameras [67] (8 cameras at 200 frames per second (FPS)). In contrast to these frame-to-frame tracking approaches, there are methods that focus on global non-rigid optimization [65], [86], [131], [132]; however, these methods either assume ground-truth optical flow [132], or they share the same drawbacks of the aforementioned frame-to-frame tracking approaches [65], [86], [131], and thus have difficulties handling fast deformable motion.

Deformation Graphs. State-of-the-art non-rigid reconstruction methods often model deformations with a sparse deformation graph, following the Embedded Deformation [24] formulation. Deformation graphs offer a robust alternative to dense motion estimation with optical flow or scene flow methods, since they can estimate plausible motion even in partially occluded shape parts, when combined with motion regularization such as

ARAP [25]. Existing non-rigid reconstruction approaches build the deformation graph incrementally, i.e., frame-by-frame, which can lead to unstable graph configurations in the case of tracking errors. In our approach, we predict a globally consistent deformation graph that can represent motion in all frames of the sequence, while being robust w.r.t. tracking errors present in challenging frames.

Sparse motion representations are common for human deformation modeling as well: the human skeletons used in [6], [8] are also instances of deformation graphs. Some works have tried to extend human skeletons to more general objects, but with limited success. In [133], a fixed generic skeleton is fitted to different object meshes, resulting in human-like re-animation of characters, but not general enough to be able to represent all degrees of freedom of general shapes. Fixed hierarchical deformation graphs are used for differentiable non-rigid tracking in [134], but a pre-computed graph template is required, with fixed connectivity on different coarseness levels. Thus, it is only applicable to specific object types (e.g., used for hand tracking). Data-driven skeleton prediction has been introduced in [135], but it requires a dataset of manually designed skeletons as supervision, which is hard to obtain for general objects. Our method, instead, estimates both deformation graph nodes and connectivity of general deformable objects in a self-supervised manner.

Implicit Surface Representation. Representing surface geometry implicitly with a signed distance field (SDF) has been extensively used in the non-rigid reconstruction community. An efficient algorithm for SDF grid construction from range images has been presented in [37] and extended to support non-rigid deformations in [4]. These methods rely on a discretized 3D grid to store the SDF, which can cause loss of detail, since grid resolution is limited by available memory. A promising direction is to not use discretized grids at all, but instead represent the SDF function continuously using a multi-layer perceptron (MLP), as introduced in [40], [41], [48]. An implicit surface representation is used in [136] for accurate human reconstruction, where the SDF is estimated in a canonical T-pose space. Since there exist no methods for estimation of canonical T-pose spaces for general non-rigid shapes, we instead base our method on the approach of Deng et al. [44]. Assuming ground-truth dense body and skeleton tracking, they represent the human body with multiple MLPs, one for each bone and in its own canonical space, centered around the bone, therefore eliminating the need for a T-pose space. In our general reconstruction approach, we estimate a deformation graph via self-supervision, and append an MLP to each deformation node to represent the surface of the observed object.

While most implicit reconstruction approaches do not produce consistent tracking, methods such as [52], [53], [137] reconstruct objects in a patch-based manner and empirically observe consistency of patches across different deformations. We compare our method, which leverages explicit consistency constraints, to these approaches to evaluate such implicit patch consistency.

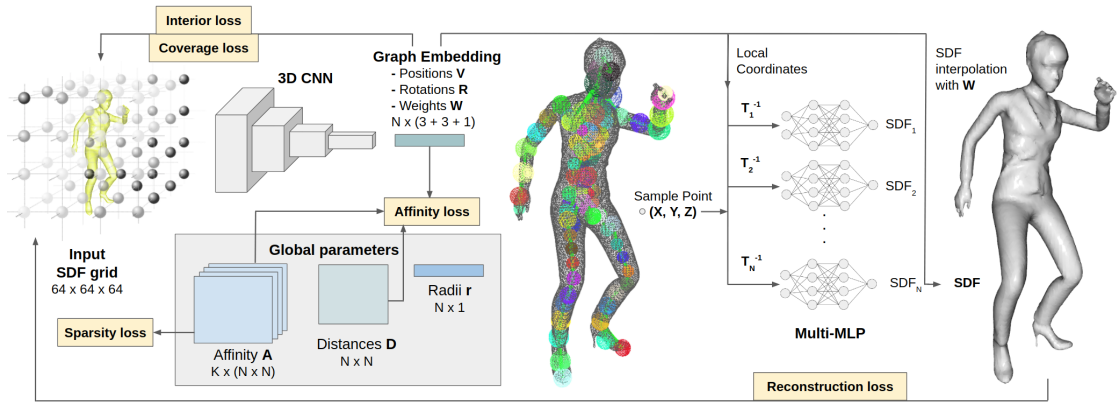


Figure 5.2: Method overview. A Neural Deformation Graph encodes a 64^3 SDF grid to a graph embedding with graph node positions \mathbf{V} , rotations \mathbf{R} and importance weights \mathbf{W} . To compute an SDF value for a sample point $(X, Y, Z) \in \mathbb{R}^3$, the point is transformed to local coordinates around each node, and passed through locally embedded implicit functions that are represented as MLPs; the global SDF value is computed by interpolating the local MLP predictions using the node radii \mathbf{r} and importance weights \mathbf{W} . For graph regularization, a set of affinity matrices $\mathbf{A}_i \in \mathbb{R}^{N \times N}$ and a node-to-node distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ are globally optimized.

5.3 Method

Given a sequence of signed distance fields observing a non-rigidly deforming surface, our method estimates the dense deformable motion in the sequence as well as reconstructs the geometry of the observed shape. Specifically, we apply self-supervised learning on the sequence that we want to reconstruct. A convolutional neural network that takes a signed distance field (SDF) as input is trained to predict a consistent deformation graph. We call this neural network *Neural Deformation Graph*, as it implicitly stores the deformation graphs of each frame. Using the predicted graph node positions and orientations, we learn implicit functions to represent the shape of each graph node and, thus, the entire shape of the object. The implicit functions are represented as a multi-layer perceptron (MLP). These MLPs take a 3D point centered around the node position as input and predict its signed distance value, defining the local geometry around the node. Warping all node MLPs to every time step and interpolating their local part reconstructions results in an accurate *implicit deformable shape reconstruction* without the need for an explicit canonical pose. In addition to the sample point locations, the MLPs are conditioned on the predicted graph positions, which enables reconstruction of pose-dependent geometry detail. Using Marching Cubes [18], the geometry can be extracted as a mesh at every time step, with dense correspondences estimated throughout the entire sequence.

5.3.1 Neural Deformation Graphs

A deformation graph consists of graph nodes and graph edges. We represent the graph nodes \mathcal{V} of each frame of the sequence implicitly by a neural network (Neural Deformation Graph). The graph connectivity is explicitly stored for the entire sequence as an affinity matrix $\mathcal{E} \in \mathbb{R}^{N \times N}$ ($N = |\mathcal{V}|$). A node $v \in \mathcal{V}$ is characterized by its 3D position $\mathbf{v} \in \mathbb{R}^3$, rotation $R \in \mathbb{R}^{3 \times 3}$, importance weight $w \in \mathbb{R}$ (describing importance of the node, explained below), radius $r \in \mathbb{R}$ (describing the spatial influence of the node), and a local implicit shape function f . We denote the set of node positions $\mathbf{V} = \{\mathbf{v}\}$, rotations $\mathbf{R} = \{R\}$, importance weights $\mathbf{W} = \{w\}$, radii $\mathbf{r} = \{r\}$, and shape functions $\mathbf{f} = \{f\}$, with $\mathcal{V} = (\mathbf{V}, \mathbf{R}, \mathbf{W}, \mathbf{r}, \mathbf{f})$.

To generate the signed distance fields needed for our method, we assume four calibrated cameras. The depth maps at each time step k are back-projected into a common coordinate system and converted into a signed distance field \mathbf{S}_k of dimension 64^3 using static volumetric reconstruction [138]. Note that due to occlusions this representation is partial, thus only an approximate signed distance field is used for our deformation graph prediction. Based on this input, we estimate $(\mathbf{V}_k, \mathbf{R}_k, \mathbf{W}_k)$ using a Neural Deformation Graph (NDG) which is based on a 3D convolutional neural network:

$$(\mathbf{V}_k, \mathbf{R}_k, \mathbf{W}_k) = \text{NDG}(\mathbf{S}_k)$$

The radii \mathbf{r} of the graph nodes as well as the graph node affinities \mathcal{E} are jointly optimized over the entire input sequence. In addition to the affinity matrix, we also store the average edge lengths (node-to-node distances) $\mathbf{D} \in \mathbb{R}^{N \times N}$, which are used for regularization. For every graph node, we also optimize for a local MLP which is used to represent the surface of the object (see Sec. 5.3.3).

We define a fixed number of graph nodes ($N = 100$) in our experiments; note that this is an upper bound on the effective number of nodes, since the importance weights allow eliminating the effect of redundant nodes, making our method applicable to shapes of different size and structure complexity. To achieve a consistent graph node prediction via self-supervised training, we employ the following constraints for each time-step k .

Graph coverage loss. A deformation graph should cover the entire object to ensure that every deformable part can be represented while simultaneously enforcing that free space is not covered. To this end, we employ a loss that encourages the coverage of the shape by the node centers (w.r.t. their radii). We define the influence of a node (with position \mathbf{v} , radius $r > 0$, and importance weight $w > 0$) on a point $\mathbf{x} \in \mathbb{R}^3$ using a *weighted Gaussian function*:

$$\mathcal{G}(\mathbf{x}, \mathbf{v}, r, w) = w \cdot \exp\left(-\frac{\|\mathbf{x} - \mathbf{v}\|_2^2}{r^2}\right)$$

The coverage of a point $\mathbf{x} \in \mathbb{R}^3$ is computed by summing the corresponding contributions of all nodes, and applying a *sigmoid* to encourage a fast transition from covered (where

coverage value is 1) to free space (where coverage value should be 0), enabling more accurate surface coverage:

$$\mathcal{C}(\mathbf{x}, \mathbf{V}_k, \mathbf{r}, \mathbf{W}_k) = \sigma \left(s \left(\left(\sum_{\mathbf{v}, r, w} \mathcal{G}(\mathbf{x}, \mathbf{v}, r, w) \right) - d \right) \right)$$

We empirically set $d = 0.07$ and $s = 100.0$. To compute the coverage loss, we sample points P_{un} uniformly in the shape’s bounding box and points P_{ns} near the surface region. Points are assigned coverage value of $c = 0$ if they are visible in at least one of the cameras, otherwise they are assigned $c = 1$. The coverage loss then compares predicted coverage of these point samples with the pre-computed coverage using an ℓ_2 loss:

$$\mathcal{L}_{\text{coverage}} = \lambda_{\text{un}} \sum_{(\mathbf{x}, c) \in P_{\text{un}}} \|\mathcal{C}(\mathbf{x}, \mathbf{V}_k, \mathbf{r}, \mathbf{W}_k) - c\|_2^2 + \lambda_{\text{ns}} \sum_{(\mathbf{x}, c) \in P_{\text{ns}}} \|\mathcal{C}(\mathbf{x}, \mathbf{V}_k, \mathbf{r}, \mathbf{W}_k) - c\|_2^2$$

Node interior loss. In addition to the graph coverage loss, we require the node positions to be predicted inside the shape. If any node’s position \mathbf{v} is predicted outside the observed surface \mathbf{S}_k , i.e., in the SDF region with positive signed distance value, we penalize it to encourage the node’s position to be inside the surface:

$$\mathcal{L}_{\text{interior}} = \sum_{\mathbf{v} \in \mathbf{V}_k} \max(\text{interp}(\mathbf{S}_k, \mathbf{v}), 0)$$

Here $\text{interp}(\mathbf{S}_k, \mathbf{v})$ is the trilinear interpolation of \mathbf{S}_k at \mathbf{v} .

Affinity consistency loss. We also optimize for a global affinity matrix $\mathcal{E} = \{e_{ij} \mid i \in [1, N], j \in [1, N]\}$ representing node-to-node affinities across the entire input sequence. We compute node-to-node Euclidean distances $\|\mathbf{v}_i^k - \mathbf{v}_j^k\|_2$ at each frame k , and weight them by connectivity weights e_{ij} ; this should remain consistent over the whole sequence (relative loss, preserving edge length) and relatively small (absolute loss, preferring close-by connections). To ensure global distance consistency, we additionally optimize over average node-to-node distances d_{ij} , resulting in the affinity loss:

$$\mathcal{L}_{\text{affinity}} = \lambda_{\text{rel}} \sum_{i \neq j} e_{ij} \left| d_{ij}^2 - \|\mathbf{v}_i^k - \mathbf{v}_j^k\|_2^2 \right|_1 + \lambda_{\text{abs}} \sum_{i \neq j} e_{ij} \|\mathbf{v}_i^k - \mathbf{v}_j^k\|_2^2$$

Neighbor diversity loss. We enforce a sparse connectivity of the graph. Specifically, each node can have up to K neighbors ($K = 2$ in our setting); we use a (soft) loss to encourage these neighbors to be different. To achieve this, we optimize over a set of matrices $\mathbf{A}_1, \dots, \mathbf{A}_K \in \mathbb{R}^{N \times N}$, and construct $\mathcal{E} \in \mathbb{R}^{N \times N}$ as:

$$\mathcal{E} = \frac{1}{K} \sum_{i=1}^K \text{softmax}(\mathbf{A}_i)$$

We use softmax over the rows of matrix \mathbf{A}_i to guarantee all affinity elements of a node to be positive and add up to 1. To enforce unique graph neighbors, a neighbor diversity loss is employed, encouraging different matrices \mathbf{A}_i to produce different neighbors:

$$\mathcal{L}_{\text{sparsity}} = \sum_{l \neq m} \|\text{softmax}(\mathbf{A}_l) \odot \text{softmax}(\mathbf{A}_m)\|_F^2$$

We use \odot to denote the element-wise product.

5.3.2 Global Deformation Optimization

We compute deformation between any pair of frames by interpolating the nodes' relative motions (translations and rotations), weighted by their influences \mathcal{G} . For a source frame s and target frame t , the warping of point $\mathbf{x} \in \mathbb{R}^3$ from frame s to frame t is defined as:

$$\mathbf{Q}_{s \rightarrow t}(\mathbf{x}) = \sum_{i=1}^N \mathcal{G}(\mathbf{x}, \mathbf{v}_i^s, r_i, w_i^s) (\mathbf{R}_i^t (\mathbf{R}_i^s)^T (\mathbf{x} - \mathbf{v}_i^s) + \mathbf{v}_i^t)$$

We denoted parameters at the source frame with $(\cdot)^s$ and at the target frame with $(\cdot)^t$. We use the Embedded Deformation formulation [24] to parameterize frame-to-frame deformation, but instead of fixed-radius skinning we employ node influence \mathcal{G} as the skinning weight, which enables different skinning effects for every node as well as frame-adaptive skinning, i.e., skinning can change depending on the deformation. To ensure globally consistent deformation, we employ a per-frame viewpoint consistency loss and a surface consistency loss.

Viewpoint consistency loss. Since input observations may see very different views, we enforce a viewpoint consistency loss for consistent graph node predictions across varying views. To this end, for each frame k , the rotated 3D input \mathbf{S}_k should produce consistent graph node positions \mathbf{V}_k , rotations \mathbf{R}_k and importance weights \mathbf{W}_k . In our experiments, we only consider view rotations around the y -axis, since the camera setup is arranged in the x - z plane. In each batch, we sample two random angles α and β for every sample, and compute rotated inputs $\pi_\alpha(\mathbf{S}_k)$ and $\pi_\beta(\mathbf{S}_k)$ by trilinear re-sampling of input SDF grid \mathbf{S}_k using rotated grid indices. Viewpoint consistency is then measured by:

$$\mathcal{L}_{\text{vc}} = \|\pi_\alpha^{-1} \text{NDG}(\pi_\alpha(\mathbf{S}_k)) - \pi_\beta^{-1} \text{NDG}(\pi_\beta(\mathbf{S}_k))\|_2^2$$

where the function π_ϕ^{-1} corrects for the input rotation of angle ϕ : $\pi_\phi^{-1}(\mathbf{V}_k, \mathbf{R}_k, \mathbf{W}_k) = (R_\phi^T \mathbf{V}_k, R_\phi^T \mathbf{R}_k, \mathbf{W}_k)$.

Surface consistency loss. Surface points from a source frame s should, after deformation to a target frame t , align well with the target frame's SDF grid \mathbf{S}_t . We sample surface points P_s in the source frame and warp them to the target frame using the predicted deformation, to trilinearly interpolate the target grid \mathbf{S}_t , encouraging surface

points to be warped to near zero (surface) SDF values:

$$\mathcal{L}_{\text{sc}} = \sum_{\mathbf{x} \in P_s} (\text{interp}(\mathbf{S}_t, \mathbf{Q}_{s \rightarrow t}(\mathbf{x})))^2$$

This consistency loss is computed between pairs of samples in the batch, with uniformly sampled batch samples.

5.3.3 Implicit Surface Reconstruction

We represent the surface of the object as an implicit function. Specifically, each graph node i defines local geometry over the influence of that node, with an implicit function f_i , represented by an MLP. This MLP takes a location in the local space as input and outputs an SDF value. Any point $\mathbf{x} \in \mathbb{R}^3$ in the current frame k can be transformed to the local coordinate system of node i as $\mathbf{Q}_{k,i}^{-1}(\mathbf{x}) = \mathbf{P}_{\text{enc}}((\mathbf{R}_i^k)^T(\mathbf{x} - \mathbf{v}_i^k))$. $\mathbf{P}_{\text{enc}} : \mathbb{R}^3 \rightarrow \mathbb{R}^F$ denotes positional encoding that transforms 3D local coordinates to a high-dimensional frequency domain (in our case $F = 30$), as presented in [139]. Inspired by Deng et al. [44], we condition each f_i on the predicted input frame’s graph parameters, such that they can encode pose-specific geometry details. We train a linear layer $\Pi_i(\cdot)$ to select a sparse pose code (of dimension $D = 32$) for every f_i from the graph predictions $\text{NDG}(\mathbf{S}_k)$. Given this input of dimension $D + F$, we use 8 linear layers with feature dimension of 32, a leaky ReLU (with negative slope of 0.01) as activation function, and skip connections between the input and the 6th linear layer.

We compute the full surface reconstruction \mathcal{S}_k as an SDF created from interpolating the SDF output values of each local MLP f_i , using the aforementioned skinning weights and transformations to the current frame by the estimated nodes’ rotations and translations:

$$\mathcal{S}_k(\mathbf{x}) = \sum_{i=1}^N \mathcal{G}(\mathbf{x}, \mathbf{v}_i^k, r_i, w_i^k) f_i(\mathbf{Q}_{k,i}^{-1}(\mathbf{x}), \Pi_i(\text{NDG}(\mathbf{S}_k)))$$

This operation is efficiently implemented using group convolutions. During training, we use the same point samples P_{un} and P_{ns} as for the graph coverage loss, sampled uniformly and near the surface, but instead of the 0/1 coverage values we use their approximate SDF values. We then optimize for $\{f_i\}$ using the SDF reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \sum_{(\mathbf{x}, \text{sdf}) \in P_{\text{un}} \cup P_{\text{ns}}} |\mathcal{S}_k(\mathbf{x}) - \text{sdf}|_1$$

5.3.4 Training Details

We use the Adam solver [12] with momentum of 0.9 to optimize the complete loss:

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_{\text{coverage}} + \lambda_{\text{interior}} \mathcal{L}_{\text{interior}} + \mathcal{L}_{\text{affinity}} + \\ & \lambda_{\text{sparsity}} \mathcal{L}_{\text{sparsity}} + \lambda_{\text{vc}} \mathcal{L}_{\text{vc}} + \lambda_{\text{sc}} \mathcal{L}_{\text{sc}} + \lambda_{\text{recon}} \mathcal{L}_{\text{recon}} \end{aligned}$$

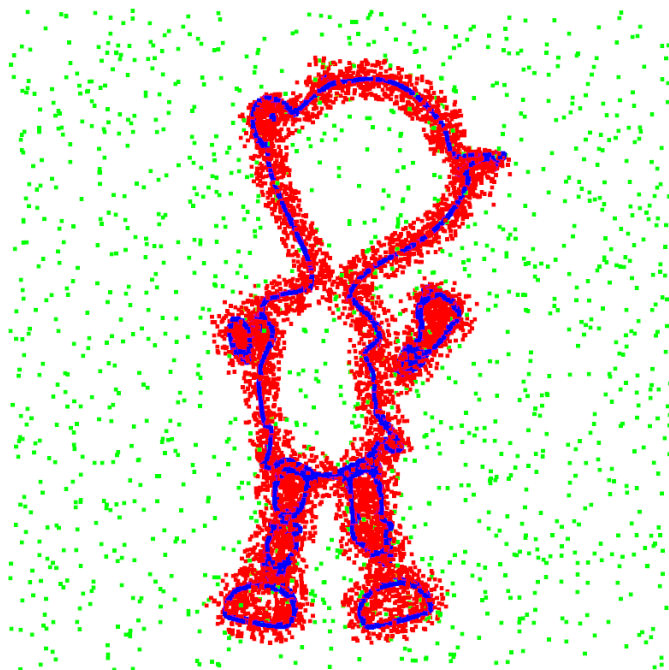


Figure 5.3: Point sample visualization. We visualize the uniform samples P_{un} (*green*), near-surface samples P_{ns} (*red*) and the surface samples P_{s} (*blue*) for a slice of samples with $z \in [-0.01, 0.01]$ at one frame of the character sequence shown in the Fig. 5.1.

Our method is trained in two stages. We initially train the CNN encoder with all losses except the reconstruction loss, and afterwards train the multi-MLP network using only the reconstruction loss, with the CNN encoder frozen.

CNN encoder optimization. The CNN encoder is trained for 500k iterations with a learning rate of $5e^{-5}$ and a batch size of 16; we balance the losses with $\lambda_{\text{un}} = 1.0$, $\lambda_{\text{ns}} = 0.1$, $\lambda_{\text{interior}} = 1.0$, $\lambda_{\text{rel}} = 0.1$, $\lambda_{\text{abs}} = 0.1$, $\lambda_{\text{sparsity}} = 1e^{-8}$, $\lambda_{\text{vc}} = (10.0, 1.0, 1e^{-4})$ (for graph node’s position, weight and rotation, respectively) and $\lambda_{\text{sc}} = 1e^{-6}$. Every 50k iterations we increase the loss weights λ_{rel} , λ_{abs} , $\lambda_{\text{sparsity}}$, λ_{sc} by a factor of 10, up to maximum weights $\lambda_{\text{rel}}^{\text{max}} = 10000.0$, $\lambda_{\text{abs}}^{\text{max}} = 1.0$, $\lambda_{\text{sparsity}}^{\text{max}} = 1e^{-3}$ and $\lambda_{\text{sc}}^{\text{max}} = 1000.0$. Note, when interpolating the SDF grid for the node interior loss, in the case that a graph node’s position is predicted outside the grid, we define the out-of-grid loss by an ℓ_2 -distance to the nearest bounding box corner. This encourages graph node positions to be always predicted inside the shape’s bounding box.

Multi-MLP network optimization. The multi-MLP network is trained for 500k iterations with a learning rate of $5e^{-4}$ and a batch size of 4, only based on the reconstruction loss with $\lambda_{\text{recon}} = 1.0$. We use a truncation of 0.1 (in normalized units of the object in the unit cube) for the signed distance field used for the reconstruction loss.

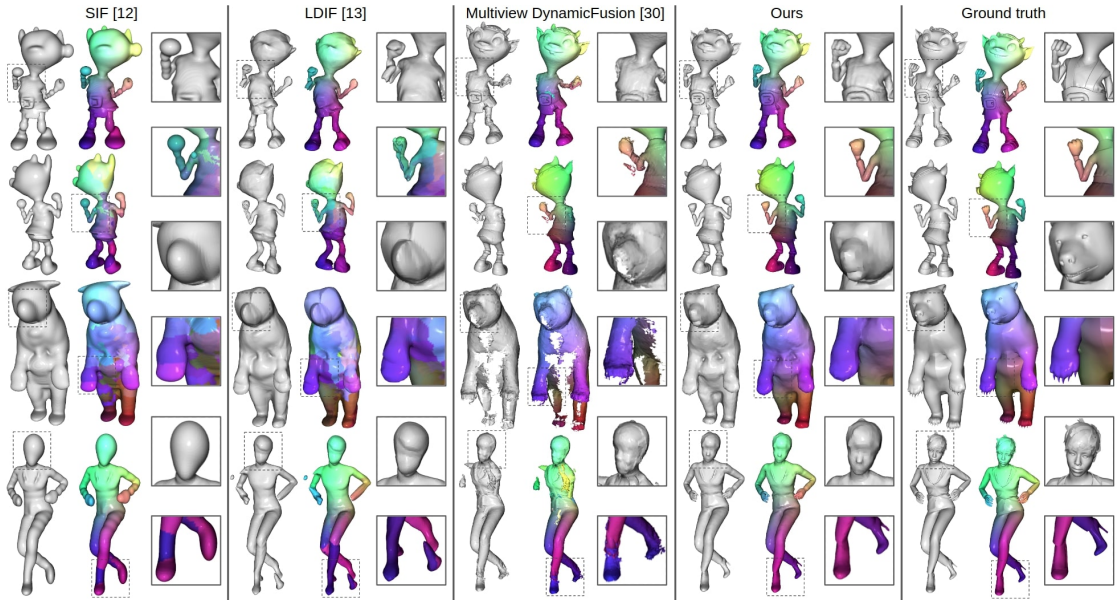


Figure 5.4: Qualitative comparison on synthetic sequences. We qualitatively compare our method to the baseline methods on synthetic data. Each point is given a color value w.r.t. its location in the bounding box of the first frame. With perfect tracking and reconstruction, a specific point on the surface will have the same color throughout the entire sequence, while errors in tracking result in changing surface colors. Our approach outperforms state of the art in both reconstruction and deformation tracking quality.

Point sampling. We visualize an example of point samples in Fig. 5.3. When training the neural deformation graph, we use $|P_{\text{un}}| = 3000$ uniform point samples, $|P_{\text{ns}}| = 3000$ near surface point samples and $|P_{\text{s}}| = 3000$ surface point samples, sampled randomly for each batch from 100k pre-processed point samples. For the graph coverage loss, we apply an additional weight of 10.0 for interior point samples (determined by the SDF sign). To train our multi-MLP network that implicitly represents the surface, we use $|P_{\text{un}}| = 1500$ uniform point samples and $|P_{\text{ns}}| = 1500$ near surface point samples. Note that this reduced set of samples is applied to satisfy memory limits of our used GPU (Nvidia Geforce 2080Ti).

5.4 Results

To evaluate our proposed approach, we conducted a series of experiments on real and synthetic recordings where ground truth data is available.

Evaluation on synthetic data. In order to quantitatively and qualitatively evaluate our method, we make use of synthetic human-like and character sequences from the DeformingThings4D dataset [141]. To mimic our real data capture setup, we render

Method	Chamfer	EPE3D
SIF [52]	1.12	8.56
LDIF [53]	2.41	10.40
OccupancyFlow [140]	53.83	16.29
Multiview DynamicFusion [4]	2.19	3.06
MV DF [4] + FlowNet3D [113]	1.93	2.55
Robust L0 Non-rigid Tracking [103]	2.31	2.50
Ours: GRAPH	0.50	8.93
Ours: GRAPH + AO	0.46	8.03
Ours: GRAPH + AO + VC	0.44	4.12
Ours: GRAPH + AO + VC + SC	0.40	1.16

Table 5.1: Quantitative comparison. We show quantitative comparisons with state-of-the-art approaches, evaluating geometry using chamfer distance ($\times 10^{-4}$), and deformation using EPE3D ($\times 10^{-2}$). We also include an ablation study of different components of our method: GRAPH = coverage and interior losses, AO = affinity optimization with affinity consistency and sparsity, VC = viewpoint consistency, SC = surface consistency.

4 fixed depth views for every frame of synthetic animation, and generate SDF grids from these 4 views. Quantitative evaluation is executed on three sequences, including human, character and bear motion, as shown in Fig. 5.4. The geometry reconstruction is evaluated using *L2 Chamfer distance*, which computes average squared bi-directional point-to-point distance between reconstructed and ground truth geometry for every time step, thus evaluating accuracy and completeness of geometry. For deformation evaluation, we uniformly sample 10 keyframes per sequence, and compute dense deformation from each of these keyframes to any other frame, measuring average *L2 End-Point-Error (EPE3D)* between estimated and ground truth motion. The depth data of every sequence is normalized such that the largest bounding box side length is equal to 1.0. All numbers are listed w.r.t. this unit cube, thus, being independent to the scale of the objects.

In Tab. 5.1 we quantitatively compare our approach to the state-of-the-art network-based reconstruction methods SIF [52], LDIF [53] and OccupancyFlow [140], as well as to the non-rigid reconstruction approach DynamicFusion [4], which we extend to the multi-view domain, and the Robust L0 Non-rigid Tracking method [103]. Among the baselines the best reconstruction performance (lower Chamfer distance) is achieved by SIF [52], while the Robust L0 Non-rigid Tracking method [103] obtains better deformation tracking performance (lower EPE3D). Our approach outperforms all methods on both reconstruction and deformation tracking metrics, achieving 64% better reconstruction and 54% better deformation tracking results. The improvement is also clearly visible in the qualitative comparisons shown in Fig. 5.4. The methods SIF [52] and LDIF [53] produce less accurate geometry reconstruction with tracking failures under larger defor-

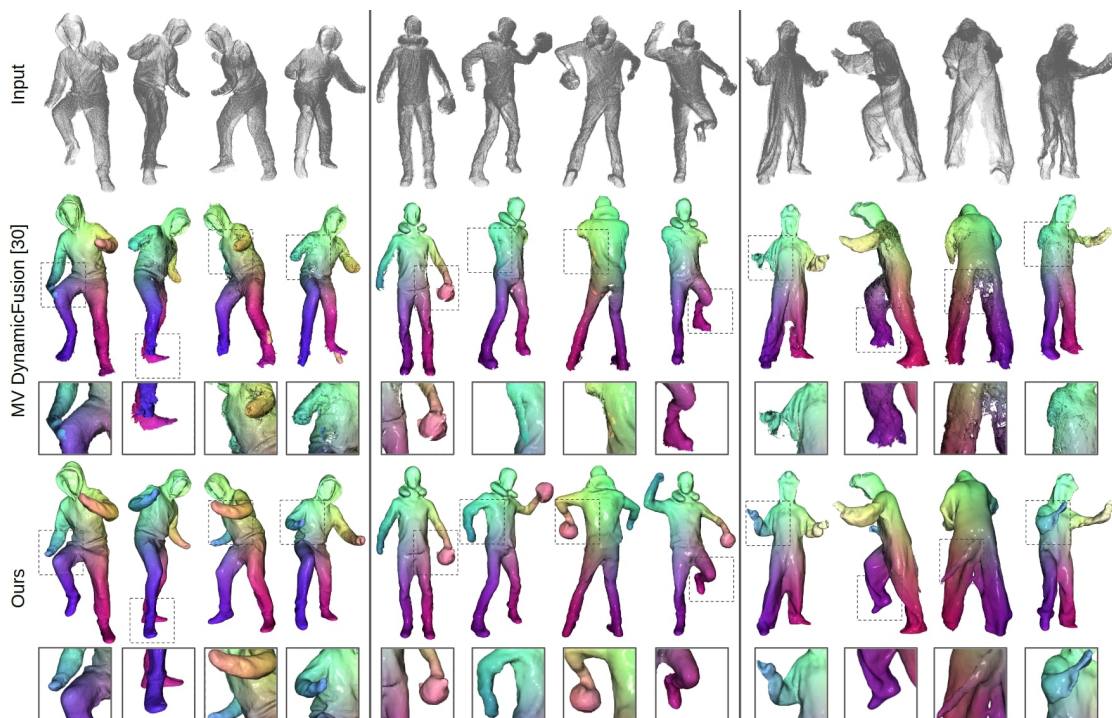


Figure 5.5: Qualitative comparison on real sequences. We show qualitative comparisons of our method with multi-view DynamicFusion [65] on real sequences captured by four Kinect Azure sensors. Colors represent corresponding locations in the first frame of the sequence to visualize the tracking quality and consistency.

mations (e.g., flipped legs in the human sequence). We trained Occupancy Flow [140] on our sequences, which are noticeably longer (about 500 frames) than the sequences the authors used (up to 50 frames), resulting in worse performance. The multi-view DynamicFusion [4] baseline is a specialized framework for both non-rigid tracking and reconstruction, with coarse-to-fine multi-frame alignment using depth iterative closest point (ICP) correspondences and non-rigid volumetric fusion. We use data-driven correspondences from the off-the-shelf flow estimator FlowNet3D [113] to further improve the method. However, it suffers from incomplete geometry because of the incremental graph construction and surface integration, and can also not recover from tracking failures. In contrast, our method is robust in the case of large deformation and produces complete and accurate geometry reconstruction.

Evaluation on real data. Our real data capture setup consists of 4 Kinect Azure sensors, as shown in Fig. 5.6. All four sensors are connected to the same computer via USB-C cables and are hardware-synced using a daisy-chain configuration (connecting the trigger of the master camera with the other 3 cameras in a chain). To avoid interference between depth sensors, we set a delay of 160 microseconds between different depth camera captures. The cameras are calibrated with a checkerboard using OpenCV [142]

and an additional refinement procedure based on ICP [143]. Before recording an actual sequence, we record the background to compute the floor plane using PCA. During capture, we filter out floor points and background points, i.e., all points outside of a cylinder with diameter 1.8 m and height 2.5 m. We use the wide-field-of-view depth capture setting with a resolution of 1024×1024 pixels, at the highest available frame-rate of 15 FPS for this resolution. In Fig. 5.5, we show a comparison between the multi-view DynamicFusion approach [4] and ours. Our approach achieves considerably more accurate deformation tracking (color is retrieved from the first frame) while also producing more complete and accurate geometry reconstruction. More qualitative results are shown in the accompanying video. We also visualize some examples of optimized neural deformation graphs in Fig. 5.8.

Ablation study of network components. To evaluate specific parts of our method, we employ an ablation study. Specifically, we analyzed the performance of our method by performing optimization without using certain losses: without affinity related losses (affinity consistency and sparsity losses), viewpoint consistency loss and surface consistency loss. As shown in Tab. 5.1, using these additional losses vastly improves the method’s performance. Especially, it results in a much lower EPE3D error, and, thus, in globally consistent tracking performance. Our approach uses a pose-conditioned multi-MLP formulation for the prediction of an implicit surface at every frame, which helps to refine local geometry details that are not captured properly by the graph deformation – Chamfer metric increases from 0.40 to 0.46 without pose conditioning. The benefits of pose conditioning over not using any pose codes can be observed also qualitatively in Fig. 5.7.

Runtime analysis. Our approach is trained independently (from scratch) for every character sequence, optimizing our neural deformation graph to best fit the observed sequence. For our unoptimized implementation, the optimization takes about 2 days with a single Nvidia GeForce 2080Ti. The dense deformation field between any two frames in the sequence is computed for (near) surface points by estimating skinning weights w.r.t. all nodes and interpolating the frame-to-frame motion of the nodes, which takes 0.02s for about $30k$ points. The mesh is extracted for every frame by sampling 128^3 grid sample points, predicting their SDF values and executing Marching Cubes on the resulting SDF grid, which takes about 3.70s per-frame.

Limitations and future work. Using our globally consistent Neural Deformation Graph, we show state-of-the-art tracking and reconstruction quality. Currently, our quality is limited by the input, i.e., a 64^3 SDF grid. Sparse 3D convolutions [144] could be applied to cope with higher resolutions. Our approach focuses on the tracking and reconstruction of the geometry, and not the texture. A texture on top of the tracked geometry could be estimated (similar to the color scheme that we show in the results figures) and additional losses based on this texture could be employed. In an over-crowded setting, with many occlusions (e.g. some parts are never observed), or in a single-view setting, the self-



Figure 5.6: Capture setup. We capture non-rigid motion using 4 Kinect Azure sensors, which are pre-calibrated and hardware-synced to guarantee spatial and temporal coherence of depth captures. We capture depth images at resolution of 1024×1024 , using the highest available frame-rate of 15 FPS.

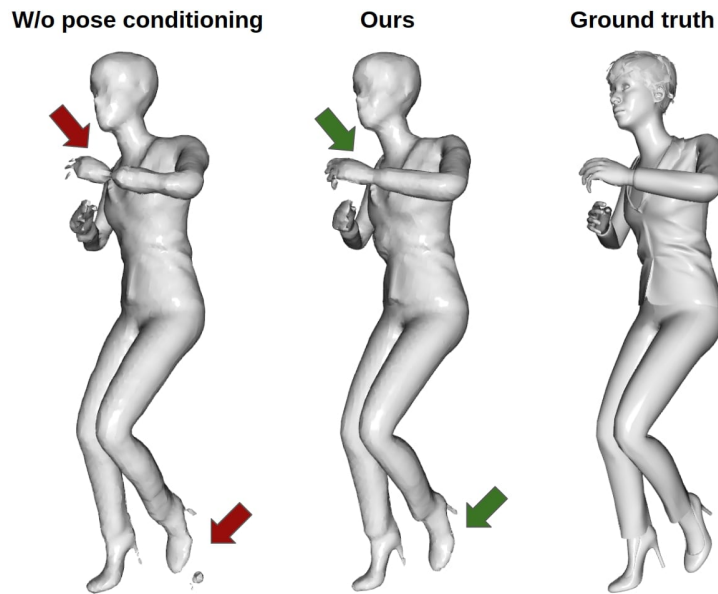


Figure 5.7: Influence of pose conditioning. We compare our implicit reconstruction with pose conditioning (*middle*) to without pose conditioning (*left*). Pose conditioning clearly improves reconstruction performance in regions of very strong deformation.

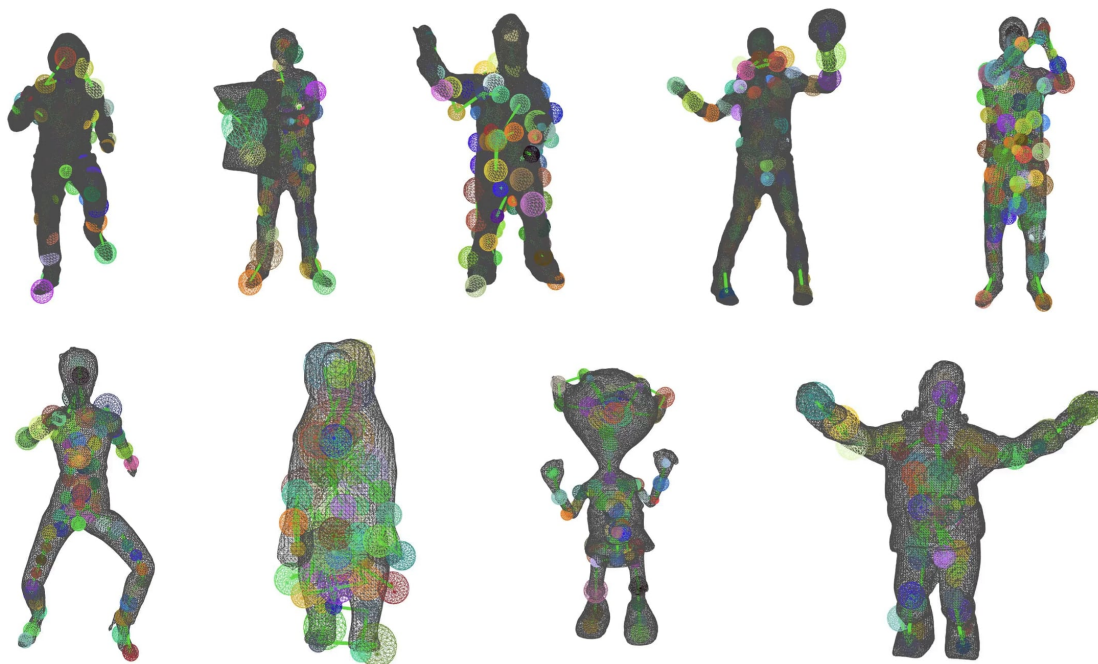


Figure 5.8: Neural deformation graph examples. We visualize a few examples of neural deformation graphs, optimized on real sequences (*top*) and synthetic data (*bottom*) using self-supervision.

supervised formulation might be under-constrained. A data-driven geometry prior could further improve the robustness of our approach. We believe that there is a potential for several high-impact follow-up works.

5.5 Conclusion

We introduced Neural Deformation Graph which allows to reconstruct and track non-rigidly deforming objects in a globally consistent fashion. It is enabled by a neural network that implicitly stores the deformation graph of the object. The network is trained with losses on global consistency, resulting in tracking and reconstruction quality that surpasses the state of the art by more than 60% w.r.t. the respective metrics. We believe that our global optimization of non-rigid motion will be a stepping stone to learn data-driven priors in the future.

6 Learning RGB Reconstruction

Abstract of paper. We introduce TransformerFusion, a transformer-based 3D scene reconstruction approach. From an input monocular RGB video, the video frames are processed by a transformer network that fuses the observations into a volumetric feature grid representing the scene; this feature grid is then decoded into an implicit 3D scene representation. Key to our approach is the transformer architecture that enables the network to learn to attend to the most relevant image frames for each 3D location in the scene, supervised only by the scene reconstruction task. Features are fused in a coarse-to-fine fashion, storing fine-level features only where needed, requiring lower memory storage and enabling fusion at interactive rates. The feature grid is then decoded to a higher-resolution scene reconstruction, using an MLP-based surface occupancy prediction from interpolated coarse-to-fine 3D features. Our approach results in an accurate surface reconstruction, outperforming state-of-the-art multi-view stereo depth estimation methods, fully-convolutional 3D reconstruction approaches, and approaches using LSTM- or GRU-based recurrent networks for video sequence fusion.

6.1 Introduction

Monocular 3D reconstruction is a core task in 3D computer vision, aiming to reconstruct a complete and accurate 3D geometry of an object or an environment from only 2D observations captured by an RGB camera. A geometric understanding is key to applications such as robotic or autonomous vehicle navigation or interaction, as well as model creation and scene editing for augmented and virtual reality. In addition, geometric scene reconstructions form the basis for 3D scene understanding, supporting tasks such as 3D object detection, semantic, and instance segmentation [145]–[152].

While state-of-the-art SLAM systems [153], [154] achieve robust and scale-accurate camera tracking leveraging both visual and inertial measurements, dense and complete 3D reconstruction of large-scale environments from monocular video remains a very challenging problem – particularly for interactive settings. Simultaneously, notable progress has been made on multi-view depth estimation, estimating depth from pairs of images by averaging features extracted from the images in a feature cost volume [57]–[59], [61], [155]. Unfortunately, averaging features across a full video sequence can lead to equal-weight treatment of each individual frame, despite some frames possibly containing less information in various regions (e.g., from motion blur, rolling shutter artifacts, very glancing or partial views of objects), making high-fidelity scene reconstruction challenging.

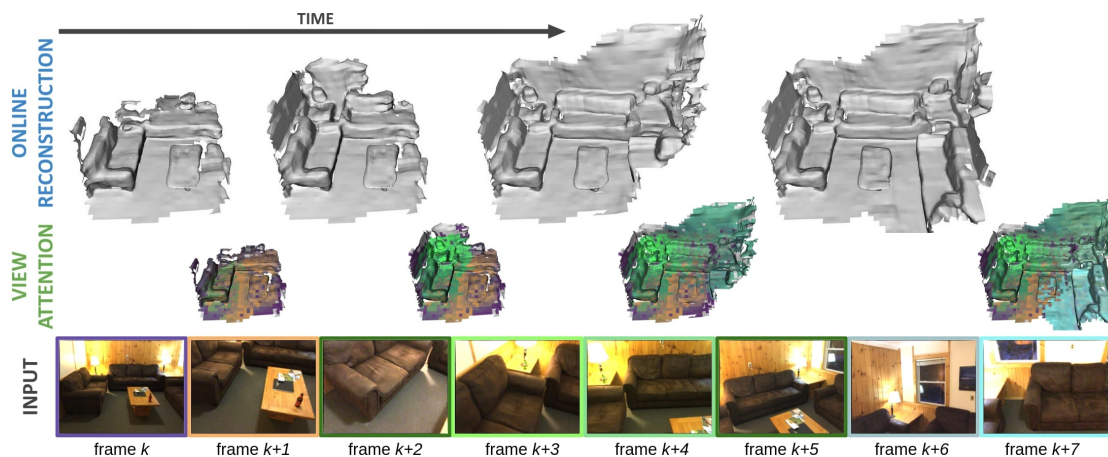


Figure 6.1: TransformerFusion. We present an online scene reconstruction method that takes a monocular RGB video as input. The features extracted from each observed image are fused incrementally with a transformer architecture. This fusion approach learns to attend to the most relevant image frames for each 3D location (see view attention color maps of the most relevant frame) achieving state-of-the-art reconstruction results.

Inspired by the recent advances in natural language processing (NLP) that leverage transformer-based models for sequence to sequence modelling [156]–[158], we propose a transformer-based method that fuses a sequence of RGB input frames into a 3D representation of a scene at interactive rates. Key to our approach is a learned feature fusion of the video frames using a transformer-based architecture, which learns to attend to the most informative image features to reconstruct a local 3D region of the scene. A new observed RGB frame is encoded into a 2D feature map, and unprojected into a 3D volume, where our transformer learns a fused 3D feature for each location in the 3D volume from the image view features. This enables extraction of the most informative view features for each location in the 3D scene. The 3D features are fused in coarse-to-fine fashion, providing both improved reconstruction performance as well as interactive runtime. These features are then decoded into high-resolution scene geometry with an MLP-based surface occupancy prediction.

In summary, our main contributions to achieve robust and accurate scene reconstructions are:

- Learned multi-view feature fusion in the temporal domain using a transformer network that attends to only the most informative features of the image views for reconstructing each location in a scene.
- A coarse-to-fine hierarchy of our transformer-based feature fusion that enables an online reconstruction approach running at interactive frame-rates.

6.2 Related Work

Multi-view depth estimation. Estimating depth from multi-view image observations has been long-studied in computer vision. COLMAP [1] introduced a patch matching based approach which achieves impressive accuracy and remains established as one of the most popular methods for multi-view stereo. While COLMAP offers robust depth estimation for distinctive features in images, the patch matching struggles to densely reconstruct areas without many distinctive color features, such as floor and walls. Recently, learning-based approaches that build data-driven priors from large-scale datasets have improved depth estimation in these challenging scenarios. Some proposed methods rely only on a 2D network with multiple images concatenated as input [57]. Several recent approaches instead build a shared 3D feature cost volume in reference camera space using feature averaging [58]–[61], [159]. These approaches estimate the reference frame’s depth within a local window of frames, but some also propagate information from previously estimated depth maps by using probabilistic filtering [159], a Gaussian process [58], or an LSTM bottleneck layer [61]. Such multi-view depth estimation approaches predict single-view depth maps, which must be fused together to construct a geometric 3D representation of the observed scene.

3D reconstruction from monocular RGB input. Multi-view depth estimation approaches can be combined with depth fusion approaches, such as volumetric fusion [37], to obtain a volumetric reconstruction of the observed scene. MonoFusion [160] is one of the first methods using depth estimate from a real-time variant of PatchMatch stereo [161]. However, fusing noisy depth estimates causes artifacts in the 3D reconstruction, which lead to the development of recent approaches that directly predict the 3D surface reconstruction instead of per-frame depth estimates. One of the first approaches to predict 3D surface occupancy from two input RGB images is SurfaceNet [162], which converts volumetrically averaged colors into 3D surface occupancies using a 3D convolutional network. Atlas [163] extends this approach to a multi-view setting, while also leveraging learned features instead of colors. Recently, NeuralRecon [55] proposed a real-time 3D reconstruction framework, adding GRU units distributed in 3D to fuse reconstructions from different local windows of frames. Our approach also fuses together learned features from RGB frame input in an online fashion, but our transformer-based multi-view feature fusion enables relying only on the most informative features from the observed frames for a particular spatial location in the reconstructed scene, producing more accurate 3D reconstructions.

Transformers in computer vision. The transformer architecture [156] has achieved profound impact in many computer vision tasks in addition to its natural language processing origins. For a detailed survey, we refer the reader to [164]. In computer vision, transformers have been leveraged successfully for tasks such as object detection [165], video classification [166], image classification [167], image generation [168], and human reconstruction [169]. In this work, we propose transformer-based feature fusion for 3D

scene reconstruction from a monocular video. Given a sequence of observed RGB frames, our approach learns to attend to the most informative features from each image to predict a dense occupancy field.

6.3 End-to-end 3D Reconstruction using Transformers

Given a set of N RGB images $I_i \in \mathbb{R}^{W \times H \times 3}$ of a scene with corresponding camera intrinsic parameters $\mathbf{K}_i \in \mathbb{R}^{3 \times 3}$ and extrinsic poses $\mathbf{P}_i \in \mathbb{R}^{4 \times 4}$, our method reconstructs the scene geometry by predicting occupancy values $o \in [0, 1]$ for every 3D point in the scene. Fig. 6.2 shows an overview of our approach. Each input image I_i is processed by a 2D convolutional encoder Θ , extracting coarse and fine image features (Φ_i^c and Φ_i^f , respectively):

$$\Theta : I_i \in \mathbb{R}^{W \times H \times 3} \mapsto (\Phi_i^c, \Phi_i^f)$$

From these 2D image features, we construct a 3D feature grid in world space. To this end, we regularly sample grid points in 3D at a coarse resolution of every $v_c = 30$ cm and a fine resolution of $v_f = 10$ cm. For these coarse and fine sample points, we query corresponding 2D features in all N images and predict fused coarse ψ^c and fine 3D features ψ^f using transformer networks [156]:

$$\begin{aligned} \mathcal{T}_c : (\Phi_1^c, \dots, \Phi_N^c) &\mapsto (\psi^c, w^c) \\ \mathcal{T}_f : (\Phi_1^f, \dots, \Phi_N^f) &\mapsto (\psi^f, w^f) \end{aligned}$$

Note that we also store the intermediate attention weights w^c and w^f of the first transformer layers for efficient view selection, which is explained in Sec. 6.3.4.

To further improve the features in the 3D spatial domain, we apply 3D convolutional networks \mathcal{C}_c and \mathcal{C}_f , at the coarse and fine level, respectively:

$$\begin{aligned} \mathcal{C}_c : \{\psi^c\}_{C \times C \times C} &\mapsto \{\tilde{\psi}^c\}_{C \times C \times C} \\ \mathcal{C}_f : \{(\tilde{\psi}^c, \psi^f)\}_{F \times F \times F} &\mapsto \{\tilde{\psi}^f\}_{F \times F \times F} \end{aligned}$$

Finally, to predict the scene geometry occupancy for a point $\mathbf{p} \in \mathbb{R}^3$, the coarse $\tilde{\psi}^c$ and fine features $\tilde{\psi}^f$ are trilinearly interpolated and a multi-layer perceptron \mathcal{S} maps these features to occupancies:

$$\mathcal{S} : (\tilde{\psi}^c, \tilde{\psi}^f) \mapsto o \in [0, 1]$$

This extraction of surface occupancies is inspired by convolutional occupancy networks [170] and IFNets [41]. From this occupancy field we extract a surface mesh with Marching cubes [18]. Note that in addition to surface occupancy, we also predict occupancy masks for near-surface locations at the coarse and fine levels. These masks are used for coarse-to-fine surface filtering (see Sec. 6.3.2), which improves reconstruction performance with a focus on the surface geometry prediction and enables interactive runtime.

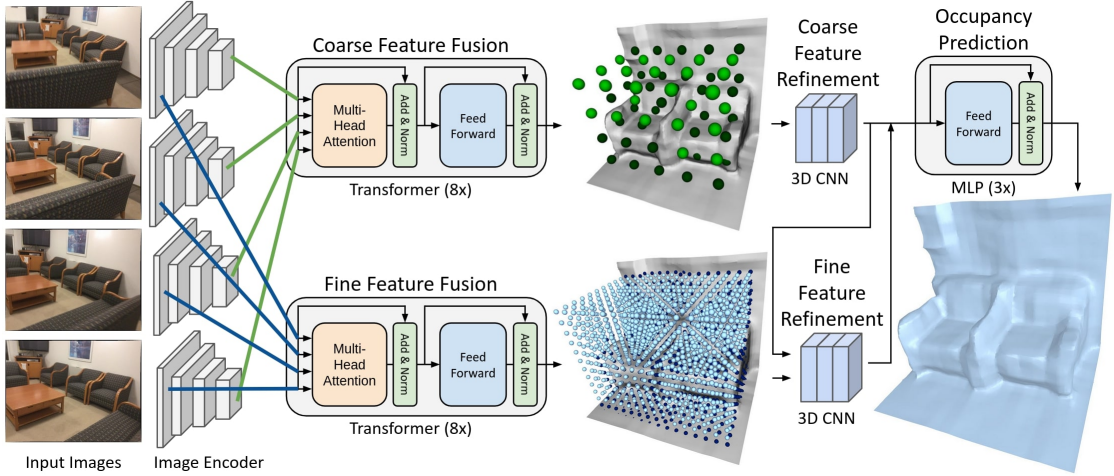


Figure 6.2: Method overview. Given multiple input images, we compute coarse and fine level features. Using a transformer architecture, we separately fuse these coarse and fine features in a voxel grid. To improve the spatial features, we use a refinement network for both the coarse and the fine features. From these feature grids, we extract an occupancy field using a lightweight MLP.

We train our approach in end-to-end fashion by supervising the surface occupancy predictions using the following loss:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_f + \mathcal{L}_o,$$

where \mathcal{L}_c and \mathcal{L}_f denote binary cross-entropy (BCE) losses on occupancy mask predictions for near-surface locations at the coarse and fine levels, respectively (see Sec. 6.3.2), and \mathcal{L}_o denotes a BCE loss for surface occupancy prediction (see Sec. 6.3.3).

6.3.1 Learning Temporal Feature Fusion via Transformers

For a spatial location $\mathbf{p} \in \mathbb{R}^3$ in the scene reconstruction, we learn to fuse coarse ψ^c and fine level features ψ^f from the N coarse and fine feature images (Φ_i^c and Φ_i^f , respectively), which are extracted by the 2D encoder Θ . Specifically, we train two instances of a transformer model, one for fusing coarse-level features ψ^c and one for fusing fine-level features ψ^f . Both transformers \mathcal{T}_c and \mathcal{T}_f share the same architecture. Thus, for simplicity, we omit the coarse and fine notation in the following.

Our transformer model \mathcal{T} is independently applied to each sample point in world space. For a point \mathbf{p} , the transformer network takes a series of 2D features ϕ_i as input that are bilinearly sampled from the feature maps Φ_i at the corresponding projective image location. The projective image location is computed via a full-perspective projection $\Pi_i(\mathbf{p}) = \pi(\mathbf{K}_i(\mathbf{R}_i\mathbf{p} + \mathbf{t}_i))$, assuming known camera intrinsics \mathbf{K}_i and extrinsics $\mathbf{P}_i = (\mathbf{R}_i, \mathbf{t}_i)$. To inform the transformer about invalid features (i.e., a sample point is projected outside an image), we also provide the pixel validity $v_i \in \{0, 1\}$ as input. In

addition to these 2D features ϕ_i , we concatenate the projected depth $d_i = (\mathbf{R}_i \mathbf{p} + \mathbf{t}_i)_z$, and the viewing ray $\mathbf{r}_i = (\mathbf{p} - \mathbf{c}_i) / \|\mathbf{p} - \mathbf{c}_i\|_2$ to the input ($\mathbf{c}_i \in \mathbb{R}^3$ denoting the camera center of view i). These input features are converted to an embedding vector $\theta_i \in \mathbb{R}^D$ using a linear layer $\theta_i = \mathbf{FCN}(\phi_i, d_i, v_i, \mathbf{r}_i)$, before feeding it into the transformer network that then predicts a fused feature $\psi \in \mathbb{R}^D$:

$$\mathcal{T} : (\theta_1, \dots, \theta_N) \mapsto (\psi, w)$$

As described above, w denotes the attention values of the initial attention layer, which are used for view selection to speed-up fusion (see Sec. 6.3.4).

Transformer architecture. We followed [167] when designing the transformer architecture \mathcal{T} . It consists of 8 modules of feed-forward and attention layers, using multi-head attention with 4 attention heads and embedding dimension $D = 256$. Feed-forward layers process the temporal inputs independently, and contain ReLU activation, linear layers with residual connection, and layer norm. The model returns both fused feature $\psi \in \mathbb{R}^D$ and attention weights $w \in \mathbb{R}^N$ over all temporal inputs from the initial attention layer that are later used for selecting which views to maintain over longer sequences of input image views.

6.3.2 Spatial Feature Refinement

While the transformer network fuses 2D observations in the temporal domain, we additionally imbue explicit spatial reasoning by applying a 3D CNN to spatially refine the fused features $\{\psi^c\}_{C \times C \times C}$ and $\{\psi^f\}_{F \times F \times F}$ that are computed by the transformers \mathcal{T}_c and \mathcal{T}_f on the coarse and fine grid, respectively. The coarse features $\{\psi^c\}_{C \times C \times C}$ are refined by a 3D CNN \mathcal{C}_c consisting of 3 residual blocks that maintain the same spatial resolution and produce refined features $\{\tilde{\psi}^c\}_{C \times C \times C}$. These features are upsampled to a fine grid resolution using nearest-neighbor upsampling, and concatenated with fused features at fine level $\{\psi^f\}_{F \times F \times F}$. A fine-level 3D CNN \mathcal{C}_f is then applied to the concatenated features, resulting in refined fine features $\{\tilde{\psi}^f\}_{F \times F \times F}$. Both, coarse $\tilde{\psi}^c$ and fine features $\tilde{\psi}^f$ are used for surface occupancy prediction.

Coarse-to-fine surface filtering. The refined features are also used to predict occupancy masks for near-surface locations at both coarse and fine levels, thus, filtering out free-space regions and sparsifying the volume, such that the higher-resolution and computationally expensive fine-scale surface extraction is performed only in regions close to the surface. To achieve this, additional 3D CNN layers \mathcal{M}_c and \mathcal{M}_f are applied to the refined features, outputting a near-surface mask $m^c, m^f \in [0, 1]$ for every grid point:

$$\begin{aligned} \mathcal{M}_c &: \{\tilde{\psi}^c\}_{C \times C \times C} \mapsto \{m^c\}_{C \times C \times C} \\ \mathcal{M}_f &: \{\tilde{\psi}^f\}_{F \times F \times F} \mapsto \{m^f\}_{F \times F \times F} \end{aligned}$$

Only spatial regions where both m^c and m^f are larger than 0.5, i.e., close to the surface, are processed further to compute the final surface reconstruction; other regions are determined to be free space. This improves the overall reconstruction performance by focusing the capacity of the surface prediction network to close-to-the-surface regions and enables a significant runtime speed-up.

Intermediate supervision of near-surface masks m^c and m^f is employed using masks m_{gt}^c and m_{gt}^f generated from the ground truth scene reconstruction, denoting the grid point as near-surface if there exists ground truth surface in the radius of v_c or v_f from the point. Binary cross entropy losses $\mathcal{L}_c = \text{BCE}(m^c, m_{\text{gt}}^c)$ and $\mathcal{L}_f = \text{BCE}(m^f, m_{\text{gt}}^f)$ are applied.

6.3.3 Surface Occupancy Prediction

The final surface reconstruction is predicted by decoding the coarse and fine feature grids to occupancy values $o \in [0, 1]$, with values $o \geq 0.5$ representing occupied points and values $o < 0.5$ representing free-space points. For a point $\mathbf{p} \in \mathbb{R}^3$, we compute its feature representation by trilinearly interpolating coarse and fine grid features:

$$\begin{aligned}\psi_{\mathbf{p}}^c &= \text{Trilinear}(\mathbf{p}, \{\tilde{\psi}^c\}_{C \times C \times C}) \\ \psi_{\mathbf{p}}^f &= \text{Trilinear}(\mathbf{p}, \{\tilde{\psi}^f\}_{F \times F \times F})\end{aligned}$$

We concatenate the interpolated features and predict the point’s occupancy as $o = \mathcal{S}(\psi_{\mathbf{p}}^c, \psi_{\mathbf{p}}^f)$, where \mathcal{S} is a multi-layer perceptron (MLP) with 3 modules of feed-forward layers, containing ReLU activation, linear layer with residual connection, and layer norm.

Surface occupancy supervision. We train on $1.5 \times 1.5 \times 1.5$ m volumetric chunks of scenes for training efficiency. To supervise the surface occupancy loss, 1k points are sampled inside the chunk, with 80% of samples drawn from a truncation region at most 10 cm from the surface, and 20% sampled uniformly inside the chunk. Ground truth occupancy values o_{gt} are computed using the ScanNet RGB-D reconstructions [17]. For uniform samples it is straightforward to generate unoccupied point samples by sampling points in free space in front of the visible surface, but it is unknown whether a point sample is occupied when it lies behind seen surfaces. In order to prevent artifacts behind walls, we follow the data processing applied in [163] and additionally label point samples as occupied, if they are sampled in areas where an entire vertical column of voxels is occluded in the scene. A binary cross entropy loss $\mathcal{L}_o = \text{BCE}(o, o_{\text{gt}})$ is then applied to the occupancy predictions o .

6.3.4 View Selection for Online Scene Reconstruction

We aim to consider all N frames as input to our transformer for each 3D location in a scene; however, this becomes extremely computationally expensive with long videos or large-scale scenes, which prohibits online scene reconstruction. Instead, we proceed with the reconstruction incrementally, processing every video frame one-by-one, while

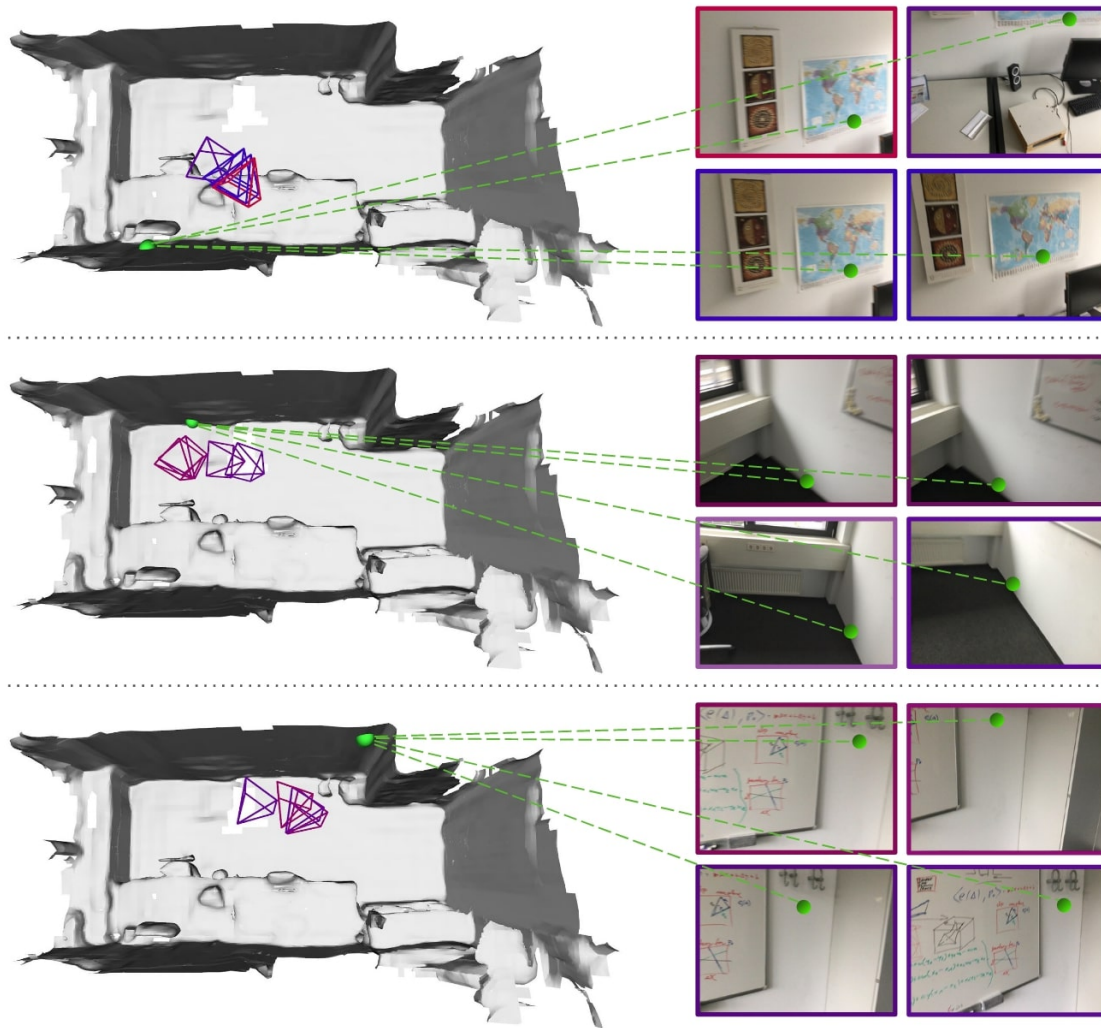


Figure 6.3: View attention visualization. Visualization of selected camera views with self-supervised attention weights (lower weights are visualized as *blue* and higher as *red*) for specific 3D locations (highlighted as *green*) in the scene.

keeping only a small number $K = 16$ of measurements for every 3D point. We visualize this online approach in Fig. 6.1. In Fig. 6.3, we show a 3D reconstruction of a scene from the test set of the ScanNet dataset [17], and render the camera views that are selected for a specific 3D location (green point) with corresponding attention weights (color temperature corresponding to the weight). On the right, we show the corresponding input images. There are different colors (and, thus, attention weights) for similar views. We observed that the attention head in the transformer architecture tends to sparsify the views, assigning a high weight to a single view, and low weights for the rest, as other similar views tend to provide only redundant information. Such behavior is well-suited for the task of reconstruction, where multiple different observations are more useful, especially ones observed under different viewpoints and camera translations. This is achieved by using multiple attention heads (in our architecture we use 8 heads), each specializing for a certain view type, and each picking only a single view representative. This leads to elimination of very similar views that are redundant, and at the same time encourages high weights for different views that are more useful – additionally, this enables the attention weights to be very effective for online view selection. It is also a notable difference to existing works [55], [163], where all views are treated the same (by averaging over view features).

During training, for efficiency, we use only K_t random images for each training volume. At test time, we leverage the attention weights w^c and w^f of the initial transformer layers to determine which views to keep in the set of K measurements. Specifically, for a new RGB frame, we extract its 2D features, and run feature fusion for every coarse and fine grid point inside the camera frustum. This returns the fused feature and also the attention weights over all currently accumulated input measurements. Whenever the maximum number of K measurements is reached, a selection is made by dropping out a measurement with lowest attention weight before adding new measurements in the latest frame. This guarantees a low number of input measurements, speeding up fusion processing times considerably. Furthermore, by using coarse-to-fine filtering, described in Sec. 6.3.2, we can further accelerate fusion by only considering higher resolution points in the area near the estimated surface. Together with incremental processing that results in high performance benefits, our approach performs per-frame feature fusion at about 7 FPS despite an unoptimized implementation.

6.3.5 Training Scheme

Our approach has been implemented using the PyTorch library [22]. To train our model we use ScanNet dataset [17], an RGB-D dataset of indoor apartments. We follow the established train-val-test split. For training, we randomly sample $1.5 \times 1.5 \times 1.5$ m volume chunks of the train scenes, sampling less chunks in free space and more samples in areas with non-structural objects, i.e. not only consisting of floor or walls. This results in ≈ 165 k training chunks. For each chunk, we randomly sample $K_t = 8$ RGB images among all frames that include the chunk in their camera frustums.

The 2D convolutional encoder Θ for image feature extraction is implemented as a ResNet-18 [171] network, pre-trained on ImageNet [172]. During training, a batch size

of 4 chunks is used with an Adam [12] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and weight regularization of 10^{-4} . We use a learning rate of 10^{-4} with 5k warm-up steps at initialization, and square root learning rate decay afterwards. When computing the losses of coarse and fine surface filtering predictions, a higher weight of 2.0 is applied to near-surface voxels, to increase recall and improve overall robustness. Training takes about 30 hours using an Intel Xeon 6242R Processor and an Nvidia RTX 3090 GPU.

6.4 Experiments

Metrics. To evaluate our monocular scene reconstruction, we use several measures of reconstruction performance. We evaluate geometric accuracy and completion, with accuracy measuring the average point-to-point error from predicted to ground truth vertices, completion measuring the error in the opposite direction, and Chamfer as the average of accuracy and completion (in cm). To account for possibly different mesh resolutions among methods, we uniformly sample 200k points over mesh faces of every reconstructed mesh. Additionally, we threshold these point-to-point errors and compute precision and recall by computing the ratio of point-to-point matches within distance ≤ 5 cm. Since it is easy to maximize either precision (by predicting only a few but accurate points) or recall (by over-completing reconstructions with noisy surface), we found the most reliable metric to be F-score, determined by both precision and recall.

Our ground truth reconstructions are obtained by automated 3D reconstruction [77] from RGB-D videos of real-world environments and, thus, they are often incomplete due to unobserved and occluded regions in the scene. To avoid penalizing methods for reconstructing a more complete scene w.r.t. the available ground truth, we apply an additional occlusion mask at evaluation.

As most state of the art, particularly for depth estimation, rely on a pre-sampled set of keyframes (based on sufficient translation or rotation difference between camera poses), we evaluate all approaches based on sequences of sampled keyframes, using the keyframe selection of [61].

6.4.1 Comparison with State of the Art

In Tab. 6.1, we compare our approach with state-of-the-art methods. All methods are trained on the ScanNet dataset [17], using the official train/val/test split. We use the pre-trained models provided by the authors for MVDepthNet [57], GPMVS [58] and DPSNet [59] which are fine-tuned on ScanNet. For baselines that predict depth in a reference camera frame instead of directly reconstructing 3D surface, a volumetric fusion method [37] is used to fuse different depth maps into a 3D truncated signed distance field. The single-view depth prediction method RevisitingSI [173] suffers from the more challenging task formulation without the use of multiple views, leading to noisier depth predictions and inconsistencies between frames. Multi-view depth estimation methods leverage the additional view information for improved performance, with the LSTM-based approach of DeepVideoMVS [61] achieving the best performance among these

Method	Acc ↓	Comp ↓	Cham ↓	Prec ↑	Rec ↑	F-sc ↑
RevisitingSI [173]	14.29	16.19	15.24	0.346	0.293	0.314
MVDepthNet [57]	12.94	8.34	10.64	0.443	0.487	0.460
GPMVS [58]	12.90	8.02	10.46	0.453	0.510	0.477
ESTDepth [60]	12.71	7.54	10.12	0.456	0.542	0.491
DPSNet [59]	11.94	7.58	9.77	0.474	0.519	0.492
DELTAS [155]	11.95	7.46	9.71	0.478	0.533	0.501
DeepVideoMVS [61]	10.68	6.90	8.79	0.541	0.592	0.563
COLMAP [1]	10.22	11.88	11.05	0.509	0.474	0.489
NeuralRecon [55]	5.09	9.13	7.11	0.630	0.612	0.619
Atlas [163]	7.16	7.61	7.38	0.675	0.605	0.636
Ours: w/o TRSF, avg	7.23	9.74	8.48	0.635	0.501	0.557
Ours: w/o TRSF, weight	6.11	11.12	8.61	0.686	0.512	0.583
Ours: w/o TRSF, conv	6.56	9.84	8.20	0.661	0.524	0.582
Ours: w/o spatial ref.	10.46	16.91	13.68	0.479	0.295	0.361
Ours: w/o C2F filter	6.57	7.69	7.13	0.678	0.592	0.631
Ours: w/o proj. depth	8.06	10.02	9.04	0.594	0.475	0.525
Ours: w/o viewing ray	5.71	8.59	7.15	0.706	0.559	0.621
Ours: 30 cm voxel size	7.92	17.33	12.63	0.491	0.258	0.335
Ours: 15 cm voxel size	5.79	9.62	7.71	0.686	0.520	0.589
Ours: 4 images, RND	8.01	10.28	9.15	0.587	0.445	0.502
Ours: 4 images	6.80	8.40	7.60	0.661	0.524	0.581
Ours: 8 images, RND	6.74	8.55	7.64	0.665	0.544	0.596
Ours: 8 images	6.17	7.69	6.93	0.704	0.584	0.636
Ours: 16 images, RND	5.80	8.56	7.18	0.711	0.584	0.638
Ours	5.52	8.27	6.89	0.728	0.600	0.655

Table 6.1: Quantitative comparison. We compare with state-of-the-art methods and perform ablations on test set of Scannet dataset [17].

Part II. Non-rigid Reconstruction using Data-driven Priors

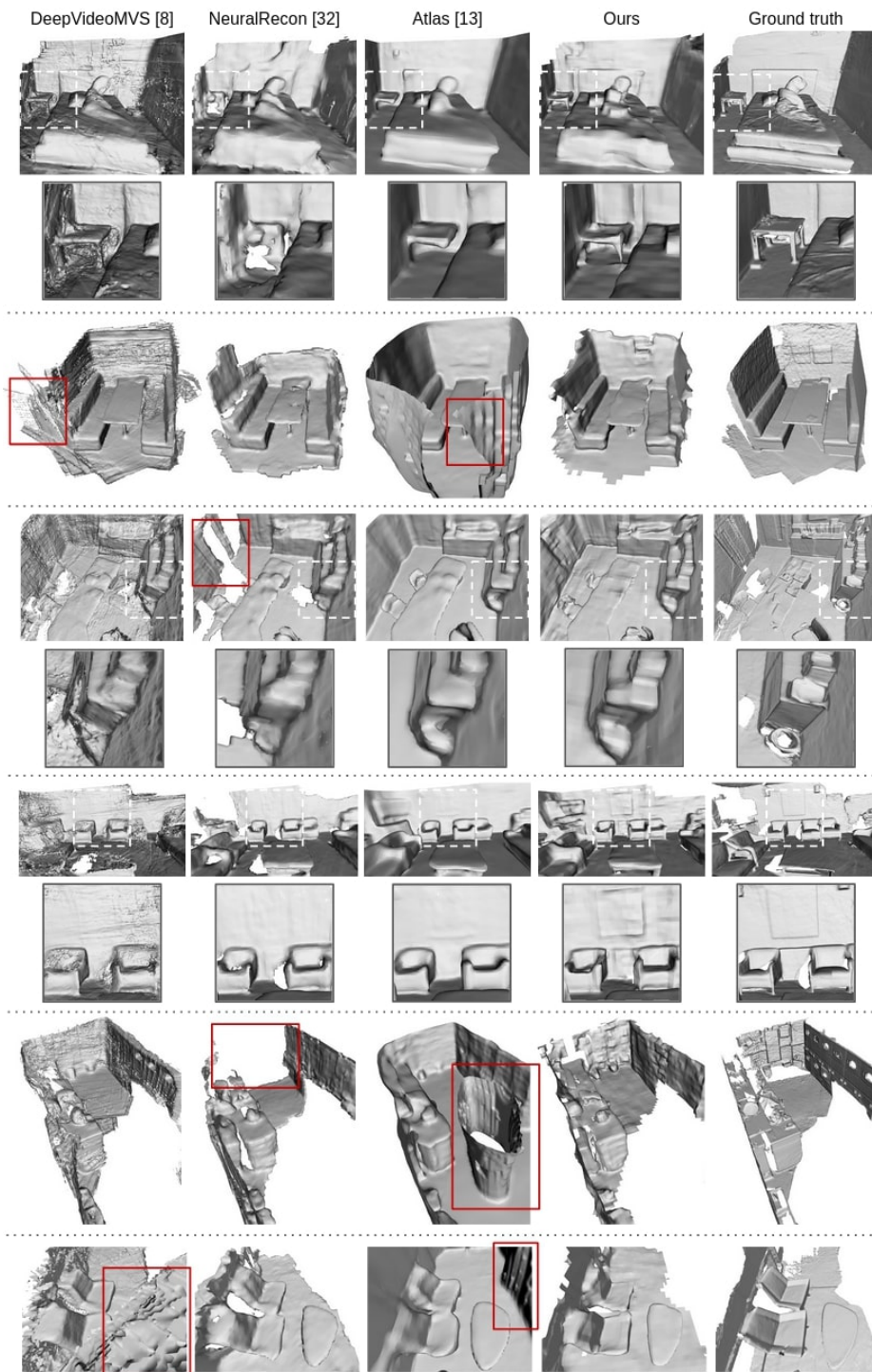


Figure 6.4: Qualitative comparison to state of the art. We compare scene reconstructions on test set of ScanNet dataset [17]; note that only RGB input is used by each method while the ground truth is reconstructed using the input depth.

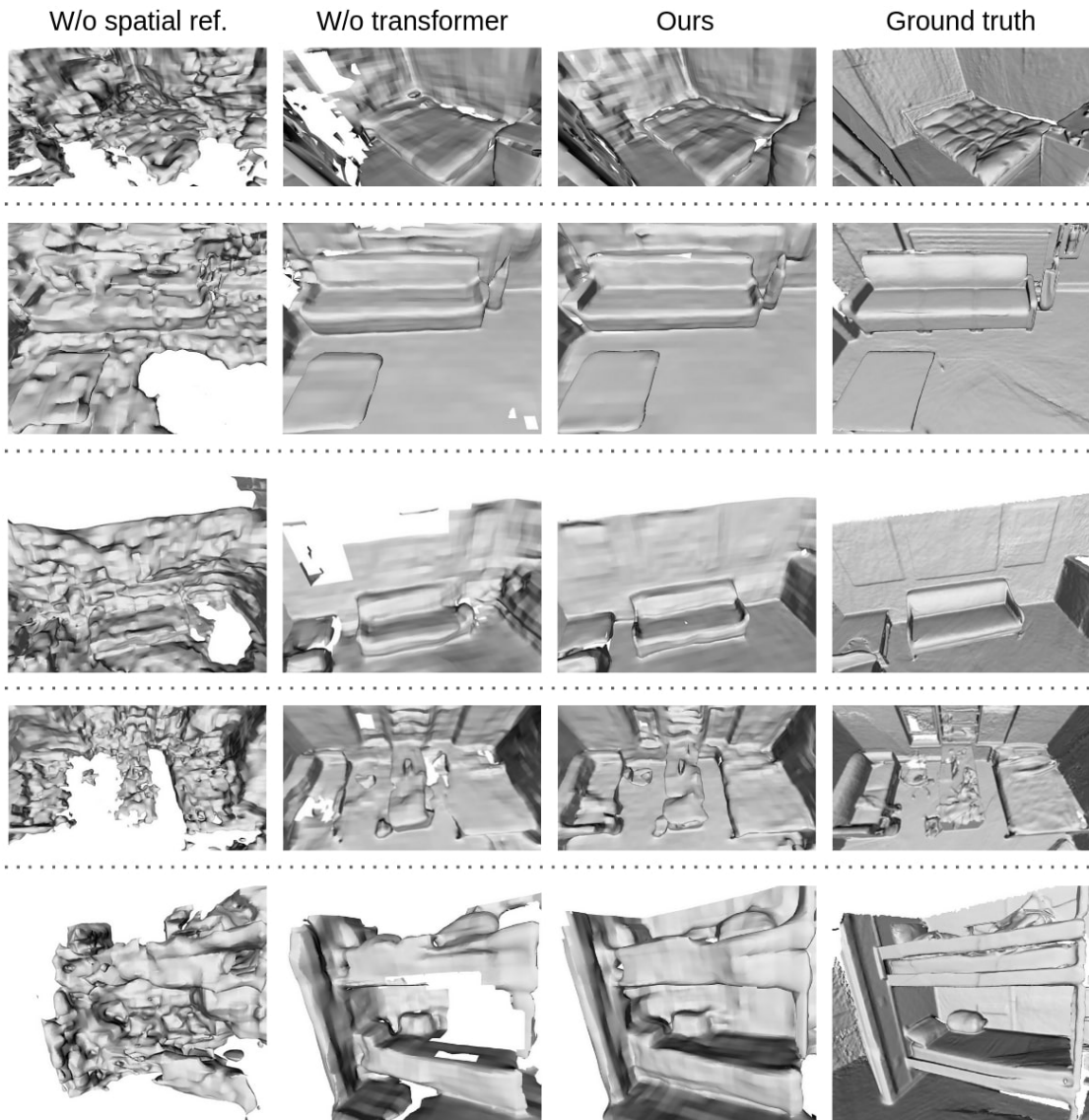


Figure 6.5: Qualitative comparison of ablations of our approach. We perform ablations on test set of ScanNet dataset [17]; note that only RGB input is used by each method while the ground truth is reconstructed using the input depth.

approaches. Reconstruction quality further improves with methods that directly predict the 3D surface geometry, such as NeuralRecon [55] and Atlas [163]. Our transformer-based feature fusion approach enables more robust reconstruction and outperforms all existing methods in both Chamfer distance and F-score. The performance improvement can also be clearly seen in the qualitative comparisons in Fig. 6.4. We provide some additional qualitative reconstruction results of our approach in Fig. 6.7.

6.4.2 Ablations

To demonstrate the effectiveness of our design choices, we conducted a quantitative ablation study which is shown in Tab. 6.1 and discussed in the following. We additionally provide qualitative ablations in Fig. 6.5.

What is the impact of learning to fuse features from different views with transformers? We evaluate the effect of our learned feature fusion by replacing the transformer blocks with a multi-layer perceptron (MLP) that processes input image observations independently. The per-view outputs of this MLP are fused using an average (*w/o TRSF, avg*) or using a weighted average with weights predicted by the MLP (*w/o TRSF, weight*). Additionally, we implemented convolutional feature fusion, using a 1-dimensional CNN that processes features in temporal domain and predicts fused features (*w/o TRSF, conv*). We find that our transformer-based view fusion effectively learns to attend to the most informative views for a specific location, resulting in significantly improved performance over these feature fusion alternatives.

Does spatial feature refinement help reconstruction performance? Spatial feature refinement is indeed very important for reconstruction quality. It enables the model to aggregate feature information in spatial domain and produce more spatially consistent and complete reconstructions, without it (*w/o spatial ref.*) the geometry completion (and recall metric) are considerably worse.

How important is coarse-to-fine filtering? Predicting the coarse and fine near-surface masks provides an additional performance improvement compared to the model without it (*w/o C2F filter*), as it allows more focus on surface geometry. Furthermore, this enables a speed-up of the fusion runtime by a factor of approximately 3.5, resulting in processing times of 7 FPS (instead of 2 FPS).

Are additional inputs to the transformer networks needed? Existing reconstruction approaches [55], [163] aggregate 2D features using a simple average operation. In comparison, our approach uses a transformer to learn the feature fusion. That makes it possible to use additional inputs that don't support a straight-forward average operation, but could be very informative for the task of multi-view surface reconstruction, such as projected depth and viewing ray. In Tab. 6.1 we conducted an additional quantitative ablation study w.r.t. the input to the transformer networks. Both the projected

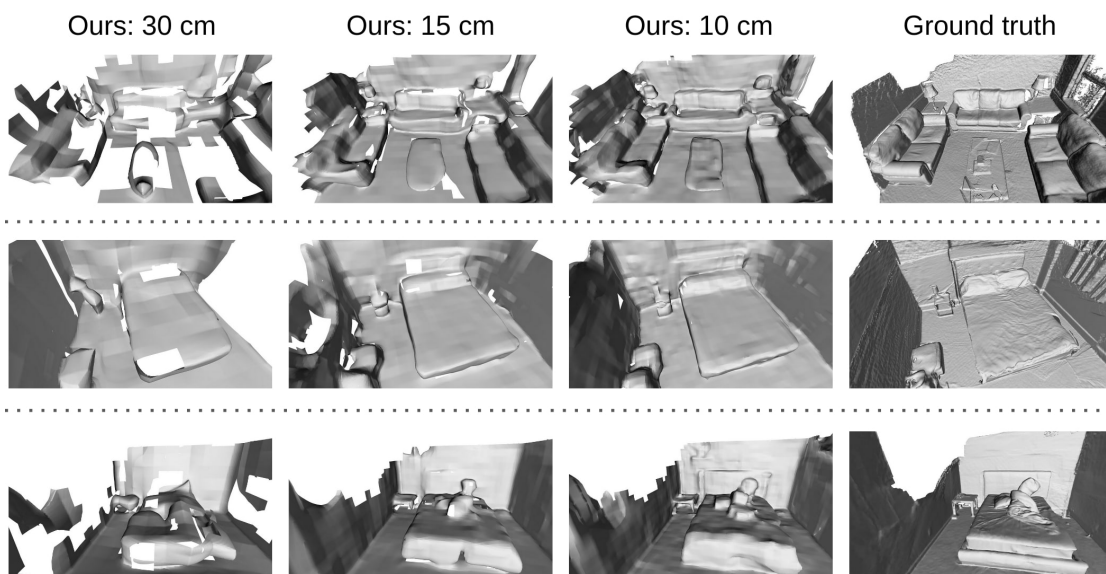


Figure 6.6: Qualitative comparison of ablations on feature voxel size. We replaced the original voxel size of 10 cm at the fine grid level with 30 cm and 15 cm.

depth as well as the view ray help the transformer to better fuse the features for the task of 3D reconstruction.

How does voxel size of feature grids influence reconstruction performance? We compared the reconstruction performance when using different voxel sizes for the feature grid. We only varied fine feature grid resolution, voxel size of coarse grid was always 30 cm. More specifically, we replaced the voxel size of 10 cm at the fine grid level with 30 cm and 15 cm. In both cases, the performance decreased considerably; i.e., the higher the resolution, the better the results. That is reflected also in qualitative comparison in Fig. 6.6.

How many views should be used for feature fusion? In our experiments, we use a limited number of $K = 16$ frame observations to inform the feature for every 3D grid location. We find that these views all contribute, with performance degrading somewhat with sparser sets of observations ($K = 8$ or $K = 4$). The number of frames is limited because of execution time and memory consumption for bigger scenes.

How effective is frame selection using attention weights? The K frames for each 3D grid feature are selected based on the computed attention weights and are updated during scanning. To evaluate this frame selection, we compare against a frame selection scheme that randomly selects frames that observe the 3D location (*RND*), which results in a noticeable drop in performance for both Chamfer and F-score. The performance difference is even larger when using less views for fusion ($K = 8$ or $K = 4$), where view

Task	Duration
Image loading / feature extraction	21.50ms
Coarse feature fusion	11.81ms
Near-surface mask prediction	24.18ms
Fine feature fusion	73.03ms
Total	130.52ms

Table 6.2: Feature fusion runtime. We analyse runtime of the per-frame feature fusion.

Task	Duration
Coarse feature refinement	28.56ms
Fine feature refinement	140.79ms
MLP (occupancy prediction)	25.21ms
Marching cubes [18]	48.74ms
Total	243.29ms

Table 6.3: Mesh extraction runtime. We analyse runtime of the per-chunk mesh extraction.

selection becomes even more important. In Fig. 6.1, we visualize the most important view for locations in the scene, selected by the highest attention weight. Relatively smooth transitions between selected views among neighboring 3D locations suggest that view selection is spatially consistent. To illustrate the frame selection, we also visualize all selected frames with corresponding attention weights for specific 3D locations in Fig. 6.3.

6.4.3 Runtime Analysis

In this section we provide further details about the runtime of our approach. We benchmarked our approach using an Intel Xeon 6242R Processor and an Nvidia RTX 3090 GPU. For every new frame coarse-to-fine image features need to be extracted, and fused into global coarse and fine feature volumes. In Tab. 6.2, we report execution times of the different feature fusion steps. Coarse features are fused into the entire camera frustum, containing all coarse voxels that fall into valid depth range [0.3m, 5m]. Fine feature on the other hand are fused only in near-surface areas, as predicted by coarse filtering. The execution times are averaged over a representative video sequence of the ScanNet dataset.

The surface reconstruction doesn’t need to be extracted for every frame. It can either be done at the end, when all image features are already fused into the feature volume, or incrementally every couple of frames, on a per-chunk basis, if interactive feedback is desired. In Tab. 6.3, we report execution times for a chunk of size $1.5 \times 1.5 \times 1.5$ m. Both, coarse and fine features are spatially refined using a 3D CNN and surface occupancy is computed using the occupancy MLP at a voxel resolution of 2 cm, but only for near-surface voxels, as predicted by coarse and fine near-surface masks. Finally, the mesh is extracted using Marching cubes [18].

Note that our implementation uses high-level PyTorch routines, as well as CPU code (e.g., for Marching Cubes) and, thus, the implementation is not optimized for runtime. A more optimized implementation can be achieved via customized CUDA code. Another interesting avenue towards higher frame rates is the use of sparse 3D convolutions instead of dense 3D convolutions. Feature fusion timings are reported for our default reconstruction setting, when we store $K = 16$ views for every feature grid voxel. The feature fusion execution can be further accelerated by using less views. The frames per

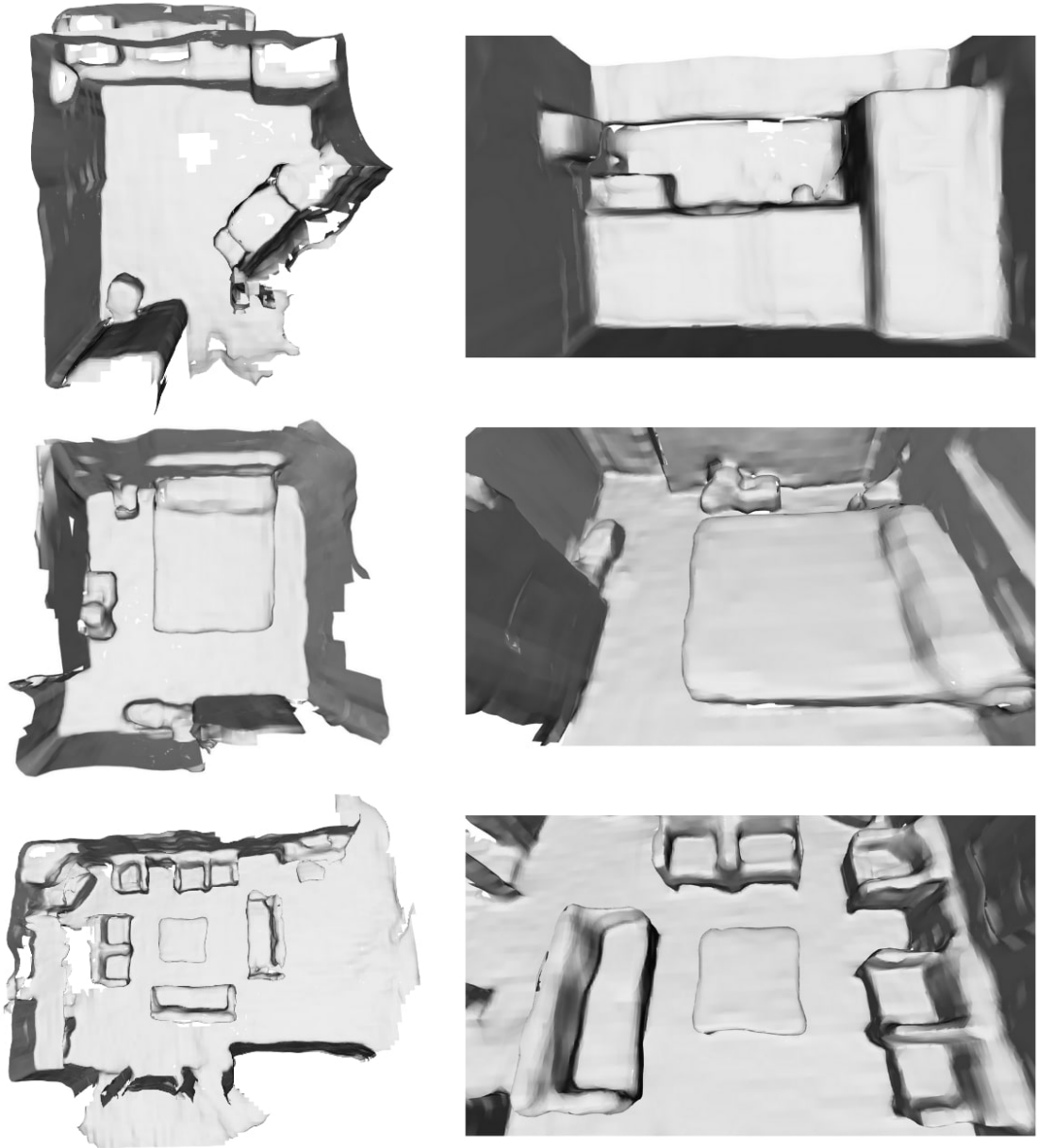


Figure 6.7: Qualitative reconstruction results. We visualize reconstructions of representative scenes from the test-set of the ScanNet dataset [17].

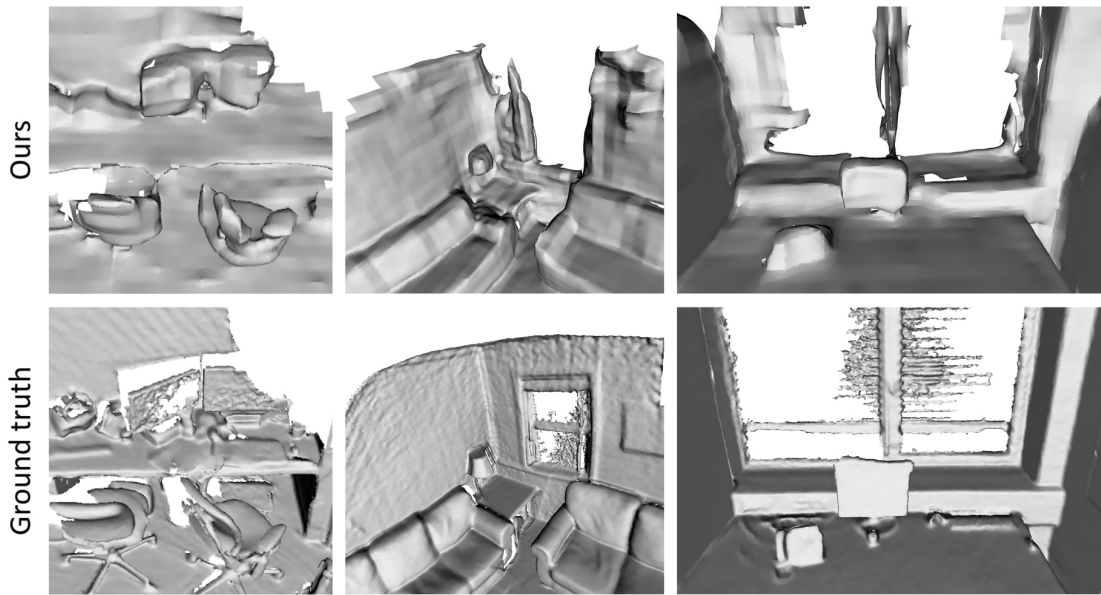


Figure 6.8: Limitations. Our approach can lack detail at partially observed and occluded objects, and result in inaccurate reconstruction of transparent surfaces, such as glass windows.

second (FPS) increase from 7.66 FPS for 16 views to 10.17 FPS for 8 and to 12.28 FPS for 4 views.

6.4.4 Limitations

Under severe occlusions and partial observation of the scene, our method can struggle to reconstruct details of certain objects, such as chair legs, monitor stands, or books on the shelves. Furthermore, transparent objects, such as glass windows without frames, are often inaccurately reconstructed as empty space. We show qualitative examples of these failure cases in Fig. 6.8. These challenging scenarios are often not properly reconstructed even when using ground truth RGB-D data, and we believe that using self-supervised losses [174] for monocular scene reconstruction could be an interesting future research direction. Additionally, higher resolution geometric fidelity could potentially be achieved by sparse operations in 3D or learning local geometric priors on detailed synthetic data [175].

6.5 Conclusion

We introduced TransformerFusion for monocular 3D scene reconstruction, leveraging a new transformer-based approach for online feature fusion from RGB input views. A coarse-to-fine formulation of our transformer-based feature fusion improves the effective reconstruction performance as well as the runtime. Our feature fusion learns to ex-

exploit the most informative image view features for geometric reconstruction, achieving state-of-the-art reconstruction performance. We believe that our interactive scanning approach provides exciting avenues for future research, and enables new possibilities in learning multi-view perception and 3D scene understanding.

Part III

Conclusion & Outlook

7 Conclusion

Non-rigid 3D reconstruction is one of the core problems in computer vision, focusing on reconstruction and motion tracking of deformable objects from videos. It has many exciting applications in VR/AR, autonomous driving, robotics, etc., but the problem is very underconstrained and existing algorithms often lack robustness in everyday situations. This dissertation explores *data-driven* non-rigid reconstruction, using learned priors to make non-rigid tracking and reconstruction more accurate and robust. We significantly improved various aspects of the existing approaches using data-driven components, and we summarize the main contributions in the following.

Learning Non-rigid Reconstruction. In Chapter 3 we introduced a large-scale dataset of 400 non-rigid RGB-D videos, with over 390,000 frames and dense correspondences between frame pairs annotated in a semi-supervised manner. Additionally, we proposed a heatmap-based neural network for non-rigid correspondence prediction, trained on our captured data and incorporated into a non-rigid reconstruction framework. Our data-driven method significantly outperforms existing non-rigid reconstruction approaches that use hand-crafted descriptors, both qualitatively and quantitatively, evaluated on the first non-rigid reconstruction benchmark that we established to objectively compare different approaches.

Neural Non-rigid Tracking. In Chapter 4 we proposed an end-to-end differentiable non-rigid tracking approach, which learns dense non-rigid correspondences best-suited for the task of non-rigid tracking. The differentiable optimization formulation enabled us to learn importance weights for outlier rejection in a purely self-supervised manner. When added to the non-rigid reconstruction pipeline, the learned non-rigid tracker not only improves the tracking performance, but also leads to better shape reconstructions. Furthermore, the dense correspondence prediction in a single forward pass is significantly faster compared to the independently predicted sparse correspondences from our previous approach.

Neural Deformation Graphs. In Chapter 5 we proposed to represent the deformation graph and shape implicitly with a neural network, and introduced a novel network-based optimization that tracks and reconstructs non-rigid objects in a globally-consistent fashion. Novel losses encouraging per-frame viewpoint consistency and inter-frame graph and surface consistency result in high-quality reconstruction and tracking that quantitatively surpass the state of the art for more than 60%. The shape is modeled by a deformable multi-MLP representation anchored on the neural deformation graph, with

different shape parts reconstructed independently, eliminating the need for a fixed canonical space that cannot handle topology changes.

Learning RGB Reconstruction. In Chapter 6 we introduced TransformerFusion, a transformer-based 3D scene reconstruction approach that takes an RGB video as input and outputs 3D geometry. Assuming camera motion is estimated, the method learns the fusion of per-frame extracted 2D pixel features in the temporal domain using a transformer network. For each location in the scene, the transformer attends to only the most informative views, enabling efficient and self-supervised view selection. Combined with a coarse-to-fine feature hierarchy, this not only provides accurate reconstructions in a very underconstrained RGB setting, surpassing state of the art, but also enables online reconstruction running at interactive frame-rates.

8 Limitations and Future Work

When describing the approaches proposed in this dissertation, we already listed the methods’ limitations, visualized failure cases and noted down opportunities for improvement. However, there are many interesting areas in data-driven non-rigid reconstruction that we haven’t explored yet, and would make great future research directions. After all, when it comes to learning priors for general deformable objects, not just human bodies or faces, we believe it’s just a start of a very promising research field. We showcased that using data-driven priors for different aspects of non-rigid reconstruction is indeed very valuable and significantly improves both the non-rigid tracking and shape reconstruction, and we hope to encourage many other researchers to explore data-driven non-rigid reconstruction in the future.

Appearance reconstruction. One aspect that we haven’t explored in this dissertation is appearance modeling of deformable shapes. A previous work [64] optimizes a simplified appearance model, based on lightning represented with spherical harmonics [176]. Recently published neural radiance fields (NeRF) [139] introduced a novel volumetric rendering approach that is very effective at optimizing a photorealistic appearance representation from RGB views. Follow-up works proposed ways to optimize this representation considerably faster [42], [177], [178], even in a few seconds [42], and render it in real-time [179], [180]. There have been a few works extending it to dynamic objects [181]–[184], and it would be interesting to explore learned priors for interactive deformable appearance reconstruction.

Motion reanimation. This dissertation focused on reconstructing the observed object’s motion, for an autonomous robot to react to it in real-time, or for people to replay it from different views in VR/AR. Once recorded, an interesting question is how to interact with this deformable object in novel ways in VR/AR, and animate it using given controls. There has been quite some progress on reenactment of human faces [7], [185], [186] and bodies [187], [188], where manually-designed body skeletons or facial keypoints can be used as driving signals. For general deformable objects we don’t have a skeleton that fits all shapes, but our *neural deformation graphs* could be extended to a generalizable formulation optimized to represent a wide variety of deformable shapes, automatically constructing a *generalized skeleton*.

Differentiable simulation. A core building block of our *neural non-rigid tracker* is differentiable optimization. While we could learn the entire optimization process and directly predict 3D motion using a scene flow network, it’s very useful to restrict the

learning to more specific tasks that require priors, since then we need less data to generalize robustly to novel objects and deformations, and convert components that naturally generalize, such as a Gauss-Newton solver, to differentiable network modules. In the future we could also integrate known physical laws in our tracker by incorporating differentiable modules from *differentiable simulation* [189]–[192], where physically-based simulators, developed for different object types, are made differentiable, in order to estimate the physical object model that fits best to the given observations.

Self-supervised learning. To learn useful priors for non-rigid reconstruction, an important requirement is having a diverse dataset. We introduced a large-scale dataset of non-rigidly deforming objects, and it’s a great starting point for learning data-driven priors. However, you can always benefit from collecting more data, covering a wider range of deformable objects. Expanding our dataset can be done very efficiently, since beside recording new RGB-D videos we only require sparse annotations that can be acquired quickly. On the other hand, there is a large number of diverse object images and videos available online, but without any annotations. Following the success of self-supervised works in natural language processing [156], [157] and image generation [193], [194], we could adapt our approaches to be trained with *limited supervision*.

Bibliography

- [1] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, “Pixelwise View Selection for Unstructured Multi-View Stereo,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [2] R. A. Newcombe, S. Izadi, O. Hilliges, *et al.*, “Kinectfusion: Real-time dense surface mapping and tracking,” in *2011 10th IEEE international symposium on mixed and augmented reality*, Ieee, 2011, pp. 127–136.
- [3] S. Orts-Escolano, C. Rhemann, S. Fanello, *et al.*, “Holoportation: Virtual 3d teleportation in real-time,” in *Proceedings of the 29th annual symposium on user interface software and technology*, 2016, pp. 741–754.
- [4] R. A. Newcombe, D. Fox, and S. M. Seitz, “Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 343–352.
- [5] K. Robinette, S. Blackwell, H. Daanen, M. Boehmer, and S. Fleming, “Civilian american and european surface anthropometry resource (caesar), final report. volume 1. summary,” p. 74, Jun. 2002.
- [6] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, “Smpl: A skinned multi-person linear model,” *ACM transactions on graphics (TOG)*, vol. 34, no. 6, pp. 1–16, 2015.
- [7] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, “Face2face: Real-time face capture and reenactment of rgb videos,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2387–2395.
- [8] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.
- [9] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, “A naturalistic open source movie for optical flow evaluation,” in *European Conf. on Computer Vision (ECCV)*, A. Fitzgibbon *et al.* (Eds.), Ed., ser. Part IV, LNCS 7577, Springer-Verlag, Oct. 2012, pp. 611–625.
- [10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.
- [11] R. A. Güler, N. Neverova, and I. Kokkinos, “Densepose: Dense human pose estimation in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7297–7306.

- [12] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014. arXiv: 1412.6980. [Online]. Available: <http://arxiv.org/abs/1412.6980>.
- [13] **A. Božič**, M. Zollhöfer, C. Theobalt, and M. Nießner, “Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [14] **A. Božič***, P. Palafox*, M. Zollhöfer, A. Dai, J. Thies, and M. Nießner, “Neural non-rigid tracking,” in *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2020.
- [15] **A. Božič**, P. Palafox, M. Zollhöfer, J. Thies, A. Dai, and M. Nießner, “Neural deformation graphs for globally-consistent non-rigid reconstruction,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [16] **A. Božič**, P. Palafox, J. Thies, A. Dai, and M. Nießner, “Transformerfusion: Monocular rgb scene reconstruction using transformers,” in *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2021.
- [17] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “Scannet: Richly-annotated 3d reconstructions of indoor scenes,” in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.
- [18] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3d surface construction algorithm,” *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.
- [19] J. R. Shewchuk *et al.*, *An introduction to the conjugate gradient method without the agonizing pain*, 1994.
- [20] Z. DeVito, M. Mara, M. Zollhöfer, *et al.*, “Opt: A domain specific language for non-linear least squares optimization in graphics and imaging,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 5, pp. 1–27, 2017.
- [21] Y. Li, **A. Božič**, T. Zhang, Y. Ji, T. Harada, and M. Nießner, “Learning to optimize non-rigid tracking,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [22] A. Paszke, S. Gross, F. Massa, *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems (NeurIPS) 32*, 2019, pp. 8024–8035.
- [23] J. T. Barron and B. Poole, “The fast bilateral solver,” in *European Conference on Computer Vision (ECCV)*, Springer, 2016, pp. 617–632.
- [24] R. W. Sumner, J. Schmid, and M. Pauly, “Embedded deformation for shape manipulation,” *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, 80–es, 2007.
- [25] O. Sorkine and M. Alexa, “As-rigid-as-possible surface modeling,” in *Symposium on Geometry processing*, vol. 4, 2007, pp. 109–116.

- [26] J.-L. Blanco, “A tutorial on $se(3)$ transformation parameterizations and on-manifold optimization,” *University of Malaga, Tech. Rep.*, vol. 3, 2010.
- [27] S. Rusinkiewicz and M. Levoy, “Efficient variants of the ICP algorithm,” in *Int. Conf. 3D Digital Imaging and Modeling (3DIM)*, IEEE, 2001, pp. 145–152.
- [28] S. Salti, F. Tombari, and L. Di Stefano, “Shot: Unique signatures of histograms for surface and texture description,” *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.
- [29] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. [Online]. Available: <https://doi.org/10.1023/B:VISI.0000029664.99615.94>.
- [30] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration,” in *2009 IEEE international conference on robotics and automation*, IEEE, 2009, pp. 3212–3217.
- [31] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008, ISSN: 1077-3142. DOI: 10.1016/j.cviu.2007.09.014. [Online]. Available: <http://dx.doi.org/10.1016/j.cviu.2007.09.014>.
- [32] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, “3dmatch: Learning local geometric descriptors from rgb-d reconstructions,” in *CVPR*, 2017.
- [33] A. Dosovitskiy, P. Fischer, E. Ilg, *et al.*, “Flownet: Learning optical flow with convolutional networks,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2758–2766.
- [34] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, “Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8934–8943.
- [35] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, “Volumedeform: Real-time volumetric non-rigid reconstruction,” in *European Conference on Computer Vision*, Springer, 2016, pp. 362–379.
- [36] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, “Real-time large-scale dense rgb-d slam with volumetric fusion,” *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.
- [37] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.
- [38] E. Tretschk, N. Kairanda, R. Dabral, *et al.*, “State of the art in dense monocular non-rigid 3d reconstruction,” *arXiv preprint arXiv:2210.15664*, 2022.
- [39] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, “Implicit neural representations with periodic activation functions,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 7462–7473, 2020.

- [40] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: Learning continuous signed distance functions for shape representation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.
- [41] J. Chibane, T. Alldieck, and G. Pons-Moll, “Implicit functions in feature space for 3d shape reconstruction and completion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6970–6981.
- [42] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *arXiv preprint arXiv:2201.05989*, 2022.
- [43] P. Palafox, **A. Božič**, J. Thies, M. Nießner, and A. Dai, “Npms: Neural parametric models for 3d deformable shapes,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [44] B. Deng, J. P. Lewis, T. Jeruzalski, *et al.*, “Nasa: Neural articulated shape approximation,” in *European Conference on Computer Vision*, Springer, 2020, pp. 612–628.
- [45] P. Palafox, N. Sarafianos, T. Tung, and A. Dai, “Spams: Structured implicit parametric models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 851–12 860.
- [46] A. X. Chang, T. Funkhouser, L. Guibas, *et al.*, “Shapenet: An information-rich 3d model repository,” *arXiv preprint arXiv:1512.03012*, 2015.
- [47] F. Bogo, J. Romero, M. Loper, and M. J. Black, “FAUST: Dataset and evaluation for 3D mesh registration,” in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA: IEEE, Jun. 2014.
- [48] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, “Scene representation networks: Continuous 3d-structure-aware neural scene representations,” in *Advances in Neural Information Processing Systems*, 2019, pp. 1121–1132.
- [49] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, “3d-r2n2: A unified approach for single and multi-view 3d object reconstruction,” in *European conference on computer vision*, Springer, 2016, pp. 628–644.
- [50] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, “Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2304–2314.
- [51] S. Saito, T. Simon, J. Saragih, and H. Joo, “Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 84–93.

- [52] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser, “Learning shape templates with structured implicit functions,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7154–7164.
- [53] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, “Local deep implicit functions for 3d shape,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4857–4866.
- [54] L. Wang, X. Zhao, T. Yu, S. Wang, and Y. Liu, “Normalgan: Learning detailed 3d human from a single rgb-d image,” in *European Conference on Computer Vision*, Springer, 2020, pp. 430–446.
- [55] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao, “NeuralRecon: Real-time coherent 3D reconstruction from monocular video,” *CVPR*, 2021.
- [56] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.
- [57] K. Wang and S. Shen, “Mvdepthnet: Real-time multiview depth estimation neural network,” in *2018 International conference on 3d vision (3DV)*, IEEE, 2018, pp. 248–257.
- [58] Y. Hou, J. Kannala, and A. Solin, “Multi-view stereo by temporal nonparametric fusion,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2651–2660.
- [59] S. Im, H.-G. Jeon, S. Lin, and I. S. Kweon, “Dpsnet: End-to-end deep plane sweep stereo,” *arXiv preprint arXiv:1905.00538*, 2019.
- [60] X. Long, L. Liu, W. Li, C. Theobalt, and W. Wang, “Multi-view depth estimation using epipolar spatio-temporal network,” *CVPR*, 2021.
- [61] A. Düzçeker, S. Galliani, C. Vogel, P. Speciale, M. Dusmanu, and M. Pollefeys, *Deepvideomvs: Multi-view stereo on video with recurrent spatio-temporal fusion*, 2020. arXiv: 2012.02177 [cs.CV].
- [62] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic, “Killingfusion: Non-rigid 3d reconstruction without correspondences,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 3, 2017, p. 7.
- [63] M. Slavcheva, M. Baust, and S. Ilic, “Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2646–2655.
- [64] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu, “Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 3, p. 32, 2017.
- [65] S. Wang, X. Zuo, C. Du, R. Wang, J. Zheng, and R. Yang, “Dynamic non-rigid objects reconstruction with a single rgb-d sensor,” *Sensors*, vol. 18, no. 3, p. 886, 2018.

- [66] M. Dou, S. Khamis, Y. Degtyarev, *et al.*, “Fusion4d: Real-time performance capture of challenging scenes,” *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 114, 2016.
- [67] M. Dou, P. Davidson, S. R. Fanello, *et al.*, “Motion2fusion: Real-time volumetric performance capture,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, p. 246, 2017.
- [68] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li, “Dense human body correspondences using convolutional networks,” in *Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [69] T. Schmidt, R. Newcombe, and D. Fox, “Self-supervised visual descriptor learning for dense correspondence,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 420–427, 2016.
- [70] M. Zollhöfer, P. Stotko, A. Görlitz, *et al.*, “State of the Art on 3D Reconstruction with RGB-D Cameras,” *Computer Graphics Forum (Eurographics State of the Art Reports 2018)*, vol. 37, no. 2, 2018.
- [71] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, “Kintinuous: Spatially extended kinectfusion,” in *Proc. RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.
- [72] J. Chen, D. Bautembach, and S. Izadi, “Scalable real-time volumetric surface reconstruction,” *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 32, no. 4, 113:1–113:16, Jul. 2013, ISSN: 0730-0301.
- [73] F. Steinbrucker, C. Kerl, and D. Cremers, “Large-scale multi-resolution surface reconstruction from rgb-d sequences,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3264–3271.
- [74] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, “Real-time 3D reconstruction at scale using voxel hashing,” *ACM Trans. on Graphics*, vol. 32, no. 6, p. 169, 2013.
- [75] Q.-Y. Zhou and V. Koltun, “Dense scene reconstruction with points of interest,” *ACM Trans. on Graphics*, vol. 32, no. 4, p. 112, 2013.
- [76] S. Choi, Q.-Y. Zhou, and V. Koltun, “Robust reconstruction of indoor scenes,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015, pp. 5556–5565.
- [77] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, “Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration,” *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017.
- [78] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, “Real-time 3D reconstruction in dynamic scenes using point-based fusion,” in *Proc. Int. Conf. 3D Vision (3DV)*, Washington, DC, USA: IEEE Computer Society, 2013, pp. 1–8, ISBN: 978-0-7695-5067-1. DOI: 10.1109/3DV.2013.9. [Online]. Available: <http://dx.doi.org/10.1109/3DV.2013.9>.

- [79] D. Lefloch, T. Weyrich, and A. Kolb, “Anisotropic point-based fusion,” in *Proc. Int. Conf. Information Fusion (FUSION)*, Jul. 2015, pp. 1–9.
- [80] D. Lefloch, M. Kluge, H. Sarbolandi, T. Weyrich, and A. Kolb, “Comprehensive use of curvature for robust and accurate online surface reconstruction,” *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 10.1109/T-PAMI.2017.2648803, 2017.
- [81] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, “Elasticfusion: Dense SLAM without a pose graph,” in *Proceedings of Robotics: Science and Systems*, 2015.
- [82] Q.-Y. Zhou, S. Miller, and V. Koltun, “Elastic fragments for dense scene reconstruction,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2013, pp. 473–480.
- [83] M. Wand, B. Adams, M. Ovsjanikov, *et al.*, “Efficient reconstruction of non-rigid shape and motion from real-time 3d scanner data,” *ACM Transactions on Graphics (TOG)*, vol. 28, no. 2, p. 15, 2009.
- [84] T. Yu, Z. Zheng, K. Guo, *et al.*, “DoubleFusion: Real-time Capture of Human Performances with Inner Body Shapes from a Single Depth Sensor,” eng, in *31st IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018.
- [85] C. Wu, C. Stoll, L. Valgaerts, and C. Theobalt, “On-set performance capture of multiple actors with a stereo camera,” *ACM Trans. Graph.*, vol. 32, no. 6, 161:1–161:11, 2013. DOI: 10.1145/2508363.2508418. [Online]. Available: <https://doi.org/10.1145/2508363.2508418>.
- [86] M. Dou, J. Taylor, H. Fuchs, A. Fitzgibbon, and S. Izadi, “3d scanning deformable objects with a single rgbd sensor,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 493–501.
- [87] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, “Performance capture from sparse multi-view video,” *ACM Trans. Graph.*, vol. 27, no. 3, 98:1–98:10, Aug. 2008, ISSN: 0730-0301. DOI: 10.1145/1360612.1360697. [Online]. Available: <http://doi.acm.org/10.1145/1360612.1360697>.
- [88] D. Vlasic, I. Baran, W. Matusik, and J. Popović, “Articulated mesh animation from multi-view silhouettes,” in *ACM SIGGRAPH 2008 papers*, 2008, pp. 1–9.
- [89] G. Ye, Y. Liu, N. Hasler, X. Ji, Q. Dai, and C. Theobalt, “Performance Capture of Interacting Characters with Handheld Kinects,” in *Proc. ECCV*, Springer, 2012, pp. 828–841.
- [90] R. Wang, L. Wei, E. Vouga, *et al.*, “Capturing dynamic textured surfaces of moving targets,” *CoRR*, vol. abs/1604.02801, 2016. arXiv: 1604.02801. [Online]. Available: <http://arxiv.org/abs/1604.02801>.

- [91] A. E. Johnson and M. Hebert, “Using spin images for efficient object recognition in cluttered 3d scenes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 433–449, May 1999, ISSN: 0162-8828. DOI: 10.1109/34.765655. [Online]. Available: <https://doi.org/10.1109/34.765655>.
- [92] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, “Recognizing objects in range data using regional point descriptors,” vol. 3, May 2004, pp. 224–237. DOI: 10.1007/978-3-540-24672-5_18.
- [93] F. Tombari, S. Salti, and L. D. Stefano, “Unique signatures of histograms for local surface description,” in *European conference on computer vision*, Springer, 2010, pp. 356–369.
- [94] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Aligning point cloud views using persistent feature histograms,” in *2008 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2008, pp. 3384–3391.
- [95] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, “Discriminative learning of deep convolutional feature point descriptors,” in *ICCV*, IEEE Computer Society, 2015, pp. 118–126.
- [96] K. Simonyan, A. Vedaldi, and A. Zisserman, “Learning local feature descriptors using convex optimisation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1573–1585, 2014.
- [97] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, “Matchnet: Unifying feature and metric learning for patch-based matching,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3279–3286.
- [98] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, “Lift: Learned invariant feature transform.,” *CoRR*, vol. abs/1603.09114, 2016. [Online]. Available: <http://dblp.uni-trier.de/db/journals/corr/corr1603.html#YiT16>.
- [99] J. Žbontar and Y. LeCun, “Computing the stereo matching cost with a convolutional neural network,” English (US), in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, vol. 07-12-June-2015, IEEE Computer Society, Oct. 2015, pp. 1592–1599, ISBN: 9781467369640. DOI: 10.1109/CVPR.2015.7298767.
- [100] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, “3d-codded: 3d correspondences by deep deformation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 230–246.
- [101] P. K. Nathan Silberman Derek Hoiem and R. Fergus, “Indoor segmentation and support inference from rgbd images,” in *ECCV*, 2012.
- [102] S. Song, S. P. Lichtenberg, and J. Xiao, “Sun rgb-d: A rgb-d scene understanding benchmark suite,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 567–576.

- [103] K. Guo, F. Xu, Y. Wang, Y. Liu, and Q. Dai, “Robust non-rigid motion tracking and surface reconstruction using l0 regularization,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3083–3091.
- [104] B. G. V. Kumar, G. Carneiro, and I. D. Reid, “Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions,” *CoRR*, vol. abs/1512.09272, 2015. arXiv: 1512.09272. [Online]. Available: <http://arxiv.org/abs/1512.09272>.
- [105] D. Mehta, S. Sridhar, O. Sotnychenko, *et al.*, “Vnect: Real-time 3d human pose estimation with a single rgb camera,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 44, 2017.
- [106] K. Zampogiannis, C. Fermuller, and Y. Aloimonos, “Topology-aware non-rigid point cloud registration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- [107] S. Wang, S. Ryan Fanello, C. Rhemann, S. Izadi, and P. Kohli, “The global patch collider,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 127–135.
- [108] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “FlowNet 2.0: Evolution of optical flow estimation with deep networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.
- [109] M. Zollhöfer, M. Nießner, S. Izadi, *et al.*, “Real-time non-rigid reconstruction using an rgb-d camera,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–12, 2014.
- [110] P. Liu, M. Lyu, I. King, and J. Xu, “SelfFlow: Self-supervised learning of optical flow,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4571–4580.
- [111] W.-C. Ma, S. Wang, R. Hu, Y. Xiong, and R. Urtasun, “Deep rigid instance scene flow,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3614–3622.
- [112] A. Behl, D. Paschalidou, S. Donn e, and A. Geiger, “PointFlowNet: Learning representations for rigid motion estimation from point clouds,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7962–7971.
- [113] X. Liu, C. R. Qi, and L. J. Guibas, “FlowNet3d: Learning scene flow in 3d point clouds,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 529–537.
- [114] Z. Wang, S. Li, H. Howard-Jenkins, V. Prisacariu, and M. Chen, “FlowNet3d++: Geometric losses for deep scene flow estimation,” in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 91–98.

- [115] X. Wang, A. Jabri, and A. A. Efros, “Learning correspondence from the cycle-consistency of time,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2566–2576.
- [116] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, and M.-H. Yang, “Joint-task self-supervised learning for temporal correspondence,” in *Advances in Neural Information Processing Systems*, 2019, pp. 318–328.
- [117] Z. Lai, E. Lu, and W. Xie, “Mast: A memory-augmented self-supervised tracker,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6479–6488.
- [118] C.-H. Chang, C.-N. Chou, and E. Y. Chang, “Clkn: Cascaded lucas-kanade networks for image alignment,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2213–2221.
- [119] L. Han, M. Ji, L. Fang, and M. Nießner, “Regnet: Learning the optimization of direct image-to-image pose registration,” *arXiv preprint arXiv:1812.10212*, 2018.
- [120] Z. Lv, F. Dellaert, J. M. Rehg, and A. Geiger, “Taking a deeper look at the inverse compositional algorithm,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4581–4590.
- [121] C. Tang and P. Tan, “Ba-net: Dense bundle adjustment network,” *arXiv preprint arXiv:1806.04807*, 2018.
- [122] A. Avetisyan, A. Dai, and M. Nießner, “End-to-end cad model retrieval and 9dof alignment in 3d scans,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 2551–2560.
- [123] M. Götz and H. Anzt, “Machine learning-aided numerical linear algebra: Convolutional neural networks for the efficient preconditioner generation,” in *2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA)*, 2018, pp. 49–56.
- [124] J. Sappl, L. Seiler, M. Harders, and W. Rauch, “Deep learning of preconditioners for conjugate gradient solvers in urban water related problems,” *arXiv preprint arXiv:1906.06925*, 2019.
- [125] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, “Spatial transformer networks,” in *Advances in Neural Information Processing Systems (NeurIPS)* 28, 2015, pp. 2017–2025.
- [126] N. Mayer, E. Ilg, P. Hausser, *et al.*, “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4040–4048.
- [127] M. Kazhdan, M. Bolitho, and H. Hoppe, “Poisson surface reconstruction,” in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

- [128] F. Mueller, F. Bernard, O. Sotnychenko, *et al.*, “Generated hands for real-time 3d hand tracking from monocular rgb,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 49–59.
- [129] A. Collet, M. Chuang, P. Sweeney, *et al.*, “High-quality streamable free-viewpoint video,” *ACM Transactions on Graphics (ToG)*, vol. 34, no. 4, pp. 1–13, 2015.
- [130] K. Guo, P. Lincoln, P. Davidson, *et al.*, “The relightables: Volumetric performance capture of humans with realistic relighting,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–19, 2019.
- [131] M. Innmann, K. Kim, J. Gu, *et al.*, “Nrmvs: Non-rigid multi-view stereo,” in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 2754–2763.
- [132] V. Sidhu, E. Tretschk, V. Golyanik, A. Agudo, and C. Theobalt, “Neural dense non-rigid structure from motion with latent space constraints,” in *European Conference on Computer Vision (ECCV)*, 2020.
- [133] I. Baran and J. Popović, “Automatic rigging and animation of 3d characters,” *ACM Transactions on graphics (TOG)*, vol. 26, no. 3, 72–es, 2007.
- [134] E. Tretschk, A. Tewari, M. Zollhöfer, V. Golyanik, and C. Theobalt, “Demea: Deep mesh autoencoders for non-rigidly deforming objects,” in *European Conference on Computer Vision*, Springer, 2020, pp. 601–617.
- [135] Z. Xu, Y. Zhou, E. Kalogerakis, C. Landreth, and K. Singh, “Rignet: Neural rigging for articulated characters,” *arXiv preprint arXiv:2005.00559*, 2020.
- [136] Z. Huang, Y. Xu, C. Lassner, H. Li, and T. Tung, “Arch: Animatable reconstruction of clothed humans,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3093–3102.
- [137] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt, “Patchnets: Patch-based generalizable deep implicit 3d shape representations,” in *European Conference on Computer Vision*, Springer, 2020, pp. 293–309.
- [138] M. Kazhdan and H. Hoppe, “Screened poisson surface reconstruction,” *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, pp. 1–13, 2013.
- [139] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.
- [140] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, “Occupancy flow: 4d reconstruction by learning particle dynamics,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5379–5389.
- [141] Y. Li, H. Takehara, T. Taketomi, B. Zheng, and M. Nießner, “4dcomplete: Non-rigid motion estimation beyond the observable surface,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12706–12716.

- [142] V. Lepetit, F. Moreno-Noguer, and P. Fua, “Epnnp: An accurate o (n) solution to the pnp problem,” *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.
- [143] Y. Chen and G. Medioni, “Object modeling by registration of multiple range images,” *Image Vision Comput.*, vol. 10, pp. 145–155, Jan. 1992. DOI: 10.1109/ROBOT.1991.132043.
- [144] B. Graham, M. Engelcke, and L. van der Maaten, “3d semantic segmentation with submanifold sparse convolutional networks,” *CVPR*, 2018.
- [145] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
- [146] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” *arXiv preprint arXiv:1706.02413*, 2017.
- [147] C. R. Qi, O. Litany, K. He, and L. J. Guibas, “Deep hough voting for 3d object detection in point clouds,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9277–9286.
- [148] Y. Nie, J. Hou, X. Han, and M. Nießner, “Rfd-net: Point scene understanding by semantic instance reconstruction,” *arXiv preprint arXiv:2011.14744*, 2020.
- [149] A. Dai and M. Nießner, “3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 452–468.
- [150] W. Wang, R. Yu, Q. Huang, and U. Neumann, “Sgpn: Similarity group proposal network for 3d point cloud instance segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2569–2578.
- [151] J. Hou, A. Dai, and M. Nießner, “3d-sis: 3d semantic instance segmentation of rgb-d scans,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4421–4430.
- [152] J. Hou, A. Dai, and M. Nießner, “Revealnet: Seeing behind objects in rgb-d scans,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2098–2107.
- [153] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, “Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam,” *arXiv preprint arXiv:2007.11898*, 2020.
- [154] L. Von Stumberg, V. Usenko, and D. Cremers, “Direct sparse visual-inertial odometry using dynamic marginalization,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2510–2517.
- [155] A. Sinha, Z. Murez, J. Bartolozzi, V. Badrinarayanan, and A. Rabinovich, “Deltas: Depth estimation by learning triangulation and densification of sparse points,” in *ECCV*, 2020. [Online]. Available: <https://arxiv.org/abs/2003.08933>.

- [156] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [157] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [158] T. B. Brown, B. Mann, N. Ryder, *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.
- [159] C. Liu, J. Gu, K. Kim, S. G. Narasimhan, and J. Kautz, “Neural rgb-to-d sensing: Depth and uncertainty from a video camera,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 986–10 995.
- [160] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche, “Mono-fusion: Real-time 3d reconstruction of small scenes with a single web camera,” in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, IEEE, 2013, pp. 83–88.
- [161] M. Bleyer, C. Rhemann, and C. Rother, “Patchmatch stereo-stereo matching with slanted support windows,” in *Bmvc*, vol. 11, 2011, pp. 1–11.
- [162] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang, “Surfacenet: An end-to-end 3d neural network for multiview stereopsis,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2307–2315.
- [163] Z. Murez, T. van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich, “Atlas: End-to-end 3d scene reconstruction from posed images,” in *ECCV*, 2020. [Online]. Available: <https://arxiv.org/abs/2003.10432>.
- [164] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, “Transformers in vision: A survey,” *arXiv preprint arXiv:2101.01169*, 2021.
- [165] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European Conference on Computer Vision*, Springer, 2020, pp. 213–229.
- [166] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.
- [167] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [168] N. Parmar, A. Vaswani, J. Uszkoreit, *et al.*, “Image transformer,” in *International Conference on Machine Learning*, PMLR, 2018, pp. 4055–4064.
- [169] P. Zins, Y. Xu, E. Boyer, S. Wuhler, and T. Tung, “Learning implicit 3d representations of dressed humans from sparse views,” *arXiv preprint arXiv:2104.08013*, 2021.

- [170] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, “Convolutional occupancy networks,” in *European Conference on Computer Vision (ECCV)*, Cham: Springer International Publishing, Aug. 2020.
- [171] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [172] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.
- [173] J. Hu, M. Ozay, Y. Zhang, and T. Okatani, “Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries,” 2019.
- [174] A. Dai, C. Diller, and M. Nießner, “Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 849–858.
- [175] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser, *et al.*, “Local implicit grid representations for 3d scenes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6001–6010.
- [176] R. Ramamoorthi, “Modeling illumination variation with spherical harmonics,” *Face Processing: Advanced Modeling Methods*, pp. 385–424, 2006.
- [177] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, “Plenoxels: Radiance fields without neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5501–5510.
- [178] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, “Tensorf: Tensorial radiance fields,” *arXiv preprint arXiv:2203.09517*, 2022.
- [179] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, “Plenotrees for real-time rendering of neural radiance fields,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.
- [180] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi, “Mobilerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures,” *arXiv preprint arXiv:2208.00277*, 2022.
- [181] K. Park, U. Sinha, J. T. Barron, *et al.*, “Nerfies: Deformable neural radiance fields,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5865–5874.
- [182] K. Park, U. Sinha, P. Hedman, *et al.*, “Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields,” *arXiv preprint arXiv:2106.13228*, 2021.
- [183] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, “D-nerf: Neural radiance fields for dynamic scenes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 318–10 327.

- [184] T. Li, M. Slavcheva, M. Zollhoefer, *et al.*, “Neural 3d video synthesis from multi-view video,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5521–5531.
- [185] S. Ma, T. Simon, J. Saragih, *et al.*, “Pixel codec avatars,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 64–73.
- [186] S. Bi, S. Lombardi, S. Saito, *et al.*, “Deep relightable appearance models for animatable faces,” *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–15, 2021.
- [187] T. Bagautdinov, C. Wu, T. Simon, *et al.*, “Driving-signal aware full-body avatars,” *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–17, 2021.
- [188] S. Starke, I. Mason, and T. Komura, “Deepphase: Periodic autoencoders for learning motion phase manifolds,” *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–13, 2022.
- [189] H.-y. Chen, E. Tretschk, T. Stuyck, *et al.*, “Virtual elastic objects,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 827–15 837.
- [190] Y. Kozlov, H. Xu, M. Bächer, D. Bradley, M. Gross, and T. Beeler, “Data-driven physical face inversion,” *arXiv preprint arXiv:1907.10402*, 2019.
- [191] D. Hahn, P. Banzet, J. M. Bern, and S. Coros, “Real2sim: Visco-elastic parameter estimation from dynamic motion,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–13, 2019.
- [192] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, “Add: Analytically differentiable dynamics for multi-body systems with frictional contact,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.
- [193] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [194] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical text-conditional image generation with clip latents,” *arXiv preprint arXiv:2204.06125*, 2022.

Appendix

A Open-source Code & Videos

A.1 DeepDeform

- Dataset and Evaluation Code: <https://github.com/AljazBozic/DeepDeform>
- Benchmark: https://kaldir.vc.in.tum.de/deepdeform_benchmark
- Project: <http://niessnerlab.org/projects/bozic2020deepdeform.html>
- Video: <https://www.youtube.com/watch?v=OrHLacCDZVQ>

A.2 Neural Non-Rigid Tracking

- Source Code: <https://github.com/DeformableFriends/NeuralTracking>
- Project: <http://niessnerlab.org/projects/bozic2020nnrt.html>
- Video: <https://www.youtube.com/watch?v=nqYaxM6Rj8I>

A.3 Neural Deformation Graphs

- Source Code: <https://github.com/AljazBozic/NeuralGraph>
- Project: https://aljazbozic.github.io/neural_deformation_graphs
- Video: <https://www.youtube.com/watch?v=vyq36eFkdWo>

A.4 TransformerFusion

- Evaluation Code: <https://github.com/AljazBozic/TransformerFusion>
- Project: <https://aljazbozic.github.io/transformerfusion>
- Video: <https://www.youtube.com/watch?v=LIpTKYfKSqw>

B Authored and Co-authored Publications

1. **A. Božič**, M. Zollhöfer, C. Theobalt, and M. Nießner, “Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020
2. **A. Božič***, P. Palafox*, M. Zollhöfer, A. Dai, J. Thies, and M. Nießner, “Neural non-rigid tracking,” in *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2020
3. **A. Božič**, P. Palafox, M. Zollhöfer, J. Thies, A. Dai, and M. Nießner, “Neural deformation graphs for globally-consistent non-rigid reconstruction,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021
4. **A. Božič**, P. Palafox, J. Thies, A. Dai, and M. Nießner, “Transformerfusion: Monocular rgb scene reconstruction using transformers,” in *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2021
5. Y. Li, **A. Božič**, T. Zhang, Y. Ji, T. Harada, and M. Nießner, “Learning to optimize non-rigid tracking,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020
6. P. Palafox, **A. Božič**, J. Thies, M. Nießner, and A. Dai, “Npms: Neural parametric models for 3d deformable shapes,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021
7. D. Chang, **A. Božič**, T. Zhang, *et al.*, “Rc-mvsnet: Unsupervised multi-view stereo with neural rendering,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022

C Original Publications

In this chapter we include the original versions of peer-reviewed publications [13]–[16]. These publications form the core of this cumulative thesis, and are summarized in Chapters 3, 4, 5, 6 (with minor content and format adapti). We additionally provide a copyright notice and a short summary for each publication, and evaluate author’s individual contributions.

C.1 DeepDeform: Learning Non-rigid RGB-D Reconstruction with Semi-supervised Data

COPYRIGHT

©2020 IEEE. Reprinted, with permission, from
 Aljaž Božič, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner
DeepDeform: Learning Non-rigid RGB-D Reconstruction with Semi-supervised Data

IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) 2020

DOI: 10.1109/CVPR42600.2020.00703

SUMMARY

Applying data-driven approaches to non-rigid 3D reconstruction has been difficult, which we believe can be attributed to the lack of a large-scale training corpus. Unfortunately, existing methods fail for important cases such as highly non-rigid deformations. We first address this problem of lack of data by introducing a novel semi-supervised strategy to obtain dense inter-frame correspondences from a sparse set of annotations. This way, we obtain a large dataset of 400 scenes, over 390,000 RGB-D frames, and 5,533 densely aligned frame pairs; in addition, we provide a test set along with several metrics for evaluation. Based on this corpus, we introduce a data-driven non-rigid feature matching approach, which we integrate into an optimization-based reconstruction pipeline. Here, we propose a new neural network that operates on RGB-D frames, while maintaining robustness under large non-rigid deformations and producing accurate predictions. Our approach significantly outperforms existing non-rigid reconstruction methods that do not use learned data terms, as well as learning-based approaches that only use self-supervision.

INDIVIDUAL CONTRIBUTIONS

Leading role in realizing the scientific project.

Problem definition	<i>significantly contributed</i>
Literature survey	<i>significantly contributed</i>
Implementation	<i>significantly contributed</i>
Experimental evaluation	<i>significantly contributed</i>
Preparation of the manuscript	<i>significantly contributed</i>

In accordance with the *IEEE Thesis / Dissertation Reuse Permissions*, we include the accepted version of the original publication [13] in the following.

DeepDeform: Learning Non-rigid RGB-D Reconstruction with Semi-supervised Data

Aljaž Božič¹

Michael Zollhöfer²

Christian Theobalt³

Matthias Nießner¹

¹Technical University of Munich

²Stanford University

³Max Planck Institute for Informatics

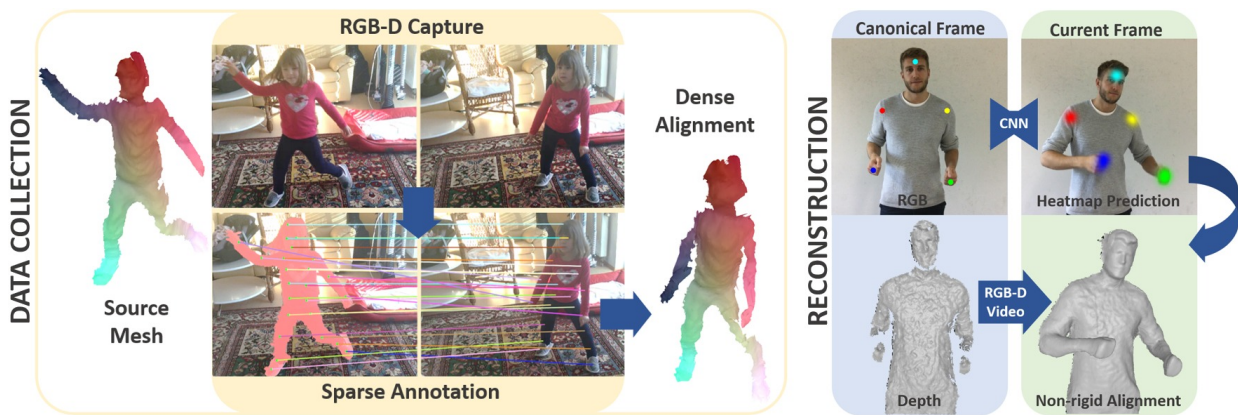


Figure 1: We propose a semi-supervised strategy combining self-supervision with sparse annotations to build a large-scale RGB-D dataset of non-rigidly deforming scenes (400 scenes, 390,000 frames, 5,533 densely aligned frame pairs). With this data, we propose a new method for non-rigid matching, which we integrate into a non-rigid reconstruction approach.

Abstract

Applying data-driven approaches to non-rigid 3D reconstruction has been difficult, which we believe can be attributed to the lack of a large-scale training corpus. Unfortunately, this method fails for important cases such as highly non-rigid deformations. We first address this problem of lack of data by introducing a novel semi-supervised strategy to obtain dense inter-frame correspondences from a sparse set of annotations. This way, we obtain a large dataset of 400 scenes, over 390,000 RGB-D frames, and 5,533 densely aligned frame pairs; in addition, we provide a test set along with several metrics for evaluation. Based on this corpus, we introduce a data-driven non-rigid feature matching approach, which we integrate into an optimization-based reconstruction pipeline. Here, we propose a new neural network that operates on RGB-D frames, while maintaining robustness under large non-rigid deformations and producing accurate predictions. Our approach significantly outperforms existing non-rigid reconstruction methods that do not use learned data terms, as well as learning-based approaches that only use self-supervision.

1. Introduction

Non-rigid 3D reconstruction, i.e., the dense, space-time coherent capture of non-rigidly deforming surfaces in full temporal correspondence, is key towards obtaining 3D abstractions of the moving real world. The wide availability of commodity RGB-D sensors, such as the Microsoft Kinect or Intel Realsense, has led to tremendous progress on static scene reconstruction methods. However, robust and high-quality reconstruction of non-rigidly moving scenes with one depth camera is still challenging. Applications for real-time non-rigid reconstruction range from augmented (AR) and virtual reality (VR) up to building realistic 3D holograms for fully immersive teleconferencing systems. The seminal DynamicFusion [28] approach was the first to show dynamic non-rigid reconstruction in real-time. Extensions primarily differ in the used energy formulation. Some methods use hand-crafted data terms based on dense geometry [28, 36, 37], dense color and geometry [14, 47], and sparse feature constraints [17]. Other approaches leverage multi-camera RGB-D setups [9, 8] for higher robustness. However, there are very few reconstruction methods that use learning-based data terms for general real-world scenes rather than specific scenarios [48], and that are trained to be robust under real-world appearance variation and diffi-

Data / Benchmark: <https://github.com/AljazBozic/DeepDeform>

cult motions. One reason for this is the lack of a large-scale training corpus. One recent approach [33] proposes self-supervision for ground truth generation, i.e., they employ DynamicFusion [28] for reconstruction and train a non-rigid correspondence descriptor on the computed inter-frame correspondences. However, we show that existing non-rigid reconstruction methods are not robust enough to handle realistic non-rigid sequences; therefore this tracking-based approach does not scale to training data generation for real world scenes. Unfortunately, this means that self-supervision exactly fails for many challenging scenarios such as highly non-rigid deformations and fast scene motion. By design, this approach trained with self-supervision cannot be better than the employed tracker. We propose to employ semi-supervised training data by combining self-supervision with sparse user annotations to obtain dense inter-frame correspondences. The annotated sparse point correspondences guide non-rigid reconstruction; this allows us to handle even challenging motions. The result is a large dataset of 400 scenes, over 390,000 RGB-D frames, and 5,533 densely aligned frame pairs. Based on this novel training corpus, we develop a new non-rigid correspondence matching approach (see Sec. 3) that finds accurate matches between RGB-D frames and is robust to difficult real world deformations. We further propose a re-weighting scheme that gives more weight to corner cases and challenging deformations during training. Given a keypoint in a source frame, our approach predicts a probability heatmap of the corresponding location in the target frame. Finally, we integrate our learned data term into a non-rigid reconstruction pipeline that combines learned heatmap matches with a dense RGB-D reconstruction objective. In addition, we introduce a new benchmark and metric for evaluating RGB-D based non-rigid 3D correspondence matching and reconstruction. We extensively compare our new data-driven approach to existing hand-crafted features. We also integrate the learned features into a non-rigid reconstruction framework, leading to significant improvement over state of the art. In sum, our contributions are:

- A semi-supervised labeling approach for dense non-rigid correspondence learning, resulting in a dataset featuring 400 annotated dynamic RGB-D sequences and 5,533 densely aligned frame pairs.
- A novel data-driven non-rigid correspondence matching strategy that leads to more robust correspondence estimation compared to the state-of-the-art hand-crafted and learned descriptors, especially in the case of extreme deformations.
- A non-rigid reconstruction approach for general scenes that combines learned and geometric data-terms and handles significantly faster and more complex motions than the state-of-the-art.



Figure 2: Our large-scale dataset contains a large variety of dynamic sequences with segmentation masks and point correspondences between different RGB-D frames.

2. Related Work

Our approach is related to several research areas, such as volumetric 3D scene reconstruction, non-rigid object tracking, and learned correspondence matching. We focus our discussion on the most related RGB-D based techniques. For a detailed discussion, we refer to the recent survey [59].

Volumetric Scene Reconstruction Reconstructing static environments with a single RGB-D sensor has had a long history in vision and graphics, including KinectFusion [27, 18], which employs a uniform voxel grid to represent the scene as a truncated signed distance function (TSDF) [4], as well as many extensions to large-scale scenes [49, 2, 40, 29]. These techniques track the 6-DoF camera motion by solving a geometric model-to-frame alignment problem using a fast data-parallel variant of the point-to-plane Iterative Closest Point (ICP) algorithm [30]. Globally consistent reconstruction based on Bundle Adjustment [57, 3] was for a long time only possible offline; data-parallel solvers now enable real-time frame rates [6]. An alternative to TSDFs are point-based scene representations [20, 23, 22]. Recent techniques also employ non-rigid registration to robustly handle loop closures [50, 58].

Non-Rigid Reconstruction The reconstruction of general non-rigidly deforming objects based on real-time scan data has a long tradition [44]. One class of methods uses pre-defined templates, e.g., human templates, to capture pose and time-varying shape of clothed humans from RGB-D [54] or stereo camera data [51]. First template-less approaches had slow offline runtimes and only worked for slow and simple motions. The approach of [10] solves a global optimization problem to reconstruct the canonical

shape of a non-rigidly deforming object given an RGB-D video sequence as input, but does not recover the time-dependent non-rigid motion across the entire sequence. The first approach to demonstrate truly dynamic reconstruction of non-rigid deformation and rest shape in real-time was DynamicFusion [28]. Since this seminal work, many extensions have been proposed. VolumeDeform [17] improves tracking quality based on sparse feature alignment. In addition, they parameterize the deformation field based on a dense volumetric grid instead of a sparse deformation graph. The KillingFusion [36] and SobolevFusion [37] approaches allow for topology changes, but do not recover dense space-time correspondence along the complete input sequence. Other approaches jointly optimize for geometry, albedo, and motion [14] to obtain higher robustness and better quality. The approach of Wang et al. [47] employs global optimization to minimize surface tracking errors. In contrast to these methods using a single RGB-D camera, other techniques use multiple color [7, 43] or depth cameras [52, 45, 9, 8], which enables high-quality reconstruction at the cost of more complex hardware. We propose a new learning-based correspondence matching and reconstruction approach that outperforms existing techniques.

Learning Rigid Correspondence Matching Historically, correspondence matching for the task of rigid registration has been based on hand-crafted geometry descriptors [19, 11, 42, 32, 31]. If color information is available in addition to depth, SIFT [24] or SURF [1] can be used to establish a sparse set of feature matches between RGB-D frames. More recently, 2D descriptors for feature matching in static scenes have been learned directly from large-scale training corpora [34, 35, 15, 53, 60]. The Matchnet [15] approach employs end-to-end training of a CNN to extract and match patch-based features in 2D image data. Descriptors for the rigid registration of static scenes can be learned and matched directly in 3D space with the 3DMatch [56] architecture. Visual descriptors for dense correspondence estimation can be learned in a self-supervised manner by employing a dense reconstruction approach to automatically label correspondences in RGB-D recordings [33]. Descriptor learning and matching for static scenes has been well-studied, but is lacking in the challenging non-rigid scenario. While class-specific dense matching of non-rigid scenes has been learned for specific object classes [48, 12], none of these techniques can handle arbitrary deforming non-rigid objects. We believe one reason for this is the lack of a large-scale training corpus. In this work, we propose such a corpus and demonstrate how non-rigid matching between depth images can be learned end-to-end.

RGB-D Datasets While we have seen a plethora of RGB-D datasets for static scenes, such as NYU [26], SUN RGB-

D [38], and ScanNet [5], the largest of which have thousands of scans, non-rigid RGB-D datasets remain in their infancy. While these datasets can be used to pretrain networks for the task of non-rigid correspondence matching, they do not capture the invariants that are useful for the much harder non-rigid setting, and thus lead to sub-par accuracy. Current non-rigid reconstruction datasets are far too small and often limited to specific scene types [7, 52], which is not sufficient to provide the required training data for supervised learning. The datasets that provide real-world depth recordings [13, 17] do not come with ground truth reconstructions, which makes objectively benchmarking different approaches challenging. Other datasets that are commonly used for evaluation do not provide real-world depth data, e.g., [7, 43]. In this work, we introduce the first large-scale dataset for non-rigid matching based on semi-supervised labeling and provide a benchmark enabling objective comparison of different approaches.

3. Data-driven Non-Rigid Matching

Our goal is to find matches between a source and a target RGB-D frame. To this end, we propose a network architecture for RGB-D matching based on a Siamese network [21] with two towers. Input to the network are two local patches of size 224×224 pixels each (with 3 color channels and 3-dimensional points in camera coordinate space). We assume that the source patch is centered at a feature point.

Heatmap The goal is to predict a probability heatmap \mathcal{H} in the target frame that gives the likelihood of the location of the source point. First, we compute a *sigmoid-heatmap*:

$$\mathcal{H}_{sg} = \sigma_{sg}(\mathbf{H}(D_{out})) .$$

It is computed based on a sigmoid activation σ_{sg} to map responses to $[0, 1]$. Here, D_{out} is the output feature map of the last layer of the decoder and \mathbf{H} is a convolutional layer converting feature space into heatmap values. This is equivalent to independent binary classification problems per pixel. Second, we also compute a *softmax-heatmap*:

$$\mathcal{H}_{sm} = \sigma_{sm}(\mathbf{H}(D_{out})) .$$

Here we use a softmax activation σ_{sm} to make the complete heatmap a probability distribution, i.e., it sums to one. As ground truth for the heatmap prediction we could take an image with zero values everywhere except at the ground truth pixel position that we set to one. To prevent the trained network from predicting only zero values, we apply a Gaussian kernel $G_{x_{gt}}$ around the ground truth pixel. It sets the ground truth pixel's value to one and decays the neighboring pixel values to zero with standard deviation of 7 pixels, resulting in the ground truth heatmap \mathcal{H}_{gt} . We also add larger weight to pixels close to the ground truth pixel, defining the

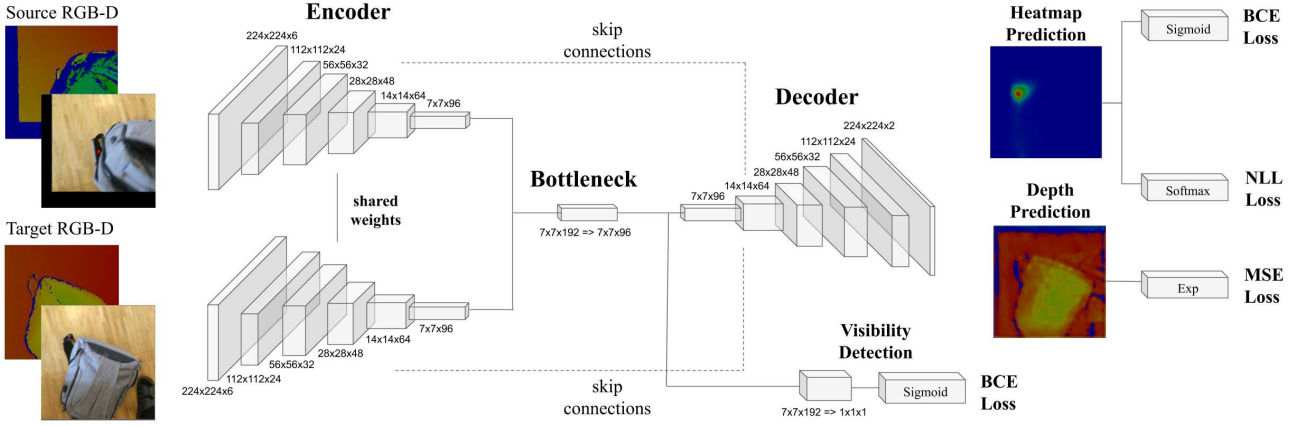


Figure 3: We devise an end-to-end architecture for RGB-D matching based on a Siamese network to find matches between a source and a target frame. Our network is based on two towers that share the encoder and have a decoder that predicts two probability heatmaps in the target frame that encode the likelihood of the location of the source point. Our network also predicts a depth value for the matched point and a visibility score that measures if the source point is visible in target frame.

pixel weight as $w_{\mathcal{H}}(x) = 1 + 10 \cdot G_{x_{gt}}(x)$. The heatmap loss is then computed as:

$$\mathcal{L}_{\mathcal{H}} = \sum_i \Phi_{bce}(w_{\mathcal{H}}(\mathcal{H}_{sg} - \mathcal{H}_{gt})) + \lambda_{nll} \sum_i \Phi_{nll}(w_{\mathcal{H}}(\mathcal{H}_{sm} - \mathcal{H}_{gt})) .$$

Here, $\Phi_{bce}(\bullet)$ denotes the binary cross entropy loss and $\Phi_{nll}(\bullet)$ the negative log-likelihood loss, and we empirically determined a weight $\lambda_{nll} = 10$. From these two probability heatmaps, a single one is computed as $\mathcal{H} = \mathcal{H}_{sg} \otimes \mathcal{H}_{sm}$, where \otimes is the Hadamard product.

Depth In addition to heatmap prediction, our network also predicts the matched point’s depth value in the target camera’s coordinate system. Inspired by [25], we predict the depth densely, predicting the same depth value for every pixel in the output image:

$$\mathcal{D} = \exp(\mathbf{D}(D_{out})) .$$

Here, \mathbf{D} is a convolutional layer converting feature space into depth values, and the exponential is applied to guarantee positive depth predictions. Ground truth for depth prediction \mathcal{D}_{gt} is the depth of the ground truth match, repeated for the whole image. Since we want to encourage depth prediction to focus on the matched pixel, we again use pixel weighting, this time in the form of $w_{\mathcal{D}}(x) = G_{x_{gt}}(x)$, setting the center pixel’s weight to 1 and decaying the weights to 0. Using the weighted version of mean squared error $\Phi_{mse}(\bullet)$ we employ the following loss for depth prediction:

$$\mathcal{L}_{\mathcal{D}} = \lambda_d \sum_i \Phi_{mse}(w_{\mathcal{D}}(\mathcal{D} - \mathcal{D}_{gt})) .$$

Visibility Furthermore, we also predict a visibility score $\in [0, 1]$ that measures whether the source point is visible (high value) or occluded (low value) in the target frame:

$$\mathcal{V} = \sigma_{sg}(\mathbf{V}(B_{out})) .$$

Here, B_{out} is the output feature map of the bottleneck layer, \mathbf{V} is a convolutional layer, and σ_{sg} a sigmoid activation. The visibility loss takes the following form:

$$\mathcal{L}_{\mathcal{V}} = \sum_i \Phi_{bce}(\mathcal{V} - \mathcal{V}_{gt}) .$$

In the end, we train the network using a weighted combination of all presented loss functions:

$$\mathcal{L} = \mathcal{L}_{\mathcal{H}} + \lambda_{\mathcal{D}} \mathcal{L}_{\mathcal{D}} + \lambda_{\mathcal{V}} \mathcal{L}_{\mathcal{V}} .$$

In all experiments we use the constant and empirically determined weights $\lambda_{\mathcal{D}} = 100$ and $\lambda_{\mathcal{V}} = 1$. An overview of the network architecture is given in Fig. 3. More network and training details can be found in the supplemental.

4. Non-Rigid Reconstruction Pipeline

We integrate the learned non-rigid matching algorithm into a non-rigid RGB-D reconstruction framework that efficiently tracks dense, space-time coherent, non-rigid deformations on the GPU and also provides an efficient volumetric fusion backend. A canonical model of the scene is reconstructed from data, in parallel to tracking the non-rigid deformations, and stored based on a truncated signed distance field (TSDF) represented by a uniform voxel grid. New observations are fused into the grid based on an exponentially moving average. The non-rigid scene motion is tracked based on the following tracking energy:

$$E_{total}(\mathcal{T}) = E_{data}(\mathcal{T}) + \lambda_{learned} E_{learned}(\mathcal{T}) + \lambda_{reg} E_{reg}(\mathcal{T}) .$$

The weights $\lambda_{\text{learned}} = 1$ and $\lambda_{\text{reg}} = 1$ are empirically determined and balance the different terms.

4.1. Deformation Model

To parameterize scene motion, similar to [41], we employ a coarse deformation graph \mathcal{G} with K deformation nodes $\mathbf{g}_i \in \mathbb{R}^3$. The graph models the deformation of space based on the convex combination of local per-node transformations that are parameterized by rotation parameters $\theta_i \in \mathbb{R}^3$ in Lie algebra space and a translation vector $\mathbf{t}_i \in \mathbb{R}^3$. In total, this leads to $2k$ free variables to describe scene motion that we jointly refer to as \mathcal{T} . This enables us to decouple the number of free variables from the complexity of the reconstructed scene. The deformation nodes are connected based on proximity, for details we refer to the original embedded deformation paper [41].

4.2. Optimization Terms

For data term $E_{\text{data}}(\mathcal{T})$, similar to [28, 17], we employ dense point-to-point and point-to-plane alignment constraints between the input depth map and the current reconstruction. For regularizer E_{reg} , we employ the as-rigid-as-possible (ARAP) constraint [39] to enforce locally rigid motion. In addition, we integrate a sparse feature alignment term based on our learned correspondences (see Sec. 3). For each node \mathbf{g}_i of the current deformation graph, we predict a probability heatmap \mathcal{H}_i that gives the likelihood of its 2D uv-position in the current input depth map, using the initial depth map as the reference frame. Furthermore, we also back-project the pixel with the maximum heatmap response into a 3D point $\mathbf{p}_i \in \mathbb{R}^3$, using its depth. We aim to align the graph nodes with the maximum in the corresponding heatmap using the following alignment constraint:

$$E_{\text{learned}}(\mathcal{T}) = \sum_{\mathbf{g}_i \in \mathcal{G}} (1 - \mathcal{H}_i(\pi(\mathbf{g}_i + \mathbf{t}_i)))^2 + \lambda_{\text{point}} \sum_{\mathbf{g}_i \in \mathcal{G}} (\mathbf{g}_i + \mathbf{t}_i - \mathbf{p}_i)^2 .$$

Here, $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the projection from 3D camera space to 2D screen space. The heatmap \mathcal{H}_i is normalized to a maximum of 1. We empirically set $\lambda_{\text{point}} = 10$. In order to handle outliers, especially in the case of occluded correspondences, we make use of the predicted visibility score and the predicted depth value of the match. We filter out all heatmap correspondences with visibility score < 0.5 . We compare the predicted depth with the queried depth from the target frame’s depth map at the pixel with the maximum heatmap response and invalidate any correspondences with a depth difference > 0.15 meters.

4.3. Energy Optimization

We efficiently tackle the underlying optimization problem using a data-parallel Gauss-Newton solver to find the

deformation graph \mathcal{G}^* that best explains the data:

$$\mathcal{G}^* = \operatorname{argmin} E_{\text{total}}(\mathcal{G}) .$$

In the Gauss-Newton solver, we solve the underlying sequence of linear problems using data-parallel preconditioned conjugate gradient (PCG). For implementation details, we refer to the supplemental document.

5. Semi-supervised Data Acquisition

In the following, we provide the details of our semi-supervised non-rigid data collection process that is used for training the non-rigid matching and for the evaluation of non-rigid reconstruction algorithms. The high-level overview of the data acquisition pipeline is shown in Fig. 1.

5.1. Data Acquisition

In order to obtain RGB-D scans of non-rigidly moving objects, we use a Structure Sensor mounted on an iPad. The depth stream is recorded at a resolution of 640×480 and 30 frames per second; the RGB stream is captured with the iPad camera at a resolution of 1296×968 pixels that is calibrated with respect to the range sensor. Regarding the scanning instructions, we follow the ScanNet [5] pipeline. However, in our case, we focus on scenes with one up to several non-rigidly moving objects in addition to a static background. In total, we recorded 400 scenes with over 390,000 RGB-D frames.

5.2. Data Annotation

We crowd sourced sparse ground truth correspondence annotations and segmentation masks for our novel data set. To this end, we employed a web-based annotation tool. The annotation was divided into two tasks. Firstly, we select up to 10 frames per sequence. All dynamic objects that are found in these frames are given unique instance ids (the same instance in the whole sequence) and their masks are annotated in each frame. To accelerate mask segmentation, we use a hierarchy of superpixels as candidate brush sizes. Secondly, among the annotated frames up to 10 frame pairs are selected, and the sparse correspondences between all dynamic objects are annotated. Expert annotators were instructed to annotate correspondences uniformly over the complete object, labeling about 20 point matches per frame pair. Furthermore, in parts of the source image that are occluded in the target image occlusion points were uniformly selected to collect data samples for visibility detection. The dynamic object segmentation task takes on average about 1 min per frame, while the correspondence labeling task takes on average about 2 min per frame.

5.3. Dense Data Alignment

Using the annotated object masks and the sparse matches, dense non-rigid alignment of the frame pairs is

performed. We follow a similar approach as for non-rigid reconstruction (see Sec. 4), based on the sparse deformation graph of [41]. The deformation graph is defined on the source frame’s depth map, covering only the dynamic object by using the source object mask. The final energy function that is optimized to align the source RGB-D frame to the target RGB-D frame is:

$$E_{\text{total}}(\mathcal{T}) = E_{\text{data}}(\mathcal{T}) + \lambda_{\text{photo}} E_{\text{photo}}(\mathcal{T}) + \lambda_{\text{silh}} E_{\text{silh}}(\mathcal{T}) + \lambda_{\text{sparse}} E_{\text{sparse}}(\mathcal{T}) + \lambda_{\text{reg}} E_{\text{reg}}(\mathcal{T}) .$$

Here, $E_{\text{photo}}(\mathcal{T})$ encourages the color gradient values from the source frame to match the target frame, E_{silh} penalizes deformation of the object outside of the target frame’s object mask, and E_{sparse} enforces annotated sparse matches to be satisfied. The weights $\lambda_{\text{photo}} = 0.001$, $\lambda_{\text{silh}} = 0.0001$, $\lambda_{\text{sparse}} = 100.0$ and $\lambda_{\text{reg}} = 10.0$ have been empirically determined. Details about the different optimization terms and a qualitative comparison of their effects can be found in the supplemental document. In order to cope with simple apparent topology changes that are very common while capturing natural non-rigid motion, such as the hand touching the body in one frame and moving away in another frame, we execute non-rigid alignment in both directions and compute the final non-rigid alignment using forward-backward motion interpolation, similar to [55]. At the end, a quick manual review step is performed, in which, if necessary, any incorrectly aligned mesh parts are removed. The review step takes about 30 seconds per frame. Examples of dense alignment results and the employed review interface can be found in the accompanying video.

6. Experiments

We provide a train-val-test split with 340 sequences in the training set, 30 in the test set, and 30 in the validation set. We made sure that there is no overlap between captured environments between training and validation/test scenes.

6.1. Non-Rigid Matching Evaluation

For a given set of pixels (and corresponding 3D points) in the source image, the task is to find the corresponding pixel (and 3D point) in the target image. We evaluate the average 2D pixel and 3D point error (in meters), and compute the matching accuracy (ratio of matches closer than 20 pixels or 0.05 meters from the ground truth correspondences). We compare our non-rigid matching approach to several hand-crafted feature matching strategies, that are based on depth or color based descriptors, and to the learned 3Dmatch [56] descriptor, see Tab. 1. Specifically, we compare to the hand-crafted geometry descriptors, such as Unique Signatures of Histograms (SHOT) [42] and the Fast Point Feature Histograms (FPFH) [31]. We also compare to color-based descriptors, e.g., SIFT [24] and SURF [1], that can be used to

Method	2D-err	3D-err	2D-acc	3D-acc
SIFT [24]	138.40	0.552	16.20	14.08
SURF [1]	125.72	0.476	22.13	19.82
SHOT [42]	105.34	0.342	13.43	11.51
FPFH [31]	109.49	0.393	10.85	9.43
3Dmatch [56]	68.98	0.273	30.50	25.33
GPC [46]	65.04	0.231	31.93	28.16
FlowNet-2.0 [16]	27.32	0.118	68.68	63.67
Ours-12.5%	78.82	0.268	27.17	23.28
Ours-25.0%	58.28	0.197	40.32	35.81
Ours-50.0%	45.43	0.156	50.70	46.57
Ours-Rigid	57.87	0.270	40.93	35.64
Ours-SelfSupervised	33.34	0.121	60.87	55.70
Ours-Sparse	31.42	0.106	58.53	52.24
Ours-NoWeighting	23.72	0.083	73.13	68.46
Ours	19.56	0.073	77.60	72.48

Table 1: We outperform all baseline matching methods by a considerable margin. 2D/3D errors are average pixel/point errors, and 2D/3D accuracy is the percentage of pixels/points with distance of at most 20 pixels/0.05 meters.

establish a sparse set of matches between RGB-D frames. Finally, we train a learned descriptor from [56], patch-based random forest matcher from [46] and optical flow prediction network from [16] on our training sequences. Our method consistently outperforms all the baselines.

6.2. Non-Rigid Reconstruction Results

We integrated our learned matching strategy into a non-rigid reconstruction pipeline. Our learned data term significantly improves reconstruction quality, both qualitatively and quantitatively. To be able to perform a quantitative comparison on our test sequences, we used our re-implementation of [28], and the code or results provided from the authors of [17, 36, 37, 14]. We also replaced our data-driven correspondence matching module with the descriptor learning network from [56], trained on our data, and used it in combination with 3D Harris keypoints. The quantitative evaluation is shown in Tab. 2. The evaluation metrics measure deformation error (a 3D distance between the annotated and computed correspondence positions) and geometry error (comparing depth values inside the object mask to the reconstructed geometry). Deformation error is the more important metric, since it also measures tangential drift within the surface. To be able to know which dynamic object to reconstruct if multiple are present, we always provide the initial ground truth segmentation mask of the selected object. All approaches in Tab. 2 were evaluated on all 30 test sequences to provide a comparison on different kinds of objects and deformable motions. [14] provided results on two challenging test sequences, their average deformation and geometry error are 21.05 cm and 14.87 cm respectively, while our approach achieves average errors of 3.63 cm and 0.48 cm. Our approach outperforms the state of the art by a large margin. The methods [36] and [37] do not

Method	Def. error (cm)	Geo. error (cm)
DynamicFusion re-impl. [28]	6.31	1.08
VolumeDeform [17]	21.27	7.78
DynamicFusion + 3Dmatch	6.64	1.59
Ours-Rigid	12.21	2.30
Ours-Sparse	8.24	0.77
Ours-SelfSupervised	5.47	0.54
Ours-Base	3.94	0.43
Ours-Occlusion	3.70	0.42
Ours-Occlusion+Depth	3.28	0.41

Table 2: Comparison with state-of-the-art approaches. Our learned correspondences significantly improve both tracking and reconstruction quality. We also provide ablation studies on training data type and different network parts.

compute explicit point correspondences from the canonical frame to other frames, so we could not evaluate these approaches quantitatively; we provide qualitative comparison on our sequences in the supplemental document. We also show qualitative comparisons with our re-implementation of DynamicFusion [28] in Fig. 4 and with the state-of-the-art approach of [14] in Fig. 5. Our learned correspondences enable us to handle faster object motion as well as challenging planar motion, where even photometric cues fail, for instance due to uniform object color.

6.3. Ablation Study

We evaluated different components of our network and their effect on the reconstruction quality, see Tab. 2. Since some sequences include motions in which large parts of the reconstructed object are occluded, as can be observed in Fig. 6, using visibility detection for correspondence pruning makes our method more robust. Furthermore, since depth measurements and heatmap predictions can both be sometimes noisy, adding correspondence filtering with depth prediction further improves the reconstruction results.

6.4. Data Generation Evaluation

To show the importance of our semi-supervised strategy for constructing the training corpus, we evaluate how different training corpora influence the performance of data-driven reconstruction methods. Aside from our training data, which has been generated using dense semi-supervised frame alignment, we used a publicly available rigid dataset of indoor scenes (from [5]), self-supervised alignment of our sequences (as in [33]), and only manually annotated sparse samples from our dataset. We provide comparison on both non-rigid matching in Tab. 1 and non-rigid reconstruction in Tab. 2. Using only rigid data does not generalize to non-rigid sequences. While sparse matches already improve network performance, there is not enough data for reliable correspondence prediction on every part of the observed object. In addition, annotated sparse matches are usually matches on image parts that are easy for hu-

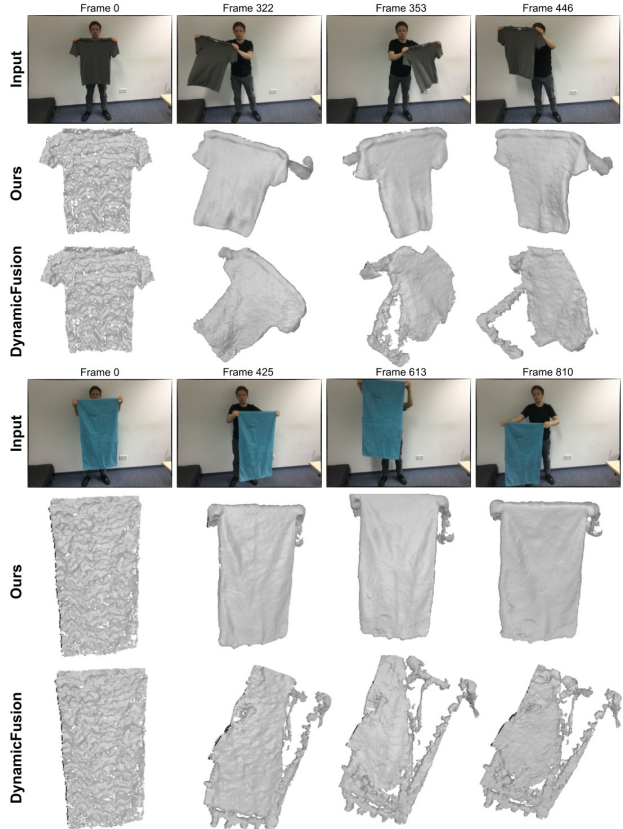


Figure 4: Qualitative comparison to DynamicFusion (reimplementation).

mans to match, and there are much less accurate correspondences in areas with uniform color. In the self-supervised setting, the network gets dense correspondence information, which improves the method’s reconstruction performance compared to using only sparse features. However, without semi-supervised densely aligned frame pairs, we can only generate matches for the simple deformations, where the DynamicFusion approach can successfully track the motion. Therefore, the performance of the network trained only on self-supervised data degrades considerably on more extreme deformations, as can be seen in Fig. 7. Dense alignment of too far away frames is needed for accurate network prediction also in the case of extreme deformations. Since the majority of the dense aligned matches is still moving rigidly, it turned out to be beneficial to sample more deformable samples during training. In order to estimate which parts of the scene are more deformable, we employed sparsely annotated matches and ran RANSAC in combination with a Procrustes algorithm to estimate the average rigid pose. The more the motion of each sampled match differs from the average rigid motion, the more often we sample it during network training using a multinomial distribution of the non-rigid displacement weights. This

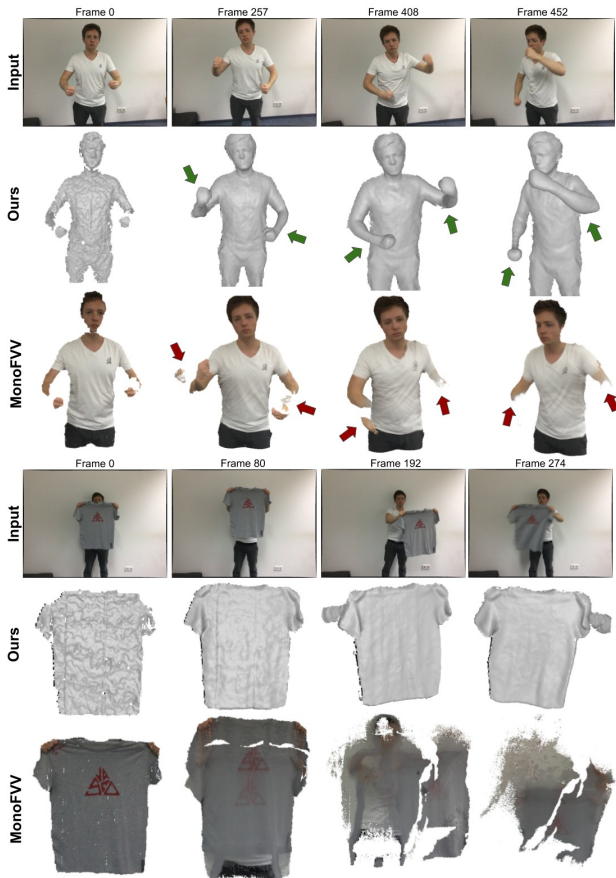


Figure 5: Qualitative comparison of reconstruction results of our method and MonoFV [14]. Reconstruction results were kindly provided by the authors.

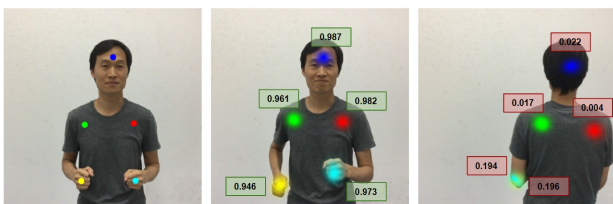


Figure 6: Visibility detection enables filtering of network correspondences that are occluded. The visibility score is in the range $[0, 1]$, it is high for visible correspondences and low for occluded parts. We filter out all correspondences with visibility score less than 0.50.

strategy improved the network performance, as is shown in Tab. 1, compared to training on non-weighted samples. Finally, we demonstrate how much data is needed to achieve robust correspondence prediction performance; i.e., using less training data considerably degrades matching accuracy, as summarized in Tab. 1, where we trained networks using only 12.5%, 25.0%, and 50.0% of the training data.



Figure 7: Comparison of correspondence prediction for reference frame (left) using self-supervised training data (middle) and semi-supervised dense data (right).

6.5. Limitations

While learned correspondences make tracking of fast motion more robust, there is still room for improvement when reconstructing dynamic objects. One pressing issue is that background clutter might be accidentally fused with the object when the object is close to the background. In this case, the reconstructed shape would slowly grow and we might also start reconstructing the background. This can cause wrong deformation graph connectivity and lead to tracking failures. A potential future avenue is to subtract and ignore the background; e.g., we could use our annotated object masks to develop a data-driven method.

7. Conclusion

We have proposed a neural network architecture for matching correspondences in non-rigid sequences that operates on RGB-D frames and demonstrated that our learned descriptors outperform existing hand-crafted ones. In addition, we introduced the first large-scale dataset that is composed of 400 scenes, over 390,000 RGB-D frames, and 5,533 densely aligned frame pairs. The dataset is obtained with a semi-supervised strategy by combining self-supervision with sparse annotations to obtain dense inter-frame correspondences. We also provide a test set along with several metrics for evaluating non-rigid matching and non-rigid reconstruction. We believe that our dataset is a first step towards enabling learning-based non-rigid matching and our benchmark will help to quantitatively and objectively compare different approaches.

Acknowledgements


We would like to thank the expert annotators Sathya Ashok, Omar Heydayat, Haya Irfan, Azza Jenane, Soh Yee Lee, Suzana Spasova, and Weile Weng for their efforts in building the dataset. We further thank Edgar Tretschk for his help with data collection, Xiaochen Fan for his valuable contributions during his research visit, and Armen Avetisyan for numerous helpful discussions. This work was supported by the Max Planck Center for Visual Computing and Communications (MPC-VCC), a TUM-IAS Rudolf Mößbauer Fellowship, the ERC Starting Grant *Scan2CAD* (804724), the ERC Consolidator Grant *4DRepLy* (770784), and the German Research Foundation (DFG) Grant *Making Machine Learning on Static and Dynamic 3D Data Practical*.

References

- [1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008. [3](#), [6](#)
- [2] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. Scalable real-time volumetric surface reconstruction. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 32(4):113:1–113:16, July 2013. [2](#)
- [3] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 5556–5565, 2015. [2](#)
- [4] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. Comp. Graph. & Interact. Techn.*, pages 303–312, 1996. [2](#)
- [5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. [3](#), [5](#), [7](#)
- [6] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. *ACM Trans. on Graphics*, 36(3), 2017. [2](#)
- [7] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, 27(3):98:1–98:10, Aug. 2008. [3](#)
- [8] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)*, 36(6):246, 2017. [1](#), [3](#)
- [9] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):114, 2016. [1](#), [3](#)
- [10] Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 3d scanning deformable objects with a single rgb-d sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 493–501, 2015. [2](#)
- [11] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. volume 3, pages 224–237, 05 2004. [3](#)
- [12] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. 3d-coded : 3d correspondences by deep deformation. In *ECCV*, 2018. [3](#)
- [13] Kaiwen Guo, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai. Robust non-rigid motion tracking and surface reconstruction using l0 regularization. *IEEE Transaction on Visualization and Computer Graphics (TVCG)*, 2017. [3](#)
- [14] Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Transactions on Graphics (TOG)*, 36(3):32, 2017. [1](#), [3](#), [6](#), [7](#), [8](#)
- [15] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3279–3286, June 2015. [3](#)
- [16] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. [6](#)
- [17] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016. [1](#), [3](#), [5](#), [6](#), [7](#)
- [18] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. ACM Symp. User Interface Softw. & Tech.*, pages 559–568, 2011. [2](#)
- [19] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, May 1999. [3](#)
- [20] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In *Proc. Int. Conf. 3D Vision (3DV)*, pages 1–8, Washington, DC, USA, 2013. IEEE Computer Society. [2](#)
- [21] B. G. Vijay Kumar, Gustavo Carneiro, and Ian D. Reid. Learning local image descriptors with deep siamese and triplet convolutional networks by minimizing global loss functions. *CoRR*, abs/1512.09272, 2015. [3](#)
- [22] D. Lefloch, M. Kluge, H. Sarbolandi, T. Weyrich, and A. Kolb. Comprehensive use of curvature for robust and accurate online surface reconstruction. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, page 10.1109/TPAMI.2017.2648803, 2017. [2](#)
- [23] Damien Lefloch, Tim Weyrich, and Andreas Kolb. Anisotropic point-based fusion. In *Proc. Int. Conf. Information Fusion (FUSION)*, pages 1–9, July 2015. [2](#)
- [24] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. [3](#), [6](#)
- [25] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):44, 2017. [4](#)
- [26] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgb-d images. In *ECCV*, 2012. [3](#)

- [27] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. KinectFusion: real-time dense surface mapping and tracking. In *Proc. IEEE Int. Symp. Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011. [2](#)
- [28] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015. [1](#), [2](#), [3](#), [5](#), [6](#), [7](#)
- [29] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans. on Graphics*, 32(6):169, 2013. [2](#)
- [30] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Int. Conf. 3D Digital Imaging and Modeling (3DIM)*, pages 145–152. IEEE, 2001. [2](#)
- [31] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation, ICRA'09*, pages 1848–1853, Piscataway, NJ, USA, 2009. IEEE Press. [3](#), [6](#)
- [32] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391, Sep. 2008. [3](#)
- [33] Tanner Schmidt, Richard A. Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2017. [2](#), [3](#), [7](#)
- [34] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, pages 118–126. IEEE Computer Society, 2015. [3](#)
- [35] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1573–1585, Aug 2014. [3](#)
- [36] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, page 7, 2017. [1](#), [3](#), [6](#)
- [37] Miroslava Slavcheva, Maximilian Baust, and Slobodan Ilic. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2646–2655, 2018. [1](#), [3](#), [6](#)
- [38] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. [3](#)
- [39] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP '07*, pages 109–116, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. [5](#)
- [40] Frank Steinbrucker, Christian Kerl, and Daniel Cremers. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3264–3271, 2013. [2](#)
- [41] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 Papers, SIGGRAPH '07*, New York, NY, USA, 2007. ACM. [5](#), [6](#)
- [42] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III, ECCV'10*, pages 356–369, Berlin, Heidelberg, 2010. Springer-Verlag. [3](#), [6](#)
- [43] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 Papers, SIGGRAPH '08*, pages 97:1–97:9, New York, NY, USA, 2008. ACM. [3](#)
- [44] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics (TOG)*, 28(2):15, 2009. [2](#)
- [45] Ruizhe Wang, Lingyu Wei, Etienne Vouga, Qi-Xing Huang, Duygu Ceylan, Gérard G. Medioni, and Hao Li. Capturing dynamic textured surfaces of moving targets. *CoRR*, abs/1604.02801, 2016. [3](#)
- [46] Shenlong Wang, Sean Ryan Fanello, Christoph Rhemann, Shahram Izadi, and Pushmeet Kohli. The global patch collider. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 127–135, 2016. [6](#)
- [47] Sen Wang, Xinxin Zuo, Chao Du, Runxiao Wang, Jiangbin Zheng, and Ruigang Yang. Dynamic non-rigid objects reconstruction with a single rgb-d sensor. *Sensors*, 18(3):886, 2018. [1](#), [3](#)
- [48] Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. [1](#), [3](#)
- [49] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard, and John McDonald. Kintinuous: Spatially extended kinectfusion. In *Proc. RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012. [2](#)
- [50] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense SLAM without a pose graph. In *Proceedings of Robotics: Science and Systems*, 2015. [2](#)
- [51] Chenglei Wu, Carsten Stoll, Levi Valgaerts, and Christian Theobalt. On-set performance capture of multiple actors with a stereo camera. *ACM Trans. Graph.*, 32(6):161:1–161:11, 2013. [2](#)
- [52] Genzhi Ye, Yebin Liu, Nils Hasler, Xiangyang Ji, Qionghai Dai, and Christian Theobalt. Performance Capture of Interacting Characters with Handheld Kinects. In *Proc. ECCV*, pages 828–841. Springer, 2012. [3](#)

- [53] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. *CoRR*, abs/1603.09114, 2016. [3](#)
- [54] Tao Yu, Zherong Zheng, Kaiwen Guo, Jianhui Zhao, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. DoubleFusion: Real-time Capture of Human Performances with Inner Body Shapes from a Single Depth Sensor. In *31st IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018. [2](#)
- [55] Konstantinos Zampogiannis, Cornelia Fermuller, and Yianis Aloimonos. Topology-aware non-rigid point cloud registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. [6](#)
- [56] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017. [3](#), [6](#)
- [57] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Trans. on Graphics*, 32(4):112, 2013. [2](#)
- [58] Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. Elastic fragments for dense scene reconstruction. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 473–480, 2013. [2](#)
- [59] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. State of the Art on 3D Reconstruction with RGB-D Cameras. *Computer Graphics Forum (Eurographics State of the Art Reports 2018)*, 37(2), 2018. [2](#)
- [60] Jure Žbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, volume 07-12-June-2015, pages 1592–1599. IEEE Computer Society, 10 2015. [3](#)



DeepDeform: Learning Non-Rigid RGB-D Reconstruction With Semi-Supervised Data

Conference Proceedings: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Author: Aljaž Božič

Publisher: IEEE

Date: June 2020

Copyright © 2020, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure C.1: Copyright authorization. Authorization for reuse of paper *DeepDeform: Learning Non-rigid RGB-D Reconstruction With Semi-Supervised Data* in this dissertation.

152

C. Original Publications

C.2 Neural Non-Rigid Tracking

COPYRIGHT

©2020 NeurIPS. Reprinted, with permission, from
 Aljaž Božič*, Pablo Palafox*, Michael Zollhöfer, Angela Dai, Justus Thies,
 and Matthias Nießner

Neural Non-Rigid Tracking

Conference on Neural Information Processing Systems (NeurIPS) 2020

Details: NeurIPS website

SUMMARY

We introduce a novel, end-to-end learnable, differentiable non-rigid tracker that enables state-of-the-art non-rigid reconstruction by a learned robust optimization. Given two input RGB-D frames of a non-rigidly moving object, we employ a convolutional neural network to predict dense correspondences and their confidences. These correspondences are used as constraints in an as-rigid-as-possible (ARAP) optimization problem. By enabling gradient back-propagation through the weighted non-linear least squares solver, we are able to learn correspondences and confidences in an end-to-end manner such that they are optimal for the task of non-rigid tracking. Under this formulation, correspondence confidences can be learned via self-supervision, informing a learned robust optimization, where outliers and wrong correspondences are automatically down-weighted to enable effective tracking. Compared to state-of-the-art approaches, our algorithm shows improved reconstruction performance, while simultaneously achieving $85\times$ faster correspondence prediction than comparable deep-learning based methods.

INDIVIDUAL CONTRIBUTIONS

Sharing the leading role in realizing the scientific project.

Problem definition	<i>significantly contributed</i>
Literature survey	<i>significantly contributed</i>
Implementation	<i>contributed</i>
Experimental evaluation	<i>contributed</i>
Preparation of the manuscript	<i>significantly contributed</i>

We include the accepted version of the original publication [14] in the following. Authors do not transfer the copyright of their papers to NeurIPS.

Neural Non-Rigid Tracking

Aljaž Božič^{1*}
aljaz.bozic@tum.de

Pablo Palafox^{1*}
pablo.palafox@tum.de

Michael Zollhöfer²

Angela Dai¹

Justus Thies¹

Matthias Nießner¹

¹Technical University of Munich

²Stanford University

Abstract

We introduce a novel, end-to-end learnable, differentiable non-rigid tracker that enables state-of-the-art non-rigid reconstruction by a learned robust optimization. Given two input RGB-D frames of a non-rigidly moving object, we employ a convolutional neural network to predict dense correspondences and their confidences. These correspondences are used as constraints in an as-rigid-as-possible (ARAP) optimization problem. By enabling gradient back-propagation through the weighted non-linear least squares solver, we are able to learn correspondences and confidences in an end-to-end manner such that they are optimal for the task of non-rigid tracking. Under this formulation, correspondence confidences can be learned via self-supervision, informing a learned robust optimization, where outliers and wrong correspondences are automatically down-weighted to enable effective tracking. Compared to state-of-the-art approaches, our algorithm shows improved reconstruction performance, while simultaneously achieving $85\times$ faster correspondence prediction than comparable deep-learning based methods. We make our code available at <https://github.com/DeformableFriends/NeuralTracking>.

1 Introduction

The capture and reconstruction of real-world environments is a core problem in computer vision, enabling numerous VR/AR applications. While there has been significant progress in reconstructing static scenes, tracking and reconstruction of dynamic objects remains a challenge. Non-rigid reconstruction focuses on dynamic objects, without assuming any explicit shape priors, such as human or face parametric models. Commodity RGB-D sensors, such as Microsoft’s Kinect or Intel’s Realsense, provide a cost-effective way to acquire both color and depth video of dynamic motion. Using a large number of RGB-D sensors can lead to an accurate non-rigid reconstruction, as shown by Dou et al. [8]. Our work focuses on non-rigid reconstruction from a single RGB-D camera, thus eliminating the need for specialized multi-camera setups.

The seminal DynamicFusion by Newcombe et al. [23] introduced a non-rigid tracking and mapping pipeline that uses depth input for real-time non-rigid reconstruction from a single RGB-D camera. Various approaches have expanded upon this framework by incorporating sparse color correspondences [13] or dense photometric optimization [10]. DeepDeform [4] presented a learned correspondence prediction, enabling significantly more robust tracking of fast motion and re-localization. Unfortunately, the computational cost of the correspondence prediction network (~ 2 seconds per frame for a relatively small number of non-rigid correspondences) inhibits real-time performance.

*Denotes equal contribution.

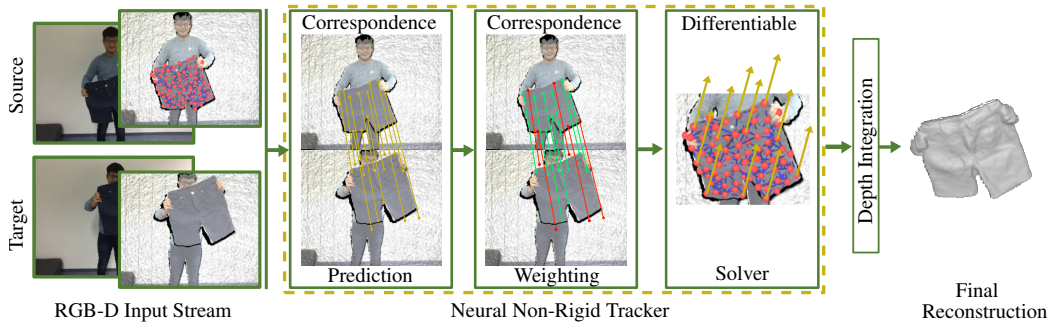


Figure 1: Neural Non-Rigid Tracking: based on RGB-D input data of a source and a target frame, our learned non-rigid tracker estimates the non-rigid deformations to align the source to the target frame. We propose an end-to-end approach, enabling correspondences and their importance weights to be informed by the non-rigid solver. Similar to robust optimization, this provides robust tracking, and the resulting deformation field can then be used to integrate the depth observations in a canonical volumetric 3D grid that implicitly represents the surface of the object (final reconstruction).

Simultaneously, work on learned optical flow has shown dense correspondence prediction at real-time rates [30]. However, directly replacing the non-rigid correspondence predictions from Božič et al. [4] with these optical flow predictions does not produce accurate enough correspondences for comparable non-rigid reconstruction performance. In our work, we propose a neural non-rigid tracker, i.e., an end-to-end differentiable non-rigid tracking pipeline which combines the advantages of classical deformation-graph-based reconstruction pipelines [23, 13] with novel learned components. Our end-to-end approach enables learning outlier rejection in a self-supervised manner, which guides a robust optimization to mitigate the effect of inaccurate correspondences or major occlusions present in single RGB-D camera scenarios.

Specifically, we cast the non-rigid tracking problem as an as-rigid-as-possible (ARAP) optimization problem, defined on correspondences between points in a source and a target frame. A differentiable Gauss-Newton solver allows us to obtain gradients that enable training a neural network to predict an importance weight for every correspondence in a completely self-supervised manner, similar to robust optimization. The end-to-end training significantly improves non-rigid tracking performance. Using our neural tracker in a non-rigid reconstruction framework results in $85\times$ faster correspondence prediction and improved reconstruction performance compared to the state of the art.

In summary, we propose a novel neural non-rigid tracking approach with two key contributions:

- an end-to-end differentiable Gauss-Newton solver, which provides gradients to better inform a correspondence prediction network used for non-rigid tracking of two frames;
- a self-supervised approach for learned correspondence weighting, which is informed by our differentiable solver and enables efficient, robust outlier rejection, thus, improving non-rigid reconstruction performance compared to the state of the art.

2 Related Work

Non-rigid Reconstruction. Reconstruction of deformable objects using a single RGB-D camera is an important research area in computer vision. State-of-the-art methods rely on deformation graphs [29, 35] that enable robust and temporally consistent 3D motion estimation. While earlier approaches required an object template, such graph-based tracking has been extended to simultaneous tracking and reconstruction approaches [7, 23]. These works used depth fitting optimization objectives in the form of iterative closest points, or continuous depth fitting in [26, 27]. Rather than relying solely on depth information, recent works have incorporated SIFT features [13], dense photometric fitting [10], and sparse learned correspondence [4].

Correspondence Prediction for Non-rigid Tracking. In non-rigid tracking, correspondences must be established between the two frames we want to align. While methods such as DynamicFusion [23] rely on projective correspondences, recent methods leverage learned correspondences [4]. DeepDeform [4] relies on sparse predicted correspondences, trained on an annotated dataset of deforming objects. Since prediction is done independently for each correspondence, this results in a high compute cost, compared to dense predictions of state-of-the-art optical flow networks. Optical flow [6, 12, 30, 18] and scene flow [21, 3, 19, 33] methods achieve promising results in predicting dense correspondences between two frames, with some approaches not even requiring direct supervision [32, 16, 15]. In our proposed neural non-rigid tracking approach, we build upon PWC-Net [30] for dense correspondence prediction to inform our non-rigid deformation energy formulation. Since our approach allows for end-to-end training, our 2D correspondence prediction finds correspondences better suited for non-rigid tracking.

Differentiable Optimization. Differentiable optimizers have been explored for various tasks, including image alignment [5], rigid pose estimation [11, 20], multi-frame direct bundle-adjustment [31], and rigid scan-to-CAD alignment [1]. In addition to achieving higher accuracy, an end-to-end differentiable optimization approach also offers the possibility to optimize run-time, as demonstrated by learning efficient pre-conditioning methods in [9, 25, 17]. Unlike Li et al. [17], which employs an image-based tracker (with descriptors defined on nodes in a pixel-aligned graph), our approach works on general graphs and learns to robustify correspondence prediction for non-rigid tracking by learning self-supervised correspondence confidences.

3 Non-Rigid Reconstruction Notation

Non-rigid alignment is a crucial part of non-rigid reconstruction pipelines. In the single RGB-D camera setup, we are given a pair of source and target RGB-D frames $\{(\mathcal{I}_s, \mathcal{P}_s), (\mathcal{I}_t, \mathcal{P}_t)\}$, where $\mathcal{I}_* \in \mathbb{R}^{H \times W \times 3}$ is an RGB image and $\mathcal{P}_* \in \mathbb{R}^{H \times W \times 3}$ a 3D point image. The goal is to estimate a warp field $\mathcal{Q} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ that transforms \mathcal{P}_s into the target frame. Note that we define the 3D point image \mathcal{P}_s as the result of back-projecting every pixel $\mathbf{u} \in \Pi_s \subset \mathbb{R}^2$ into the camera coordinate system with given camera intrinsic parameters. To this end, we define the inverse of the perspective projection to back-project a pixel \mathbf{u} given the pixel’s depth $d_{\mathbf{u}}$ and the intrinsic camera parameters \mathbf{c} :

$$\pi_{\mathbf{c}}^{-1} : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^3, \quad (\mathbf{u}, d_{\mathbf{u}}) \mapsto \pi_{\mathbf{c}}^{-1}(\mathbf{u}, d_{\mathbf{u}}) = \mathbf{p}. \quad (1)$$

To maintain robustness against noise in the depth maps, state-of-the-art approaches define an embedded deformation graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ over the *source* RGB-D frame, where \mathcal{V} is the set of graph nodes defined by their 3D coordinates $\mathbf{v}_i \in \mathbb{R}^3$ and \mathcal{E} the set of edges between nodes, as described in [29] and illustrated in Fig. 1. Thus, for every node in \mathcal{G} , a global translation vector $\mathbf{t}_{\mathbf{v}_i} \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R}_{\mathbf{v}_i} \in \mathbb{R}^{3 \times 3}$, must be estimated in the alignment process. We parameterize rotations with a 3-dimensional axis-angle vector $\boldsymbol{\omega} \in \mathbb{R}^3$. We use the exponential map $\exp : \mathfrak{so}(3) \rightarrow \text{SO}(3)$, $\hat{\boldsymbol{\omega}} \mapsto e^{\hat{\boldsymbol{\omega}}} = \mathbf{R}$ to convert from axis-angle to matrix rotation form, where the $\hat{\cdot}$ -operator creates a 3×3 skew-symmetric matrix from a 3-dimensional vector. The resulting graph motion is denoted by $\mathcal{T} = (\boldsymbol{\omega}_{\mathbf{v}_1}, \mathbf{t}_{\mathbf{v}_1}, \dots, \boldsymbol{\omega}_{\mathbf{v}_N}, \mathbf{t}_{\mathbf{v}_N}) \in \mathbb{R}^{N \times 6}$ for a graph with N nodes.

Dense motion can then be computed by interpolating the nodes’ motion \mathcal{T} by means of a warping function \mathcal{Q} . When applied to a 3D point $\mathbf{p} \in \mathbb{R}^3$, it produces the point’s deformed position

$$\mathcal{Q}(\mathbf{p}, \mathcal{T}) = \sum_{\mathbf{v}_i \in \mathcal{V}} \alpha_{\mathbf{v}_i} (e^{\hat{\boldsymbol{\omega}}_{\mathbf{v}_i}} (\mathbf{p} - \mathbf{v}_i) + \mathbf{v}_i + \mathbf{t}_{\mathbf{v}_i}). \quad (2)$$

The weights $\alpha_{\mathbf{v}_i} \in \mathbb{R}$, also known as *skinning* weights, measure the influence of each node on the current point \mathbf{p} and are computed as in [34]. Please see the supplemental material for further detail.

4 Neural Non-rigid Tracking

Given a pair of source and target RGB-D frames $(\mathcal{Z}_s, \mathcal{Z}_t)$, where $\mathcal{Z}_* = (\mathcal{I}_* | \mathcal{P}_*) \in \mathbb{R}^{H \times W \times 6}$ is the concatenation of an RGB and a 3D point image as defined in Section 3, we aim to find a function Θ that estimates the motion \mathcal{T} of a deformation graph \mathcal{G} with N nodes (given by their 3D coordinates \mathcal{V}) defined over the source RGB-D frame. This implicitly defines source-to-target dense 3D motion (see Figure 2). Formally, we have:

$$\Theta : \mathbb{R}^{H \times W \times 6} \times \mathbb{R}^{H \times W \times 6} \times \mathbb{R}^{N \times 3} \rightarrow \mathbb{R}^{N \times 6}, \quad (\mathcal{Z}_s, \mathcal{Z}_t, \mathcal{V}) \mapsto \Theta(\mathcal{Z}_s, \mathcal{Z}_t, \mathcal{V}) = \mathcal{T}. \quad (3)$$

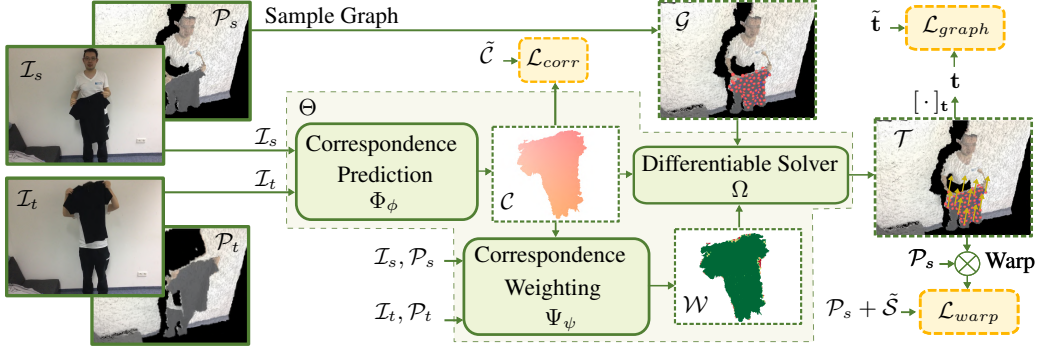


Figure 2: Overview of our neural non-rigid tracker. Given a pair of source and target images, \mathcal{I}_s and \mathcal{I}_t , a dense correspondence map \mathcal{C} between the frames is estimated via a convolutional neural network Φ . Importance weights \mathcal{W} for these correspondences are computed through a function Ψ . Together with a graph \mathcal{G} defined over the source RGB-D frame \mathcal{P}_s , both \mathcal{C} and \mathcal{W} are input to a differentiable solver Ω . The solver outputs the graph motion \mathcal{T} , i.e., the non-rigid alignment between source and target frames. Our approach is optimized end-to-end, with losses on the final alignment using $\mathcal{L}_{\text{graph}}$ and $\mathcal{L}_{\text{warp}}$, and an intermediate loss on the correspondence map $\mathcal{L}_{\text{corr}}$.

To estimate \mathcal{T} , we first establish dense 2D correspondences between the source and target frame using a deep neural network Φ . These correspondences, denoted as \mathcal{C} , are used to construct the data term in our non-rigid alignment optimization. Since the presence of outlier correspondence predictions has a strong negative impact on the performance of non-rigid tracking, we introduce a weighting function Ψ , inspired by robust optimization, to down-weight inaccurate predictions. Function Ψ outputs importance weights \mathcal{W} and is learned in a self-supervised manner. Finally, both correspondence predictions \mathcal{C} and importance weights \mathcal{W} are input to a differentiable, non-rigid alignment optimization module Ω . By optimizing the non-rigid alignment energy (see Section 4.3), the differentiable optimizer Ω estimates the deformation graph parameters \mathcal{T} that define the motion from source to target frame:

$$\mathcal{T} = \Theta(\mathcal{Z}_s, \mathcal{Z}_t, \mathcal{V}) = \Omega(\Phi(\cdot), \Psi(\cdot), \mathcal{V}) = \Omega(\mathcal{C}, \mathcal{W}, \mathcal{V}). \quad (4)$$

In the following, we define the dense correspondence predictor Φ , the importance weighting Ψ and the optimizer Ω , and describe a fully differentiable approach for optimizing Φ and Ψ such that we can estimate dense correspondences with importance weights best suited for non-rigid tracking.

4.1 Dense Correspondence Prediction

The dense correspondence prediction function Φ takes as input a pair of source and target RGB images $(\mathcal{I}_s, \mathcal{I}_t)$, and for each source pixel location $\mathbf{u} \in \Pi_s \subset \mathbb{R}^2$ it outputs a corresponding pixel location in the target image \mathcal{I}_t , which we denote by $\mathbf{c}_{\mathbf{u}} \in \Pi_t \subset \mathbb{R}^2$. Formally, Φ is defined as

$$\Phi: \mathbb{R}^{H \times W \times 3} \times \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{H \times W \times 2}, \quad (\mathcal{I}_s, \mathcal{I}_t) \mapsto \Phi(\mathcal{I}_s, \mathcal{I}_t) = \mathcal{C}, \quad (5)$$

where \mathcal{C} is the resulting dense correspondence map. The function Φ is represented by a deep neural network that leverages the architecture of a state-of-the-art optical flow estimator [30].

4.2 Correspondence Importance Weights

For each source pixel $\mathbf{u} \in \Pi_s \subset \mathbb{R}^2$ and its correspondence $\mathbf{c}_{\mathbf{u}} \in \Pi_t \subset \mathbb{R}^2$, we additionally predict an importance weight $w_{\mathbf{u}} \in (0, 1)$ by means of the weighting function Ψ . The latter takes as input the source RGB-D image \mathcal{Z}_s , the corresponding sampled target frame values \mathcal{Z}'_t , and intermediate features from the correspondence network Φ , and outputs weights for the correspondences between source and target. Note that \mathcal{Z}'_t is the result of bilinearly sampling [14] the target image \mathcal{Z}_t at the predicted correspondence locations \mathcal{C} . The last layer of features \mathcal{H} of the correspondence network Φ , with dimension $D = 565$, are used to inform Ψ . The weighting function is thus defined as

$$\Psi: \mathbb{R}^{H \times W \times 6} \times \mathbb{R}^{H \times W \times 6} \times \mathbb{R}^{H \times W \times D} \rightarrow \mathbb{R}^{H \times W \times 1}, \quad (\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H}) \mapsto \Psi(\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H}) = \mathcal{W}. \quad (6)$$

4.3 Differentiable Optimizer

We introduce a differentiable optimizer Ω to estimate the deformation graph parameters \mathcal{T} , given the correspondence map \mathcal{C} , importance weights \mathcal{W} , and N graph nodes \mathcal{V} :

$$\Omega : \mathbb{R}^{H \times W \times 2} \times \mathbb{R}^{H \times W \times 1} \times \mathbb{R}^{N \times 3} \rightarrow \mathbb{R}^{N \times 6}, \quad (\mathcal{C}, \mathcal{W}, \mathcal{V}) \mapsto \Omega(\mathcal{C}, \mathcal{W}, \mathcal{V}) = \mathcal{T}, \quad (7)$$

with \mathcal{C} and \mathcal{W} estimated by functions Φ (Eq. 5) and Ψ (Eq. 6), respectively. Using the predicted dense correspondence map \mathcal{C} , we establish the data term for the non-rigid tracking optimization. Specifically, we use a 2D data term that operates in image space and a depth data term that leverages the depth information of the input frames. In addition to the data terms, we employ an As-Rigid-As-Possible regularizer [28] to encourage node deformations to be locally rigid, enabling robust deformation estimates even in the presence of noisy input cues. Note that the resulting optimizer module Ω is fully differentiable, but contains no learnable parameters. In summary, we formulate non-rigid tracking as the following nonlinear optimization problem:

$$\arg \min_{\mathcal{T}} (\lambda_{2D} E_{2D}(\mathcal{T}) + \lambda_{\text{depth}} E_{\text{depth}}(\mathcal{T}) + \lambda_{\text{reg}} E_{\text{reg}}(\mathcal{T})). \quad (8)$$

2D reprojection term. Given the outputs of the dense correspondence predictor and weighting function, $\Phi(\mathcal{I}_s, \mathcal{I}_t)$ and $\Psi(\mathcal{Z}_s, \mathcal{Z}_t', \mathcal{H})$, respectively, we query for every pixel \mathbf{u} in the source frame its correspondence $\mathbf{c}_{\mathbf{u}}$ and weight $w_{\mathbf{u}}$ to build the following energy term:

$$E_{2D}(\mathcal{T}) = \sum_{\mathbf{u} \in \Pi_s} w_{\mathbf{u}}^2 \|\pi_{\mathbf{c}}(\mathbf{Q}(\mathbf{p}_{\mathbf{u}}, \mathcal{T})) - \mathbf{c}_{\mathbf{u}}\|_2^2, \quad (9)$$

where $\pi_{\mathbf{c}} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$, $\mathbf{p} \mapsto \pi_{\mathbf{c}}(\mathbf{p})$ is a perspective projection with intrinsic parameters \mathbf{c} and $\mathbf{p}_{\mathbf{u}} = \pi_{\mathbf{c}}^{-1}(\mathbf{u}, d_{\mathbf{u}})$ as defined in Eq. 1. Each pixel is back-projected to 3D, deformed using the current graph motion estimate as described in Eq. 2 and projected onto the target image plane. The projected deformed location is compared to the predicted correspondence $\mathbf{c}_{\mathbf{u}}$.

Depth term. The depth term leverages the depth cues of the source and target images. Specifically, it compares the z components of a warped source point, i.e., $[\mathbf{Q}(\mathbf{p}_{\mathbf{u}}, \mathcal{T})]_z$, and a target point sampled at the corresponding location $\mathbf{c}_{\mathbf{u}}$ using bilinear interpolation:

$$E_{\text{depth}}(\mathcal{T}) = \sum_{\mathbf{u} \in \Pi_s} w_{\mathbf{u}}^2 ([\mathbf{Q}(\mathbf{p}_{\mathbf{u}}, \mathcal{T})]_z - [P_t(\mathbf{c}_{\mathbf{u}})]_z)^2. \quad (10)$$

Regularization term. We encourage the deformation of neighboring nodes in the deformation graph to be locally rigid. Each node $\mathbf{v}_i \in \mathcal{V}$ has at most $K = 8$ neighbors in the set of edges \mathcal{E} , computed as nearest nodes using geodesic distances. The regularization term follows [28]:

$$E_{\text{reg}}(\mathcal{T}) = \sum_{(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E}} \left\| e^{\hat{\omega}_{\mathbf{v}_i}} (\mathbf{v}_j - \mathbf{v}_i) + \mathbf{v}_i + \mathbf{t}_{\mathbf{v}_i} - (\mathbf{v}_j + \mathbf{t}_{\mathbf{v}_j}) \right\|_2^2. \quad (11)$$

Equation 8 is minimized using the Gauss-Newton algorithm, as described in Algorithm 1. In the following, we denote the number of correspondences by $|\mathcal{C}|$ and the number of graph edges by $|\mathcal{E}|$. Moreover, we transform all energy terms into a residual vector $\mathbf{r} \in \mathbb{R}^{3|\mathcal{C}|+3|\mathcal{E}|}$. For every graph node, we compute partial derivatives with respect to translation and rotation parameters, constructing a Jacobian matrix $\mathbf{J} \in \mathbb{R}^{(3|\mathcal{C}|+3|\mathcal{E}|) \times 6N}$, where N is the number of nodes in the set of vertices \mathcal{V} . Analytic formulas for partial derivatives are described in the supplemental material.

Initially, the deformation parameters are initialized to $\mathcal{T}_0 = \mathbf{0}$, corresponding to zero translation and identity rotations. In each iteration n , the residual vector \mathbf{r}_n and the Jacobian matrix \mathbf{J}_n are computed using the current estimate \mathcal{T}_n , and the following linear system is solved (using LU decomposition) to compute an increment $\Delta\mathcal{T}$:

$$\mathbf{J}_n^T \mathbf{J}_n \Delta\mathcal{T} = -\mathbf{J}_n^T \mathbf{r}_n. \quad (12)$$

At the end of every iteration, the motion estimate \mathcal{T} is updated as $\mathcal{T}_{n+1} = \mathcal{T}_n + \Delta\mathcal{T}$. Most operations are matrix-matrix or matrix-vector multiplications, which are trivially differentiable. Derivatives of the linear system solve operation are computed analytically, as described in [2] and detailed in the supplement. We use $\text{max_iter} = 3$ Gauss-Newton iterations, which encourages the correspondence prediction and weight functions, Φ and Ψ , respectively, to make predictions such that convergence in 3 iterations is possible. In our experiments we use $(\lambda_{2D}, \lambda_{\text{depth}}, \lambda_{\text{reg}}) = (0.001, 1, 1)$.

Algorithm 1 Gauss-Newton Optimization

```
1:  $\mathcal{C} \leftarrow \Phi(\mathcal{I}_s, \mathcal{I}_t)$  ▷ Estimate correspondences
2:  $\mathcal{W} \leftarrow \Psi(\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H})$  ▷ Estimate importance weights
3: function SOLVER( $\mathcal{C}, \mathcal{W}, \mathcal{V}$ )
4:    $\mathcal{T} \leftarrow \mathbf{0}$ 
5:   for  $n \leftarrow 0$  to  $max\_iter$  do
6:      $\mathbf{J}, \mathbf{r} \leftarrow \text{ComputeJacobianAndResidual}(\mathcal{V}, \mathcal{T}, \mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{C}, \mathcal{W})$ 
7:      $\Delta \mathcal{T} \leftarrow \text{LUdecomposition}(\mathbf{J}^T \mathbf{J} \Delta \mathcal{T} = -\mathbf{J}^T \mathbf{r})$  ▷ Solve linear system
8:      $\mathcal{T} \leftarrow \mathcal{T} + \Delta \mathcal{T}$  ▷ Apply increment
9:   return  $\mathcal{T}$ 
```

4.4 End-to-end Optimization

Given a dataset of samples $\mathcal{X}_{s,t} = \{[\mathcal{I}_s|\mathcal{P}_s], [\mathcal{I}_t|\mathcal{P}_t], \mathcal{V}\}$, our goal is to find the parameters ϕ and ψ of Φ_ϕ and Ψ_ψ , respectively, so as to estimate the motion \mathcal{T} of a deformation graph \mathcal{G} defined over the source RGB-D frame. This can be formulated as a differentiable optimization problem (allowing for back-propagation) with the following objective:

$$\arg \min_{\phi, \psi} \sum_{\mathcal{X}_{s,t}} \lambda_{\text{corr}} \mathcal{L}_{\text{corr}}(\phi) + \lambda_{\text{graph}} \mathcal{L}_{\text{graph}}(\phi, \psi) + \lambda_{\text{warp}} \mathcal{L}_{\text{warp}}(\phi, \psi) \quad (13)$$

Correspondence loss. We use a robust q -norm as in [30] to enforce closeness of correspondence predictions to ground-truth:

$$\mathcal{L}_{\text{corr}}(\phi) = \tilde{M}^{\mathcal{C}} (|\Phi_\phi(\mathcal{I}_s, \mathcal{I}_t) - \tilde{\mathcal{C}}| + \epsilon)^q. \quad (14)$$

Operator $|\cdot|$ denotes the ℓ_1 norm, $q < 1$ (we set it to $q = 0.4$) and ϵ is a small constant. Ground-truth correspondences are denoted by $\tilde{\mathcal{C}}$. Since valid ground truth for all pixels is not available, we employ a ground-truth mask $\tilde{M}^{\mathcal{C}}$ to avoid propagating gradients through invalid pixels.

Graph loss. We impose an l_2 -loss on node translations \mathbf{t} (ground-truth rotations are not available):

$$\mathcal{L}_{\text{graph}}(\phi, \psi) = \tilde{M}^{\mathcal{V}} \left\| \underbrace{\left[\Omega(\Phi_\phi(\mathcal{I}_s, \mathcal{I}_t), \Psi_\psi(\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H}), \mathcal{V}) \right]}_{\mathcal{T}} - \tilde{\mathbf{t}} \right\|_2^2, \quad (15)$$

where $[\cdot]_{\mathbf{t}} : \mathbb{R}^{N \times 6} \rightarrow \mathbb{R}^{N \times 3}$, $\mathcal{T} \mapsto [\mathcal{T}]_{\mathbf{t}} = \mathbf{t}$ extracts the translation part from the graph motion \mathcal{T} . Node translation ground-truth is denoted by $\tilde{\mathbf{t}}$ and $\tilde{M}^{\mathcal{V}}$ masks out invalid nodes. Please see the supplement for further details on how $\tilde{M}^{\mathcal{V}}$ is computed.

Warp loss. We have found that it is beneficial to use the estimated graph deformation \mathcal{T} to deform the dense source point cloud \mathcal{P}_s and enforce the result to be close to the source point cloud when deformed with the ground-truth scene flow $\tilde{\mathcal{S}}$:

$$\mathcal{L}_{\text{warp}}(\phi, \psi) = \tilde{M}^{\mathcal{S}} \left\| \text{Q} \left(\mathcal{P}_s, \underbrace{\Omega(\Phi_\phi(\mathcal{I}_s, \mathcal{I}_t), \Psi_\psi(\mathcal{Z}_s, \mathcal{Z}'_t, \mathcal{H}), \mathcal{V})}_{\mathcal{T}} \right) - (\mathcal{P}_s + \tilde{\mathcal{S}}) \right\|_2^2. \quad (16)$$

Here, we extend the warping operation Q (Eq. 2) to operate on the dense point cloud \mathcal{P}_s element-wise, and define $\tilde{M}^{\mathcal{S}}$ to mask out invalid points.

Note that We found it to be a more general notation to disentangle them (e.g., for scenarios where graph nodes are not sampled on the RGB-D frame).

4.5 Neural Non-rigid Tracking for 3D Reconstruction

We introduce our differentiable tracking module into the non-rigid reconstruction framework of Newcombe et al. [23]. In addition to the dense depth ICP correspondences employed in the original method, which help towards local deformation refinement, we employ a keyframe-based tracking objective. Without loss of generality, every 50th frame of the sequence is chosen as a keyframe, to

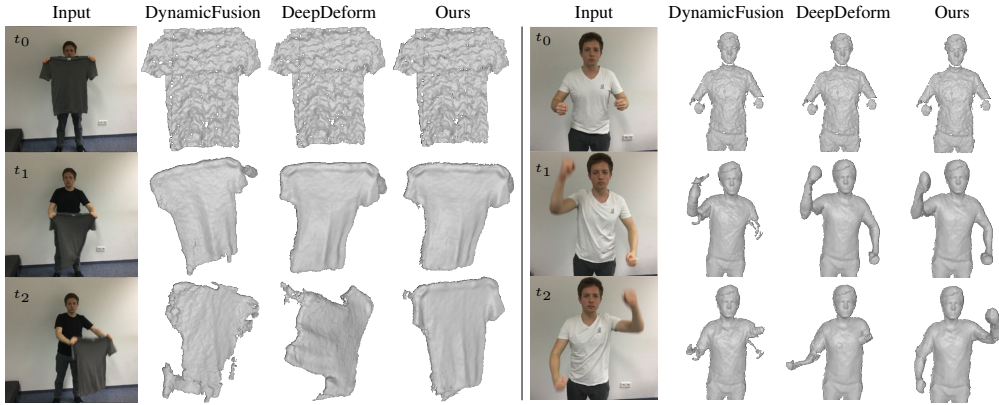


Figure 3: Qualitative comparison of our method with DynamicFusion [23] and DeepDeform [4] on test sequences from [4]. The rows show different time steps of the sequence.

which we establish dense correspondences including the respective weights. We apply a conservative filtering of the predicted correspondences based on the predicted correspondence weights using a fixed threshold $\delta = 0.35$ and re-weight the correspondences based on bi-directional consistency, i.e., keyframe-to-frame and frame-to-keyframe. Using the correspondence predictions and correspondence weights of valid keyframes ($> 50\%$ valid correspondences), the non-rigid tracking optimization problem is solved. The resulting deformation field is used to integrate the depth frame into the canonical volume of the object. We refer to the original reconstruction paper [23] for details regarding the fusion process.

5 Experiments

In the following, we evaluate our method quantitatively and qualitatively on both non-rigid tracking and non-rigid reconstruction. To this end, we use the DeepDeform dataset [4] for training, with the given 340-30-30 train-val-test split of RGB-D sequences. Both non-rigid tracking and reconstruction are evaluated on the hidden test set of the DeepDeform benchmark.

5.1 Training Scheme

The non-rigid tracking module has been implemented using the PyTorch library [24] and trained using stochastic gradient descent with momentum 0.9 and learning rate 10^{-5} . We use an Intel Xeon 6240 Processor and an Nvidia RTX 2080Ti GPU. The parameters of the dense correspondence prediction network ϕ are initialized with a PWC-Net model pre-trained on FlyingChairs [6] and FlyingThings3D [22]. We use a 10-factor learning rate decay every 10k iterations, requiring

Table 1: We evaluate non-rigid tracking on the DeepDeform dataset [4], showing the benefit of end-to-end differentiable optimizer losses and self-supervised correspondence weighting. We denote correspondence prediction as Φ_c , Φ_{c+g} and Φ_{c+g+w} , depending on which losses $\mathcal{L}_{\text{corr}}$, $\mathcal{L}_{\text{graph}}$, $\mathcal{L}_{\text{warp}}$ are used, and correspondence weighting as $\Psi_{\text{supervised}}$ and $\Psi_{\text{self-supervised}}$, either using an additional supervised loss or not.

Model	EPE 3D (mm)	Graph Error 3D (mm)
Φ_c	44.05	67.25
Φ_{c+g}	39.12	57.34
Φ_{c+g+w}	36.96	54.24
$\Phi_c + \Psi_{\text{supervised}}$	28.95	36.77
$\Phi_{c+g+w} + \Psi_{\text{supervised}}$	27.42	34.68
$\Phi_{c+g+w} + \Psi_{\text{self-supervised}}$	26.29	31.00

about 30k iterations in total for convergence, with a batch size of 4. For optimal performance, we first optimize the correspondence predictor Φ_ϕ with $(\lambda_{\text{corr}}, \lambda_{\text{graph}}, \lambda_{\text{warp}}) = (5, 5, 5)$, without the weighting function Ψ_ψ . Afterwards, we optimize the weighting function parameters ψ with $(\lambda_{\text{corr}}, \lambda_{\text{graph}}, \lambda_{\text{warp}}) = (0, 1000, 1000)$, while keeping ϕ fixed. Finally, we fine-tune both ϕ and ψ together, with $(\lambda_{\text{corr}}, \lambda_{\text{graph}}, \lambda_{\text{warp}}) = (5, 5, 5)$.

5.2 Non-rigid Tracking Evaluation

For any frame pair $\mathcal{X}_{s,t}$ in the DeepDeform data [4], we define a deformation graph \mathcal{G} by uniformly sampling graph nodes \mathcal{V} over the source object in the RGB-D frame, given a segmentation mask of the former. Graph node connectivity \mathcal{E} is computed using geodesic distances on a triangular mesh defined over the source depth map. As a pre-processing step, we filter out any frame pairs where more than 30% of the source object is occluded in the target frame. In Table 1 non-rigid tracking performance is evaluated by the mean translation error over node translations \mathbf{t} (Graph Error 3D), where the latter are compared to ground-truth with an l_2 metric. In addition, we evaluate the dense end-point-error (EPE 3D) between the source point cloud deformed with the estimated graph motion, $Q(\mathcal{P}_s, \mathcal{T})$, and the source point cloud deformed with the ground-truth scene flow, $\mathcal{P}_s + \tilde{\mathcal{S}}$. To support reproducibility, we report the mean error metrics of multiple experiments, running every setting 3 times. We visualize the standard deviation with an error plot in the supplement.

We show that using graph and warp losses, $\mathcal{L}_{\text{graph}}$ and $\mathcal{L}_{\text{warp}}$, and differentiating through the non-rigid optimizer considerably improves both EPE 3D and Graph Error 3D compared to only using the correspondence loss $\mathcal{L}_{\text{corr}}$. Adding self-supervised correspondence weighting further decreases the errors by a large margin. Supervised outlier rejection with binary cross-entropy loss does bring an improvement compared to models that do not optimize for the weighting function Ψ_ψ (please see supplemental material for details on this supervised training of Ψ_ψ). However, optimizing Ψ_ψ in a *self-supervised* manner clearly outperforms the former supervised setup. This is due to the fact that, in the self-supervised scenario, gradients that flow from $\mathcal{L}_{\text{graph}}$ and $\mathcal{L}_{\text{warp}}$ through the differentiable solver Ω can better inform the optimization of Ψ_ψ by minimizing the end-to-end alignment losses.

5.3 Non-rigid Reconstruction Evaluation

We evaluate the performance of our non-rigid reconstruction approach on the DeepDeform benchmark [4] (see Table 2). The evaluation metrics measure *deformation error*, a 3D end-point-error between tracked and annotated correspondences, and *geometry error*, which compares reconstructed shapes with annotated foreground object masks. Our approach performs about 8.9% better than the state-of-the-art non-rigid reconstruction approach of Božič et al. [4] on the deformation metric. While our approach consistently shows better performance on both metrics, we also significantly lower the per-frame runtime to 27 ms per keyframe, in contrast to [4], which requires 2299 ms. Thus, our approach can also be used with multiple keyframes at interactive frames rates, e.g., 90 ms for 5 keyframes and 199 ms for 10 keyframes.

Table 2: Our method achieves state-of-the-art non-rigid reconstruction results on the DeepDeform benchmark [4]. Both our end-to-end differentiable optimizer and the self-supervised correspondence weighting are necessary for optimal performance. Not only does our approach achieve lower deformation and geometry error compared to state of the art, our correspondence prediction is about $85\times$ faster.

Method	Deformation error (mm)	Geometry error (mm)
DynamicFusion [23]	61.79	10.78
VolumeDeform [13]	208.41	74.85
DeepDeform [4]	31.52	4.16
Ours (Φ_c)	54.85	5.92
Ours (Φ_{c+g+w})	53.27	5.84
Ours ($\Phi_c + \Psi_{\text{supervised}}$)	40.21	5.39
Ours ($\Phi_{c+g+w} + \Psi_{\text{self-supervised}}$)	28.72	4.03

To show the influence of the different learned components of our method, we perform an ablation study by disabling either of our two main components: the end-to-end differentiable optimizer or the self-supervised correspondence weighting. As can be seen, our end-to-end trained method with self-supervised correspondence weighting demonstrates the best performance. Qualitatively, we show this in Figure 3. In contrast to DynamicFusion [23] and DeepDeform [4], our method is notably more robust in fast motion scenarios. Additional qualitative results and comparisons to the methods of Guo et al. [10] and Slavcheva et al. [26] are shown in the supplemental material.

6 Conclusion

We propose Neural Non-Rigid Tracking, a differentiable non-rigid tracking approach that allows learning the correspondence prediction and weighting of traditional tracking pipelines in an end-to-end manner. The differentiable formulation of the entire tracking pipeline enables back-propagation to the learnable components, guided by a loss on the tracking performance. This not only achieves notably improved tracking error in comparison to state-of-the-art tracking approaches, but also leads to better reconstructions, when integrated into a reconstruction framework like DynamicFusion [23]. We hope that this work inspires further research in the direction of neural non-rigid tracking and believe that it is a stepping stone towards fully differentiable non-rigid reconstruction.

Broader Impact

Our paper presents learned non-rigid tracking. It is establishing the basis for the important research field of non-rigid tracking and reconstruction, which is needed for a variety of applications where man-machine and machine-environment interaction is required. These applications range from the field of augmented and virtual reality to autonomous driving and robot control. In the former, a precise understanding of dynamic and deformable objects is of major importance in order to provide an immersive experience to the user. Applications such as holographic calls would greatly benefit from research like ours. This, in turn, could provide society with the next generation of 3D communication tools. On the other hand, as a low-level building block, our work has no direct negative outcome, other than what could arise from the aforementioned applications.

Acknowledgments and Disclosure of Funding

This work was supported by the ZD.B (Zentrum Digitalisierung.Bayern), the Max Planck Center for Visual Computing and Communications (MPC-VCC), a TUM-IAS Rudolf Mößbauer Fellowship, the ERC Starting Grant Scan2CAD (804724), and the German Research Foundation (DFG) Grant Making Machine Learning on Static and Dynamic 3D Data Practical.

References

- [1] A. Avetisyan, A. Dai, and M. Nießner. End-to-end cad model retrieval and 9dof alignment in 3d scans. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2551–2560, 2019.
- [2] J. T. Barron and B. Poole. The fast bilateral solver. In *European Conference on Computer Vision (ECCV)*, pages 617–632. Springer, 2016.
- [3] A. Behl, D. Paschalidou, S. Donné, and A. Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7962–7971, 2019.
- [4] A. Božič, M. Zollhöfer, C. Theobalt, and M. Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [5] C.-H. Chang, C.-N. Chou, and E. Y. Chang. Clkn: Cascaded lucas-kanade networks for image alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2213–2221, 2017.
- [6] 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 (ICCV)*, pages 2758–2766, 2015.

- [7] M. Dou, J. Taylor, H. Fuchs, A. Fitzgibbon, and S. Izadi. 3d scanning deformable objects with a single rgb-d sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 493–501, 2015.
- [8] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):114, 2016.
- [9] M. Götz and H. Anzt. Machine learning-aided numerical linear algebra: Convolutional neural networks for the efficient preconditioner generation. In *2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA)*, pages 49–56, 2018.
- [10] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Transactions on Graphics (TOG)*, 36(3):32, 2017.
- [11] L. Han, M. Ji, L. Fang, and M. Nießner. Regnet: Learning the optimization of direct image-to-image pose registration. *arXiv preprint arXiv:1812.10212*, 2018.
- [12] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2462–2470, 2017.
- [13] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016.
- [14] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)* 28, pages 2017–2025, 2015.
- [15] Z. Lai, E. Lu, and W. Xie. Mast: A memory-augmented self-supervised tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6479–6488, 2020.
- [16] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, and M.-H. Yang. Joint-task self-supervised learning for temporal correspondence. In *Advances in Neural Information Processing Systems*, pages 318–328, 2019.
- [17] Y. Li, A. Božič, T. Zhang, Y. Ji, T. Harada, and M. Nießner. Learning to optimize non-rigid tracking. *arXiv preprint arXiv:2003.12230*, 2020.
- [18] P. Liu, M. Lyu, I. King, and J. Xu. Selfflow: Self-supervised learning of optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4571–4580, 2019.
- [19] X. Liu, C. R. Qi, and L. J. Guibas. FlowNet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019.
- [20] Z. Lv, F. Dellaert, J. M. Rehg, and A. Geiger. Taking a deeper look at the inverse compositional algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4581–4590, 2019.
- [21] W.-C. Ma, S. Wang, R. Hu, Y. Xiong, and R. Urtasun. Deep rigid instance scene flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3614–3622, 2019.
- [22] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.
- [23] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015.
- [24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimeshain, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)* 32, pages 8024–8035, 2019.
- [25] J. Sappl, L. Seiler, M. Harders, and W. Rauch. Deep learning of preconditioners for conjugate gradient solvers in urban water related problems. *arXiv preprint arXiv:1906.06925*, 2019.
- [26] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1386–1395, 2017.
- [27] M. Slavcheva, M. Baust, and S. Ilic. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2646–2655, 2018.
- [28] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

- [29] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *ACM Transactions on Graphics (TOG)*, 26(3):80–es, 2007.
- [30] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018.
- [31] C. Tang and P. Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018.
- [32] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2566–2576, 2019.
- [33] Z. Wang, S. Li, H. Howard-Jenkins, V. Prisacariu, and M. Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 91–98, 2020.
- [34] T. Yu, Z. Zheng, K. Guo, J. Zhao, Q. Dai, H. Li, G. Pons-Moll, and Y. Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 7287–7296, 2018.
- [35] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.

Who holds the Copyright on a NeurIPS paper

According to U.S. Copyright Office's page, **What is a Copyright**, when you create an original work you are the author and the owner and hold the copyright, unless you have an agreement to transfer the copyright to a third party such as the company or school you work for.

Authors do not transfer the copyright of their papers to NeurIPS. Instead, they grant NeurIPS a non-exclusive, perpetual, royalty-free, fully-paid, fully-assignable license to copy, distribute and publicly display all or part of the paper.

Figure C.2: Copyright authorization. Authorization for reuse of paper *Neural Non-rigid Tracking* in this dissertation.

C.3 Neural Deformation Graphs for Globally-consistent Non-rigid Reconstruction

COPYRIGHT

©2021 IEEE. Reprinted, with permission, from
Aljaž Božič, Pablo Palafox, Michael Zollhöfer, Justus Thies, Angela Dai, and
Matthias Nießner

Neural Deformation Graphs for Globally-consistent Non-rigid Reconstruction

IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) 2021

DOI: 10.1109/CVPR46437.2021.00150

SUMMARY

We introduce Neural Deformation Graphs for globally-consistent deformation tracking and 3D reconstruction of non-rigid objects. Specifically, we implicitly model a deformation graph via a deep neural network. This neural deformation graph does not rely on any object-specific structure and, thus, can be applied to general non-rigid deformation tracking. Our method globally optimizes this neural graph on a given sequence of depth camera observations of a non-rigidly moving object. Based on explicit viewpoint consistency as well as inter-frame graph and surface consistency constraints, the underlying network is trained in a self-supervised fashion. We additionally optimize for the geometry of the object with an implicit deformable multi-MLP shape representation. Our approach does not assume sequential input data, thus enabling robust tracking of fast motions or even temporally disconnected recordings. Our experiments demonstrate that our Neural Deformation Graphs outperform state-of-the-art non-rigid reconstruction approaches both qualitatively and quantitatively, with 64% improved reconstruction and 54% improved deformation tracking performance.

INDIVIDUAL CONTRIBUTIONS

Leading role in realizing the scientific project.

Problem definition	<i>significantly contributed</i>
Literature survey	<i>significantly contributed</i>
Implementation	<i>significantly contributed</i>
Experimental evaluation	<i>significantly contributed</i>
Preparation of the manuscript	<i>significantly contributed</i>

In accordance with the *IEEE Thesis / Dissertation Reuse Permissions*, we include the accepted version of the original publication [15] in the following.

Neural Deformation Graphs for Globally-consistent Non-rigid Reconstruction

Aljaž Božič¹ Pablo Palafox¹ Michael Zollhöfer² Justus Thies¹ Angela Dai¹ Matthias Nießner¹

¹Technical University of Munich ²Facebook Reality Labs Research

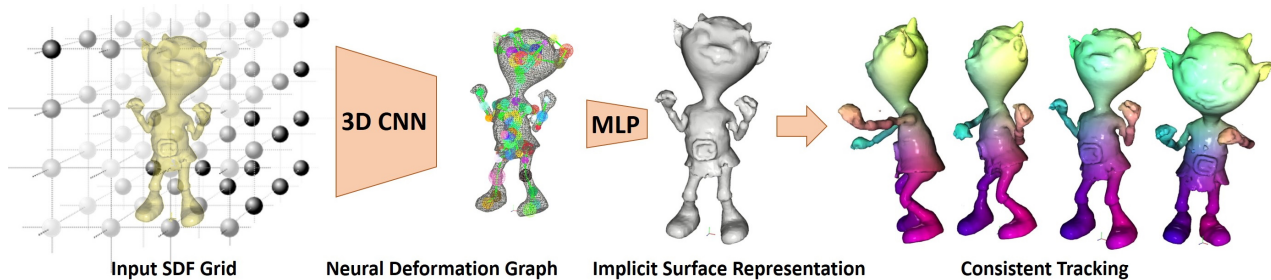


Figure 1: Neural Deformation Graphs: given range input data, represented as a signed distance field, our method predicts globally-consistent deformation graph that is used to reconstruct the non-rigidly deforming surface of an object. The surface of the object is represented as a set of implicit functions centered around the deformation graph nodes. Our global optimization provides consistent surface and deformation prediction, enabling robust tracking of an observed input sequence and even multiple disjoint captures of the same object (as we do not assume sequential input data).

Abstract

We introduce *Neural Deformation Graphs* for globally-consistent deformation tracking and 3D reconstruction of non-rigid objects. Specifically, we implicitly model a deformation graph via a deep neural network. This neural deformation graph does not rely on any object-specific structure and, thus, can be applied to general non-rigid deformation tracking. Our method globally optimizes this neural graph on a given sequence of depth camera observations of a non-rigidly moving object. Based on explicit viewpoint consistency as well as inter-frame graph and surface consistency constraints, the underlying network is trained in a self-supervised fashion. We additionally optimize for the geometry of the object with an implicit deformable multi-MLP shape representation. Our approach does not assume sequential input data, thus enabling robust tracking of fast motions or even temporally disconnected recordings. Our experiments demonstrate that our *Neural Deformation Graphs* outperform state-of-the-art non-rigid reconstruction approaches both qualitatively and quantitatively, with 64% improved reconstruction and 54% improved deformation tracking performance. Code is publicly available.¹

¹aljazbozic.github.io/neural-deformation-graphs

1. Introduction

Capturing non-rigidly deforming surfaces is essential towards reconstructing and understanding real-world environments, which are often highly dynamic. While impressive advances have been made in reconstructing static 3D scenes [8, 21], dynamic tracking and reconstruction remains very challenging. A plethora of domain-specific dynamic tracking methods has been developed (e.g., human bodies, faces, hands), leveraging strong domain shape and motion priors for robust tracking [4, 28, 31, 41]. However, real-world environments encompass a vast diversity of deformable objects – including people with clothing or animals – making domain specific shape priors often intractable for general deformable reconstruction; in this work, we thus focus on general non-rigid 3D reconstruction without shape or motion priors for general object tracking and reconstruction.

A seminal work in non-rigid 3D reconstruction is *DynamicFusion* [32], which was the first approach to demonstrate real-time dense reconstruction of dynamic scenes using just a single RGB-D sensor. *DynamicFusion* showed promising results towards dynamic reconstruction, but still struggles in many real-world scenarios, which typically include strong deformations and fast frame-to-frame motion, due to its low-level, local correspondence association step.

In particular, the incremental construction of a deformation graph is prone to error aggregation and can lead to tracking failures. Recently, data-driven methods based on deep learning have been introduced [3, 25, 2] that learn priors of non-rigidly deforming objects from dense flow annotations. These approaches leverage a similar incremental deformation graph construction as DynamicFusion, but learn to establish more robust tracking via more sophisticated correspondence optimization based on data-driven priors. However, despite more robust correspondences, these methods still operate on a frame-by-frame basis, thus, aggregate tracking errors and are unable to recover if tracking fails. In order to address these shortcomings without assuming data-driven priors, we propose a globally-consistent neural deformation graph which allows for non-rigid reconstruction from commodity sensor observations, represented as signed distance fields (see Fig. 1). The neural deformation graph gives access to the per frame deformation graph nodes and stores the global graph connectivity. To robustly optimize for consistent deformations over fast motions, we introduce viewpoint consistency (independently for every frame) as well as graph and surface consistency constraints (between pairs of frames). Our viewpoint consistency loss measures the consistency of graph node position predictions w.r.t. rotation augmentation. The graph and surface consistency losses encourage deformations to be modeled in our Neural Deformation Graph such that local graph edge distances are preserved between frames and the deformed surface geometry of a source frame aligns well with the geometry of the target frame. Additionally, our approach does not assume temporally close frames, thus making it easily applicable to low FPS settings or the combination of independently captured depth recordings.

Since there exists no general canonical pose (like a T-pose of a human [28]) that fits all deformable objects, we avoid modeling it explicitly. Instead, we propose to employ a set of implicit functions that are centered around the deformation graph nodes. Specifically, we model local signed distance functions (SDFs) using multi-layer perceptrons (MLPs) that can be deformed to fit any frame, without requiring an explicit canonical pose. The global shape is evaluated by the integration of these local MLP predictions.

To summarize, our technical contributions are:

- a globally-optimized deformation graph that is able to handle deformations present in all frames of an unstructured dataset or a sequence of an object;
- a combination of per-frame viewpoint consistency and frame-to-frame graph and surface consistency for robust tracking of fast deformations;
- an implicit deformable multi-MLP shape representation anchored on the scene-specific deformation graph.

2. Related Work

Our approach is leveraging a low dimensional deformation graph to model the non-rigid deformations of an object, while the actual surface is represented by an implicit function by means of a multi-layer perceptron (MLP). We will discuss the most related approaches in these two fields.

Non-rigid Reconstruction Non-rigid reconstruction is a highly active research field, in particular using commodity RGB-D sensors such as the Kinect. The seminal work DynamicFusion of Newcombe et al. [32] tracks deformable motion and reconstructs the object’s shape in an incremental fashion, i.e., frame-by-frame. While this approach relies on local depth correspondences, follow-up methods additionally use sparse SIFT features [20], dense color tracking [17] or dense SDF alignment [37, 38]. These methods show impressive results, but often struggle with fast frame-to-frame motion given their use of hand-crafted correspondences. Bozic et al. [3] introduced an annotated dataset of non-rigid motions that allows to train data-driven non-rigid reconstruction methods with learned correspondences [3, 25, 2]. While learned correspondences improve tracking performance, the approaches are still inherently limited by the employed frame-to-frame tracking paradigm, i.e., tracking errors accumulate over time, and if tracking is lost it is unable to recover. Tracking robustness can also be improved without any learned priors by using multi-view input data [7, 15] (setups with more than 50 cameras) and high-speed cameras [10] (8 cameras at 200 frames per second (FPS)). In contrast to these frame-to-frame tracking approaches, there are methods that focus on global non-rigid optimization [11, 44, 19, 35]; however, these methods either assume ground-truth optical flow [35], or they share the same drawbacks of the aforementioned frame-to-frame tracking approaches [11, 44, 19], and thus have difficulties handling fast deformable motion.

Deformation Graphs State-of-the-art non-rigid reconstruction methods often model deformations with a sparse deformation graph, following the Embedded Deformation [40] formulation. Deformation graphs offer a robust alternative to dense motion estimation with optical flow or scene flow methods, since they can estimate plausible motion even in partially occluded shape parts, when combined with motion regularization such as ARAP [39]. Existing non-rigid reconstruction approaches build the deformation graph incrementally, i.e., frame-by-frame, which can lead to unstable graph configurations in the case of tracking errors. In our approach, we predict a globally consistent deformation graph that can represent motion in all frames of the sequence, while being robust w.r.t. tracking errors present in challenging frames.

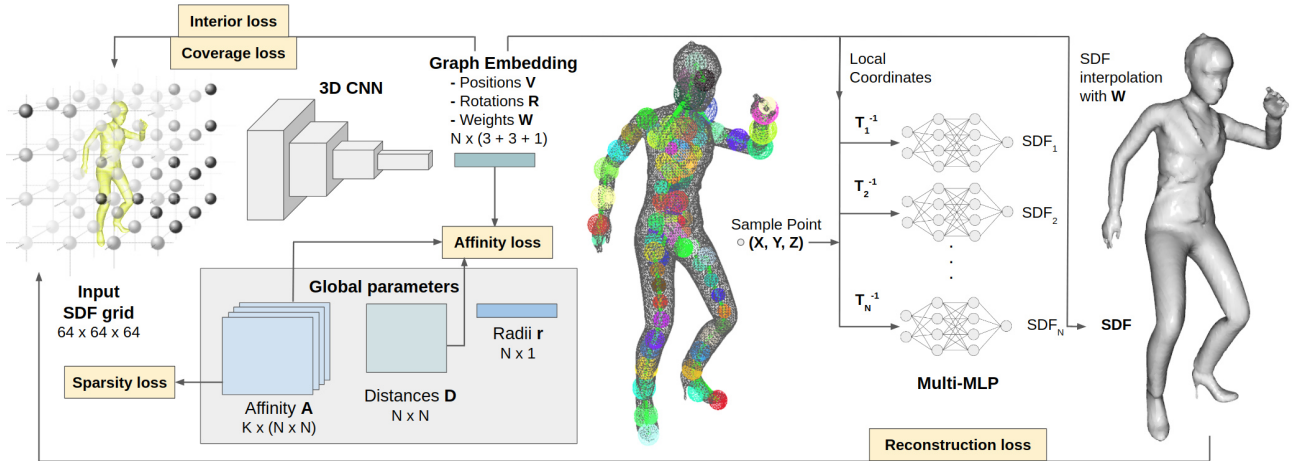


Figure 2: A Neural Deformation Graph encodes a 64^3 SDF grid to a graph embedding with graph node positions \mathbf{V} , rotations \mathbf{R} and importance weights \mathbf{W} . To compute an SDF value for a sample point $(X, Y, Z) \in \mathbb{R}^3$, the point is transformed to local coordinates around each node, and passed through locally embedded implicit functions that are represented as MLPs; the global SDF value is computed by interpolating the local MLP predictions using the node radii \mathbf{r} and importance weights \mathbf{W} . For graph regularization, a set of affinity matrices $\mathbf{A}_i \in \mathbb{R}^{N \times N}$ and a node-to-node distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ are globally optimized.

Sparse motion representations are common for human deformation modeling as well: the human skeletons used in [28, 4] are also instances of deformation graphs. Some works have tried to extend human skeletons to more general objects, but with limited success. In [1], a fixed generic skeleton is fitted to different object meshes, resulting in human-like re-animation of characters, but not general enough to be able to represent all degrees of freedom of general shapes. Fixed hierarchical deformation graphs are used for differentiable non-rigid tracking in [43], but a pre-computed graph template is required, with fixed connectivity on different coarseness levels. Thus, it is only applicable to specific object types (e.g., used for hand tracking). Data-driven skeleton prediction has been introduced in [45], but it requires a dataset of manually designed skeletons as supervision, which is hard to obtain for general objects. Our method, instead, estimates both deformation graph nodes and connectivity of general deformable objects in a self-supervised manner.

Implicit Surface Representation Representing surface geometry implicitly with a signed distance field (SDF) has been extensively used in the non-rigid reconstruction community. An efficient algorithm for SDF grid construction from range images has been presented in [8] and extended to support non-rigid deformations in [32]. These methods rely on a discretized 3D grid to store the SDF, which can cause loss of detail, since grid resolution is limited by available memory. A promising direction is to not use discretized grids at all, but instead represent the SDF function continu-

ously using a multi-layer perceptron (MLP), as introduced in [34, 36, 6]. An implicit surface representation is used in [18] for accurate human reconstruction, where the SDF is estimated in a canonical T-pose space. Since there exist no methods for estimation of canonical T-pose spaces for general non-rigid shapes, we instead base our method on the approach of Deng et al. [9]. Assuming ground-truth dense body and skeleton tracking, they represent the human body with multiple MLPs, one for each bone and in its own canonical space, centered around the bone, therefore eliminating the need for a T-pose space. In our general reconstruction approach, we estimate a deformation graph via self-supervision, and append an MLP to each deformation node to represent the surface of the observed object.

While most implicit reconstruction approaches do not produce consistent tracking, methods such as [13, 12, 42] reconstruct objects in a patch-based manner and empirically observe consistency of patches across different deformations. We compare our method, which leverages explicit consistency constraints, to these approaches to evaluate such implicit patch consistency.

3. Method

Given a sequence of signed distance fields observing a non-rigidly deforming surface, our method estimates the dense deformable motion in the sequence as well as reconstructs the geometry of the observed shape. Specifically, we apply self-supervised learning on the sequence that we want to reconstruct. A convolutional neural network that takes a

signed distance field (SDF) as input is trained to predict a consistent deformation graph. We call this neural network *Neural Deformation Graph*, as it implicitly stores the deformation graphs of each frame. Using the predicted graph node positions and orientations, we learn implicit functions to represent the shape of each graph node and, thus, the entire shape of the object. The implicit functions are represented as a multi-layer perceptron (MLP). These MLPs take a 3D point centered around the node position as input and predict its signed distance value, defining the local geometry around the node. Warping all node MLPs to every time step and interpolating their local part reconstruction results in an accurate *implicit deformable shape reconstruction* without the need for an explicit canonical pose. In addition to the sample point locations, the MLPs are conditioned on the predicted graph positions, which enables reconstruction of pose-dependent geometry detail. Using Marching Cubes [29], the geometry can be extracted as a mesh at every time step, with dense correspondences estimated throughout the entire sequence.

3.1. Neural Deformation Graphs

A deformation graph consists of graph nodes and graph edges. We represent the graph nodes \mathcal{V} of each frame of the sequence implicitly by a neural network (Neural Deformation Graph). The graph connectivity is explicitly stored for the entire sequence as an affinity matrix $\mathcal{E} \in \mathbb{R}^{N \times N}$ ($N = |\mathcal{V}|$). A node $v \in \mathcal{V}$ is characterized by its 3D position $\mathbf{v} \in \mathbb{R}^3$, rotation $R \in \mathbb{R}^{3 \times 3}$, importance weight $w \in \mathbb{R}$ (describing importance of the node, explained below), radius $r \in \mathbb{R}$ (describing the spatial influence of the node), and a local implicit shape function f . We denote the set of node positions $\mathbf{V} = \{\mathbf{v}\}$, rotations $\mathbf{R} = \{R\}$, importance weights $\mathbf{W} = \{w\}$, radii $\mathbf{r} = \{r\}$, and shape functions $\mathbf{f} = \{f\}$, with $\mathcal{V} = (\mathbf{V}, \mathbf{R}, \mathbf{W}, \mathbf{r}, \mathbf{f})$.

To generate the signed distance fields needed for our method, we assume four calibrated cameras. The depth maps at each time step k are back-projected into a common coordinate system and converted into a signed distance field \mathbf{S}_k of dimension 64^3 using static volumetric reconstruction [22]. Note that due to occlusions this representation is partial, thus only an approximate signed distance field is used for our deformation graph prediction. Based on this input, we estimate $(\mathbf{V}_k, \mathbf{R}_k, \mathbf{W}_k)$ using a Neural Deformation Graph (NDG) which is based on a 3D convolutional neural network (see supplemental material for architecture details):

$$(\mathbf{V}_k, \mathbf{R}_k, \mathbf{W}_k) = \text{NDG}(\mathbf{S}_k).$$

The radii \mathbf{r} of the graph nodes as well as the graph node affinities \mathcal{E} are jointly optimized over the entire input sequence. In addition to the affinity matrix, we also store the average edge lengths (node-to-node distances) $\mathbf{D} \in \mathbb{R}^{N \times N}$, which are used for regularization. For every graph node, we

also optimize for a local MLP which is used to represent the surface of the object (see Sec 3.3).

We define a fixed number of graph nodes ($N = 100$) in our experiments; note that this is an upper bound on the effective number of nodes, since the importance weights allow eliminating the effect of redundant nodes, making our method applicable to shapes of different size and structure complexity. To achieve a consistent graph node prediction via self-supervised training, we employ the following constraints for each time-step k .

Graph coverage loss. A deformation graph should cover the entire object to ensure that every deformable part can be represented while simultaneously enforcing that free space is not covered. To this end, we employ a loss that encourages the coverage of the shape by the node centers (w.r.t. their radii). We define the influence of a node (with position \mathbf{v} , radius $r > 0$, and importance weight $w > 0$) on a point $\mathbf{x} \in \mathbb{R}^3$ using a *weighted Gaussian function*:

$$\mathcal{G}(\mathbf{x}, \mathbf{v}, r, w) = w \cdot \exp\left(-\frac{\|\mathbf{x} - \mathbf{v}\|_2^2}{r^2}\right)$$

The coverage of a point $\mathbf{x} \in \mathbb{R}^3$ is computed by summing the corresponding contributions of all nodes, and applying a *sigmoid* to encourage a fast transition from covered (where coverage value is 1) to free space (where coverage value should be 0), enabling more accurate surface coverage:

$$\mathcal{C}(\mathbf{x}, \mathbf{V}_k, \mathbf{r}, \mathbf{W}_k) = \sigma\left(s \left(\left(\sum_{\mathbf{v}, r, w} \mathcal{G}(\mathbf{x}, \mathbf{v}, r, w) \right) - d \right)\right)$$

We empirically set $d = 0.07$ and $s = 100.0$. To compute the coverage loss, we sample points P_{un} uniformly in the shape’s bounding box and points P_{ns} near the surface region. Points are assigned coverage value of $c = 0$ if they are visible in at least one of the cameras, otherwise they are assigned $c = 1$. The coverage loss then compares predicted coverage of these point samples with the pre-computed coverage using an ℓ_2 loss:

$$\mathcal{L}_{\text{coverage}} = \lambda_{\text{un}} \sum_{(\mathbf{x}, c) \in P_{\text{un}}} \|\mathcal{C}(\mathbf{x}, \mathbf{V}_k, \mathbf{r}, \mathbf{W}_k) - c\|_2^2 + \lambda_{\text{ns}} \sum_{(\mathbf{x}, c) \in P_{\text{ns}}} \|\mathcal{C}(\mathbf{x}, \mathbf{V}_k, \mathbf{r}, \mathbf{W}_k) - c\|_2^2$$

Node interior loss. In addition to the graph coverage loss, we require the node positions to be predicted inside the shape. If any node’s position \mathbf{v} is predicted outside the observed surface \mathbf{S}_k , i.e., in the SDF region with positive signed distance value, we penalize it to encourage the node’s position to be inside the surface:

$$\mathcal{L}_{\text{interior}} = \sum_{\mathbf{v} \in \mathbf{V}_k} \max(\text{interp}(\mathbf{S}_k, \mathbf{v}), 0)$$

Here $\text{interp}(\mathbf{S}_k, \mathbf{v})$ is the trilinear interpolation of \mathbf{S}_k at \mathbf{v} .

Affinity consistency loss. We also optimize for a global affinity matrix $\mathcal{E} = \{e_{ij} \mid i \in [1, N], j \in [1, N]\}$ representing node-to-node affinities across the entire input sequence. We compute node-to-node Euclidean distances $\|\mathbf{v}_i^k - \mathbf{v}_j^k\|_2$ at each frame k , and weight them by connectivity weights e_{ij} ; this should remain consistent over the whole sequence (relative loss, preserving edge length) and relatively small (absolute loss, preferring close-by connections). To ensure global distance consistency, we additionally optimize over average node-to-node distances d_{ij} , resulting in the affinity loss:

$$\mathcal{L}_{\text{affinity}} = \lambda_{\text{rel}} \sum_{i \neq j} e_{ij} \left| d_{ij}^2 - \|\mathbf{v}_i^k - \mathbf{v}_j^k\|_2^2 \right|_1 + \lambda_{\text{abs}} \sum_{i \neq j} e_{ij} \|\mathbf{v}_i^k - \mathbf{v}_j^k\|_2^2$$

Neighbor diversity loss. We enforce a sparse connectivity of the graph. Specifically, each node can have up to K neighbors ($K = 2$ in our setting); we use a (soft) loss to encourage these neighbors to be different. To achieve this, we optimize over a set of matrices $\mathbf{A}_1, \dots, \mathbf{A}_K \in \mathbb{R}^{N \times N}$, and construct $\mathcal{E} \in \mathbb{R}^{N \times N}$ as:

$$\mathcal{E} = \frac{1}{K} \sum_{i=1}^K \text{softmax}(\mathbf{A}_i)$$

We use softmax over the rows of matrix \mathbf{A}_i to guarantee all affinity elements of a node to be positive and add up to 1. To enforce unique graph neighbors, a neighbor diversity loss is employed, encouraging different matrices \mathbf{A}_i to produce different neighbors:

$$\mathcal{L}_{\text{sparsity}} = \sum_{l \neq m} \|\text{softmax}(\mathbf{A}_l) \odot \text{softmax}(\mathbf{A}_m)\|_F^2$$

We use \odot to denote the element-wise product.

3.2. Global Deformation Optimization

We compute deformation between any pair of frames by interpolating the nodes' relative motions (translations and rotations), weighted by their influences \mathcal{G} . For a source frame s and target frame t , the warping of point $\mathbf{x} \in \mathbb{R}^3$ from frame s to frame t is defined as:

$$\mathcal{W}_{s \rightarrow t}(\mathbf{x}) = \sum_{i=1}^N \mathcal{G}(\mathbf{x}, \mathbf{v}_i^s, r_i, w_i^s) (\mathbf{R}_i^t (\mathbf{R}_i^s)^T (\mathbf{x} - \mathbf{v}_i^s) + \mathbf{v}_i^t)$$

We denoted parameters at the source frame with $(\cdot)^s$ and at the target frame with $(\cdot)^t$. We use the Embedded Deformation formulation [40] to parameterize frame-to-frame

deformation, but instead of fixed-radius skinning we employ node influence \mathcal{G} as the skinning weight, which enables different skinning effects for every node as well as frame-adaptive skinning, i.e., skinning can change depending on the deformation. To ensure globally consistent deformation, we employ a per-frame viewpoint consistency loss and a surface consistency loss.

Viewpoint consistency loss. Since input observations may see very different views, we enforce a viewpoint consistency loss for consistent graph node predictions across varying views. To this end, for each frame k , the rotated 3D input \mathbf{S}_k should produce consistent graph node positions \mathbf{V}_k , rotations \mathbf{R}_k and importance weights \mathbf{W}_k . In our experiments, we only consider view rotations around the y-axis, since the camera setup is arranged in the x-z plane. In each batch, we sample two random angles α and β for every sample, and compute rotated inputs $\pi_\alpha(\mathbf{S}_k)$ and $\pi_\beta(\mathbf{S}_k)$ by trilinear re-sampling of input SDF grid \mathbf{S}_k using rotated grid indices. Viewpoint consistency is then measured by:

$$\mathcal{L}_{\text{vc}} = \|\pi_\alpha^{-1} \text{NDG}(\pi_\alpha(\mathbf{S}_k)) - \pi_\beta^{-1} \text{NDG}(\pi_\beta(\mathbf{S}_k))\|_2^2$$

where the function π_ϕ^{-1} corrects for the input rotation of angle ϕ : $\pi_\phi^{-1}(\mathbf{V}_k, \mathbf{R}_k, \mathbf{W}_k) = (R_\phi^T \mathbf{V}_k, R_\phi^T \mathbf{R}_k, \mathbf{W}_k)$.

Surface consistency loss. Surface points from a source frame s should, after deformation to a target frame t , align well with the target frame's SDF grid \mathbf{S}_t . We sample surface points P_s in the source frame and warp them to the target frame using the predicted deformation, to trilinearly interpolate the target grid \mathbf{S}_t , encouraging surface points to be warped to near zero (surface) SDF values:

$$\mathcal{L}_{\text{sc}} = \sum_{\mathbf{x} \in P_s} (\text{interp}(\mathbf{S}_t, \mathcal{W}_{s \rightarrow t}(\mathbf{x})))^2$$

This consistency loss is computed between pairs of samples in the batch, with uniformly sampled batch samples.

3.3. Implicit Surface Reconstruction

We represent the surface of the object as an implicit function. Specifically, each graph node i defines local geometry over the influence of that node, with an implicit function f_i , represented by an MLP. This MLP takes a location in the local space as input and outputs an SDF value. Any point $\mathbf{x} \in \mathbb{R}^3$ in the current frame k can be transformed to the local coordinate system of node i as $\mathcal{W}_{k,i}^{-1}(\mathbf{x}) = \mathbf{P}_{\text{enc}}((\mathbf{R}_i^k)^T (\mathbf{x} - \mathbf{v}_i^k))$. $\mathbf{P}_{\text{enc}} : \mathbb{R}^3 \rightarrow \mathbb{R}^F$ denotes positional encoding that transforms 3D local coordinates to a high-dimensional frequency domain (in our case $F = 30$), as presented in [30]. Inspired by Deng et al. [9], we condition each f_i on the predicted input frame's

graph parameters, such that they can encode pose-specific geometry details. We train a linear layer $\Pi_i(\cdot)$ to select a sparse pose code (of dimension $D = 32$) for every f_i from the graph predictions $\text{NDG}(\mathbf{S}_k)$. Given this input of dimension $D + F$, we use 8 linear layers with feature dimension of 32, a leaky ReLU (with negative slope of 0.01) as activation function, and skip connections between the input and the 6th linear layer.

We compute the full surface reconstruction \mathcal{S}_k as an SDF created from interpolating the SDF output values of each local MLP f_i , using the aforementioned skinning weights and transformations to the current frame by the estimated nodes' rotations and translations:

$$\mathcal{S}_k(\mathbf{x}) = \sum_{i=1}^N \mathcal{G}(\mathbf{x}, \mathbf{v}_i^k, r_i, w_i^k) f_i(\mathcal{W}_{k,i}^{-1}(\mathbf{x}), \Pi_i(\text{NDG}(\mathbf{S}_k)))$$

This operation is efficiently implemented using group convolutions. During training, we use the same point samples P_{un} and P_{ns} as for the graph coverage loss, sampled uniformly and near the surface, but instead of the 0/1 coverage values we use their approximate SDF values. We then optimize for $\{f_i\}$ using the SDF reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \sum_{(\mathbf{x}, \text{sdf}) \in P_{\text{un}} \cup P_{\text{ns}}} |\mathcal{S}_k(\mathbf{x}) - \text{sdf}|_1$$

3.4. Training Details

We use the Adam solver [23] with momentum of 0.9 to optimize the complete loss:

$$\mathcal{L} = \mathcal{L}_{\text{coverage}} + \lambda_{\text{interior}} \mathcal{L}_{\text{interior}} + \mathcal{L}_{\text{affinity}} + \lambda_{\text{sparsity}} \mathcal{L}_{\text{sparsity}} + \lambda_{\text{vc}} \mathcal{L}_{\text{vc}} + \lambda_{\text{sc}} \mathcal{L}_{\text{sc}} + \lambda_{\text{recon}} \mathcal{L}_{\text{recon}}$$

Our method is trained in two stages. We initially train the CNN encoder with all losses except the reconstruction loss, and afterwards train the multi-MLP network using only the reconstruction loss, with the CNN encoder frozen.

The CNN encoder is trained for 500k iterations with a learning rate of $5e^{-5}$ and a batch size of 16; we balance the losses with $\lambda_{\text{un}} = 1.0$, $\lambda_{\text{ns}} = 0.1$, $\lambda_{\text{interior}} = 1.0$, $\lambda_{\text{rel}} = 0.1$, $\lambda_{\text{abs}} = 0.1$, $\lambda_{\text{sparsity}} = 1e^{-8}$, $\lambda_{\text{vc}} = (10.0, 1.0, 1e^{-4})$ (for graph node's position, weight and rotation, respectively) and $\lambda_{\text{sc}} = 1e^{-6}$. Every 50k iterations we increase the loss weights λ_{rel} , λ_{abs} , $\lambda_{\text{sparsity}}$, λ_{sc} by a factor of 10, up to maximum weights $\lambda_{\text{rel}}^{\text{max}} = 10000.0$, $\lambda_{\text{abs}}^{\text{max}} = 1.0$, $\lambda_{\text{sparsity}}^{\text{max}} = 1e^{-3}$ and $\lambda_{\text{sc}}^{\text{max}} = 1000.0$.

The multi-MLP network is trained for 500k iterations with a learning rate of $5e^{-4}$ and a batch size of 4, only based on the reconstruction loss with $\lambda_{\text{recon}} = 1.0$.

4. Results

To evaluate our proposed approach, we conducted a series of experiments on real and synthetic recordings where ground truth data is available.

Method	Chamfer	EPE3D
SIF [13]	1.12	8.56
LDIF [12]	2.41	10.40
OccupancyFlow [33]	53.83	16.29
Multiview DynamicFusion [32]	2.19	3.06
MV DF [32] + FlowNet3D [27]	1.93	2.55
Robust L0 Non-rigid Tracking [16]	2.31	2.50
Ours: GRAPH	0.50	8.93
Ours: GRAPH + AO	0.46	8.03
Ours: GRAPH + AO + VC	0.44	4.12
Ours: GRAPH + AO + VC + SC	0.40	1.16

Table 1: We show quantitative comparisons with state-of-the-art approaches, evaluating geometry using chamfer distance ($\times 10^{-4}$), and deformation using EPE3D ($\times 10^{-2}$). We also include an ablation study of different components of our method: GRAPH = coverage and interior losses, AO = affinity optimization with affinity consistency and sparsity, VC = viewpoint consistency, SC = surface consistency.

Evaluation on Synthetic Data In order to quantitatively and qualitatively evaluate our method, we make use of synthetic human-like and character sequences from the DeformingThings4D dataset [26]. To mimic our real data capture setup, we render 4 fixed depth views for every frame of synthetic animation, and generate SDF grids from these 4 views. Quantitative evaluation is executed on three sequences, including human, character and bear motion, as shown in Fig. 3. The geometry reconstruction is evaluated using $L2$ Chamfer distance, which computes average squared bi-directional point-to-point distance between reconstructed and ground truth geometry for every time step, thus evaluating accuracy and completeness of geometry. For deformation evaluation, we uniformly sample 10 keyframes per sequence, and compute dense deformation from each of these keyframes to any other frame, measuring average $L2$ End-Point-Error (EPE3D) between estimated and ground truth motion. The depth data of every sequence is normalized such that the largest bounding box side length is equal to 1.0. All numbers are listed w.r.t. this unit cube, thus, being independent to the scale of the objects.

In Tab. 1 we quantitatively compare our approach to the state-of-the-art network-based reconstruction methods SIF [13], LDIF [12] and OccupancyFlow [33], as well as to the non-rigid reconstruction approach DynamicFusion [32], which we extend to the multi-view domain, and the Robust L0 Non-rigid Tracking method [16]. Among the baselines the best reconstruction performance (lower Chamfer distance) is achieved by SIF [13], while the Robust L0 Non-rigid Tracking method [16] obtains better deformation

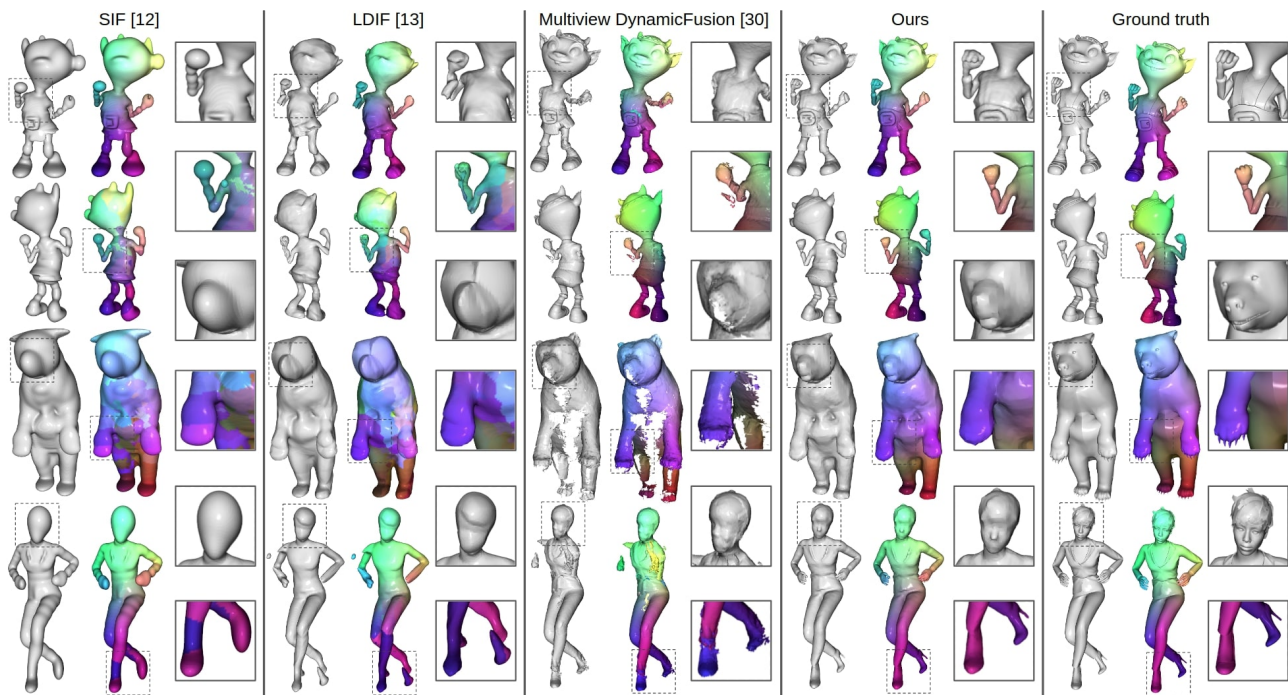


Figure 3: We qualitatively compare our method to the baseline methods on synthetic data. Each point is given a color value w.r.t. its location in the bounding box of the first frame. With perfect tracking and reconstruction, a specific point on the surface will have the same color throughout the entire sequence, while errors in tracking result in changing surface colors. Our approach outperforms state of the art in both reconstruction and deformation tracking quality.

tracking performance (lower EPE3D). Our approach outperforms all methods on both reconstruction and deformation tracking metrics, achieving 64% better reconstruction and 54% better deformation tracking results. The improvement is also clearly visible in the qualitative comparisons shown in Fig. 3. The methods SIF [13] and LDIF [12] produce less accurate geometry reconstruction with tracking failures under larger deformations (e.g., flipped legs in the human sequence). We trained Occupancy Flow [33] on our sequences, which are noticeably longer (about 500 frames) than the sequences the authors used (up to 50 frames), resulting in worse performance. The multi-view DynamicFusion [32] baseline is a specialized framework for both non-rigid tracking and reconstruction, with coarse-to-fine multi-frame alignment using depth iterative closest point (ICP) correspondences and non-rigid volumetric fusion. We use data-driven correspondences from the off-the-shelf flow estimator FlowNet3D [27] to further improve the method. However, it suffers from incomplete geometry because of the incremental graph construction and surface integration, and can also not recover from tracking failures. In contrast, our method is robust in the case of large deformation and produces complete and accurate geometry reconstruction.

Evaluation on Real Data. Our real data capture setup consists of 4 Kinect Azure sensors with hardware synchronization. The cameras are calibrated with a checkerboard using OpenCV [24] and an additional refinement procedure based on ICP [5]. Before recording an actual sequence, we record the background to compute the floor plane using PCA. During capture, we filter out floor points and background points, i.e., all points outside of a cylinder with diameter 1.8 m and height 2.5 m. We use the wide-field-of-view depth capture setting with a resolution of 1024×1024 pixels, at the highest available frame-rate of 15 FPS for this resolution. In Fig. 4, we show a comparison between the multi-view DynamicFusion approach [32] and ours. Our approach achieves considerably more accurate deformation tracking (color is retrieved from the first frame) while also producing more complete and accurate geometry reconstruction. More qualitative results are shown in the accompanying video.

Ablation Study of Network Components. To evaluate specific parts of our method, we employ an ablation study. Specifically, we analyzed the performance of our method by performing optimization without using certain losses: without affinity related losses (affinity consistency and sparsity

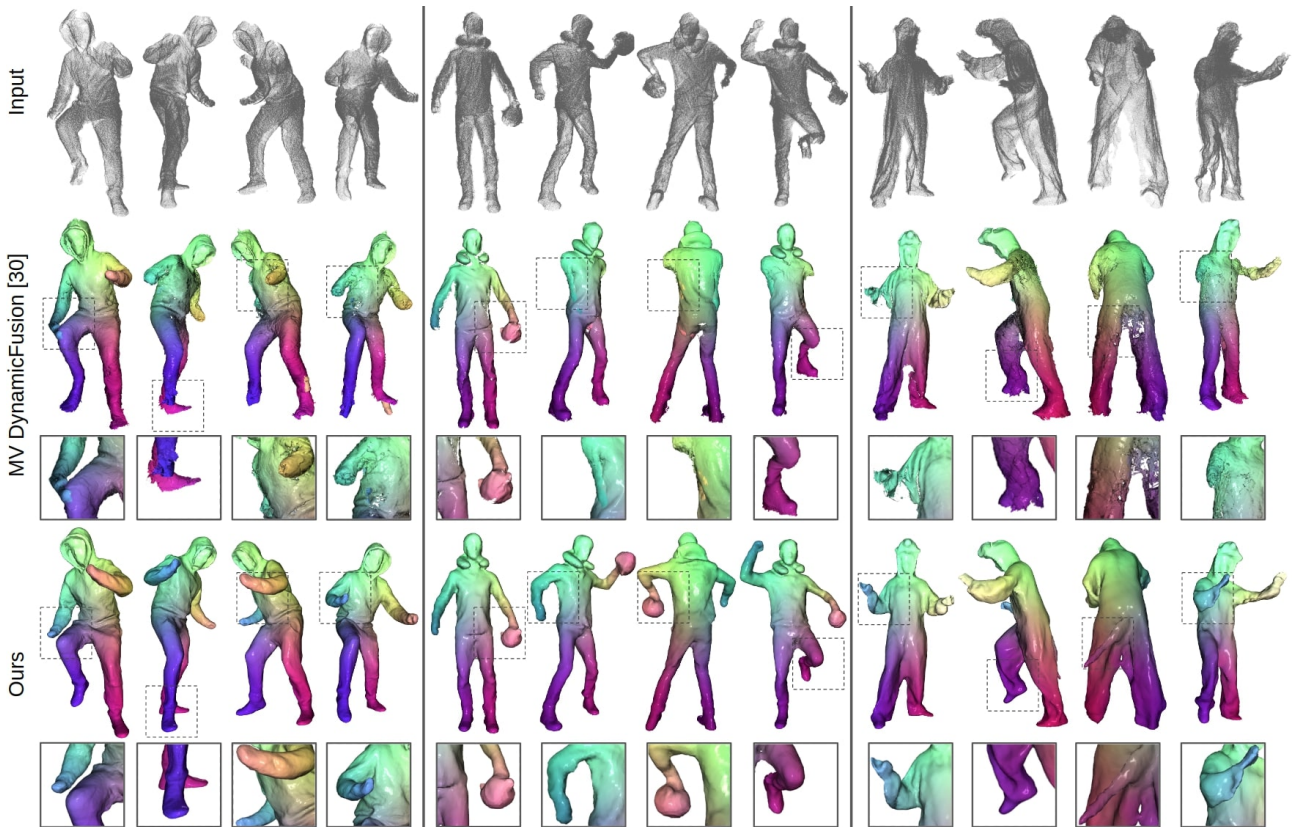


Figure 4: We show qualitative comparisons of our method with multi-view DynamicFusion [44] on real sequences captured by four Kinect Azure sensors. Colors represent corresponding locations in the first frame of the sequence to visualize the tracking quality and consistency.

losses), viewpoint consistency loss and surface consistency loss. As shown in Tab. 1, using these additional losses vastly improves the method’s performance. Especially, it results in a much lower EPE3D error, and, thus, in globally consistent tracking performance.

Limitations and Future Work. Using our globally consistent Neural Deformation Graph, we show state-of-the-art tracking and reconstruction quality. Currently, our quality is limited by the input, i.e., a 64^3 SDF grid. Sparse 3D convolutions [14] could be applied to cope with higher resolutions. Our approach focuses on the tracking and reconstruction of the geometry, and not the texture. A texture on top of the tracked geometry could be estimated (similar to the color scheme that we show in the results figures) and additional losses based on this texture could be employed. In an over-crowded setting, with many occlusions (e.g. some parts are never observed), or in a single-view setting, the self-supervised formulation might be under-constrained. A data-driven geometry prior could further improve the robustness of our approach. We believe that there is a potential for several high-impact follow-up works.

5. Conclusion

We introduced Neural Deformation Graph which allows to reconstruct and track non-rigidly deforming objects in a globally consistent fashion. It is enabled by a neural network that implicitly stores the deformation graph of the object. The network is trained with losses on global consistency, resulting in tracking and reconstruction quality that surpasses the state of the art by more than 60% w.r.t. the respective metrics. We believe that our global optimization of non-rigid motion will be a stepping stone to learn data-driven priors in the future.


Acknowledgments

TUM was supported by the ZD.B (Zentrum Digitalisierung Bayern), a TUM-IAS Rudolf Mößbauer Fellowship, the ERC Starting Grant Scan2CAD (804724), and the German Research Foundation (DFG) Grant Making Machine Learning on Static and Dynamic 3D Data Practical. FRLR was not involved in the data captures, evaluations, and implementation of the code. All rights belong to TUM.

References

- [1] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007. 3
- [2] Aljaž Božič, Pablo Palafox, Michael Zollhöfer, Angela Dai, Justus Thies, and Matthias Nießner. Neural non-rigid tracking. In *NeurIPS*, 2020. 2
- [3] Aljaž Božič, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2
- [4] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018. 1, 3
- [5] Yang Chen and Gérard Medioni. Object modeling by registration of multiple range images. *Image Vision Comput.*, 10:145–155, 01 1992. 7
- [6] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020. 3
- [7] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (ToG)*, 34(4):1–13, 2015. 2
- [8] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. 1, 3
- [9] Boyang Deng, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Neural articulated shape approximation. In *The European Conference on Computer Vision (ECCV)*. Springer, August 2020. 3, 5
- [10] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: Real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)*, 36(6):1–16, 2017. 2
- [11] Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 3d scanning deformable objects with a single rgb-d sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 493–501, 2015. 2
- [12] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020. 3, 6, 7
- [13] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3, 6, 7
- [14] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018. 8
- [15] Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, et al. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Transactions on Graphics (TOG)*, 38(6):1–19, 2019. 2
- [16] Kaiwen Guo, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai. Robust non-rigid motion tracking and surface reconstruction using l0 regularization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3083–3091, 2015. 6
- [17] Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Transactions on Graphics (TOG)*, 36(3):32, 2017. 2
- [18] Zeng Huang, Yuanlu Xu, Christoph Lassner, Hao Li, and Tony Tung. Arch: Animatable reconstruction of clothed humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3093–3102, 2020. 3
- [19] Matthias Innmann, Kihwan Kim, Jinwei Gu, Matthias Nießner, Charles Loop, Marc Stamminger, and Jan Kautz. Nrmvs: Non-rigid multi-view stereo. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2754–2763, 2020. 2
- [20] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016. 2
- [21] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 1
- [22] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 4
- [23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6
- [24] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnnp: An accurate o(n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009. 7
- [25] Yang Li, Aljaz Bozic, Tianwei Zhang, Yanli Ji, Tatsuya Harada, and Matthias Nießner. Learning to optimize non-rigid tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4910–4918, 2020. 2
- [26] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. 6
- [27] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3, 6, 7

- ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019. 6, 7
- [28] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 1, 2, 3
- [29] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87*, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery. 4
- [30] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020. 5
- [31] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Gnerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018. 1
- [32] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015. 1, 2, 3, 6, 7
- [33] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5379–5389, 2019. 6, 7
- [34] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 3
- [35] Vikramjit Sidhu, Edgar Tretschk, Vladislav Golyanik, Antonio Agudo, and Christian Theobalt. Neural dense non-rigid structure from motion with latent space constraints. In *European Conference on Computer Vision (ECCV)*, 2020. 2
- [36] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1121–1132, 2019. 3
- [37] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1386–1395, 2017. 2
- [38] Miroslava Slavcheva, Maximilian Baust, and Slobodan Ilic. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2646–2655, 2018. 2
- [39] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007. 2
- [40] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 papers*, pages 80–es. 2007. 2, 5
- [41] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2387–2395, 2016. 1
- [42] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt. Patchnets: Patch-based generalizable deep implicit 3d shape representations. In *European Conference on Computer Vision*, pages 293–309. Springer, 2020. 3
- [43] Edgar Tretschk, Ayush Tewari, Michael Zollhöfer, Vladislav Golyanik, and Christian Theobalt. Demea: Deep mesh autoencoders for non-rigidly deforming objects. *arXiv preprint arXiv:1905.10290*, 2019. 3
- [44] Sen Wang, Xinxin Zuo, Chao Du, Runxiao Wang, Jiangbin Zheng, and Ruigang Yang. Dynamic non-rigid objects reconstruction with a single rgb-d sensor. *Sensors*, 18(3):886, 2018. 2, 8
- [45] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*, 2020. 3



Neural Deformation Graphs for Globally-consistent Non-rigid Reconstruction

Conference Proceedings: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Author: Aljaž Božič

Publisher: IEEE

Date: June 2021

Copyright © 2021, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure C.3: Copyright authorization. Authorization for reuse of paper *Neural Deformation Graphs for Globally-consistent Non-rigid Reconstruction* in this dissertation.

C.4 TransformerFusion: Monocular RGB Scene Reconstruction using Transformers

COPYRIGHT

©2021 NeurIPS. Reprinted, with permission, from
Aljaž Božič, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner
TransformerFusion: Monocular RGB Scene Reconstruction using Transformers

Conference on Neural Information Processing Systems (NeurIPS) 2021

Details: NeurIPS website

SUMMARY

We introduce TransformerFusion, a transformer-based 3D scene reconstruction approach. From an input monocular RGB video, the video frames are processed by a transformer network that fuses the observations into a volumetric feature grid representing the scene; this feature grid is then decoded into an implicit 3D scene representation. Key to our approach is the transformer architecture that enables the network to learn to attend to the most relevant image frames for each 3D location in the scene, supervised only by the scene reconstruction task. Features are fused in a coarse-to-fine fashion, storing fine-level features only where needed, requiring lower memory storage and enabling fusion at interactive rates. The feature grid is then decoded to a higher-resolution scene reconstruction, using an MLP-based surface occupancy prediction from interpolated coarse-to-fine 3D features. Our approach results in an accurate surface reconstruction, outperforming state-of-the-art multi-view stereo depth estimation methods, fully-convolutional 3D reconstruction approaches, and approaches using LSTM- or GRU-based recurrent networks for video sequence fusion.

INDIVIDUAL CONTRIBUTIONS

Leading role in realizing the scientific project.

Problem definition	<i>significantly contributed</i>
Literature survey	<i>significantly contributed</i>
Implementation	<i>significantly contributed</i>
Experimental evaluation	<i>significantly contributed</i>
Preparation of the manuscript	<i>significantly contributed</i>

We include the accepted version of the original publication [16] in the following. Authors do not transfer the copyright of their papers to NeurIPS.

TransformerFusion: Monocular RGB Scene Reconstruction using Transformers

Aljaž Božič¹ Pablo Palafox¹ Justus Thies^{1,2} Angela Dai¹ Matthias Nießner¹

¹Technical University of Munich

²Max Planck Institute for Intelligent Systems, Tübingen, Germany
aljazbozic.github.io/transformerfusion

Abstract

We introduce TransformerFusion, a transformer-based 3D scene reconstruction approach. From an input monocular RGB video, the video frames are processed by a transformer network that fuses the observations into a volumetric feature grid representing the scene; this feature grid is then decoded into an implicit 3D scene representation. Key to our approach is the transformer architecture that enables the network to learn to attend to the most relevant image frames for each 3D location in the scene, supervised only by the scene reconstruction task. Features are fused in a coarse-to-fine fashion, storing fine-level features only where needed, requiring lower memory storage and enabling fusion at interactive rates. The feature grid is then decoded to a higher-resolution scene reconstruction, using an MLP-based surface occupancy prediction from interpolated coarse-to-fine 3D features. Our approach results in an accurate surface reconstruction, outperforming state-of-the-art multi-view stereo depth estimation methods, fully-convolutional 3D reconstruction approaches, and approaches using LSTM- or GRU-based recurrent networks for video sequence fusion.

1 Introduction

Monocular 3D reconstruction is a core task in 3D computer vision, aiming to reconstruct a complete and accurate 3D geometry of an object or an environment from only 2D observations captured by an RGB camera. A geometric understanding is key to applications such as robotic or autonomous vehicle navigation or interaction, as well as model creation and scene editing for augmented and virtual reality. In addition, geometric scene reconstructions form the basis for 3D scene understanding, supporting tasks such as 3D object detection, semantic, and instance segmentation [34, 35, 36, 29, 7, 43, 15, 16].

While state-of-the-art SLAM systems [3, 41] achieve robust and scale-accurate camera tracking leveraging both visual and inertial measurements, dense and complete 3D reconstruction of large-scale environments from monocular video remains a very challenging problem – particularly for interactive settings. Simultaneously, notable progress has been made on multi-view depth estimation, estimating depth from pairs of images by averaging features extracted from the images in a feature cost volume [42, 17, 19, 38, 13]. Unfortunately, averaging features across a full video sequence can lead to equal-weight treatment of each individual frame, despite some frames possibly containing less information in various regions (e.g., from motion blur, rolling shutter artifacts, very glancing or partial views of objects), making high-fidelity scene reconstruction challenging.

Inspired by the recent advances in natural language processing (NLP) that leverage transformer-based models for sequence to sequence modelling [40, 11, 2], we propose a transformer-based method that fuses a sequence of RGB input frames into a 3D representation of a scene at interactive rates. Key to

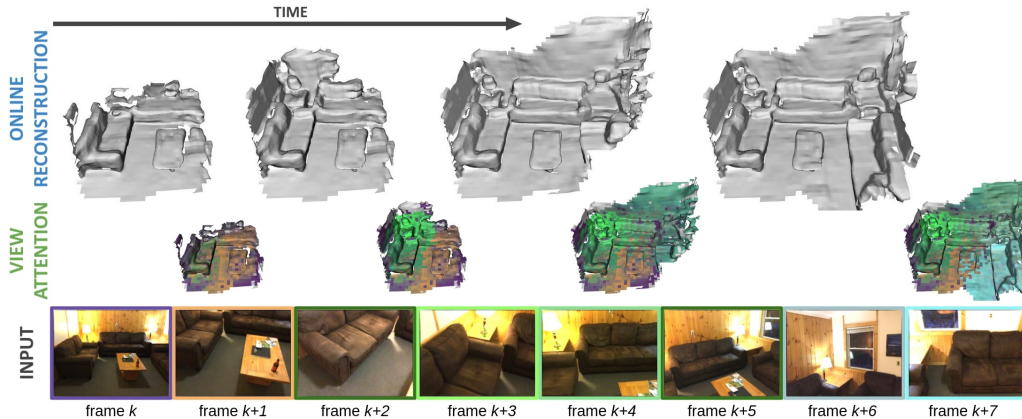


Figure 1: TransformerFusion is an online scene reconstruction method that takes a monocular RGB video as input. The features extracted from each observed image are fused incrementally with a transformer architecture. This fusion approach learns to attend to the most relevant image frames for each 3D location (see view attention color maps of the most relevant frame) achieving state-of-the-art reconstruction results.

our approach is a learned feature fusion of the video frames using a transformer-based architecture, which learns to attend to the most informative image features to reconstruct a local 3D region of the scene. A new observed RGB frame is encoded into a 2D feature map, and unprojected into a 3D volume, where our transformer learns a fused 3D feature for each location in the 3D volume from the image view features. This enables extraction of the most informative view features for each location in the 3D scene. The 3D features are fused in coarse-to-fine fashion, providing both improved reconstruction performance as well as interactive runtime. These features are then decoded into high-resolution scene geometry with an MLP-based surface occupancy prediction.

In summary, our main contributions to achieve robust and accurate scene reconstructions are:

- Learned multi-view feature fusion in the temporal domain using a transformer network that attends to only the most informative features of the image views for reconstructing each location in a scene.
- A coarse-to-fine hierarchy of our transformer-based feature fusion that enables an online reconstruction approach running at interactive frame-rates.

2 Related Work

Multi-view depth estimation. Estimating depth from multi-view image observations has been long-studied in computer vision. COLMAP [37] introduced a patch matching based approach which achieves impressive accuracy and remains established as one of the most popular methods for multi-view stereo. While COLMAP offers robust depth estimation for distinctive features in images, the patch matching struggles to densely reconstruct areas without many distinctive color features, such as floor and walls. Recently, learning-based approaches that build data-driven priors from large-scale datasets have improved depth estimation in these challenging scenarios. Some proposed methods rely only on a 2D network with multiple images concatenated as input [42]. Several recent approaches instead build a shared 3D feature cost volume in reference camera space using feature averaging [13, 17, 19, 25, 26]. These approaches estimate the reference frame’s depth within a local window of frames, but some also propagate information from previously estimated depth maps by using probabilistic filtering [25], a Gaussian process [17], or an LSTM bottleneck layer [13]. Such multi-view depth estimation approaches predict single-view depth maps, which must be fused together to construct a geometric 3D representation of the observed scene.

3D reconstruction from monocular RGB input. Multi-view depth estimation approaches can be combined with depth fusion approaches, such as volumetric fusion [6], to obtain a volumetric

reconstruction of the observed scene. MonoFusion [33] is one of the first methods using depth estimate from a real-time variant of PatchMatch stereo [1]. However, fusing noisy depth estimates causes artifacts in the 3D reconstruction, which lead to the development of recent approaches that directly predict the 3D surface reconstruction instead of per-frame depth estimates. One of the first approaches to predict 3D surface occupancy from two input RGB images is SurfaceNet [20], which converts volumetrically averaged colors into 3D surface occupancies using a 3D convolutional network. Atlas [28] extends this approach to a multi-view setting, while also leveraging learned features instead of colors. Recently, NeuralRecon [39] proposed a real-time 3D reconstruction framework, adding GRU units distributed in 3D to fuse reconstructions from different local windows of frames. Our approach also fuses together learned features from RGB frame input in an online fashion, but our transformer-based multi-view feature fusion enables relying only on the most informative features from the observed frames for a particular spatial location in the reconstructed scene, producing more accurate 3D reconstructions.

Transformers in computer vision. The transformer architecture [40] has achieved profound impact in many computer vision tasks in addition to its natural language processing origins. For a detailed survey, we refer the reader to [22]. In computer vision, transformers have been leveraged successfully for tasks such as object detection [4], video classification [44], image classification [12], image generation [30], and human reconstruction [45]. In this work, we propose transformer-based feature fusion for 3D scene reconstruction from a monocular video. Given a sequence of observed RGB frames, our approach learns to attend to the most informative features from each image to predict a dense occupancy field.

3 End-to-end 3D Reconstruction using Transformers

Given a set of N RGB images $I_i \in \mathbb{R}^{W \times H \times 3}$ of a scene with corresponding camera intrinsic parameters $\mathbf{K}_i \in \mathbb{R}^{3 \times 3}$ and extrinsic poses $\mathbf{P}_i \in \mathbb{R}^{4 \times 4}$, our method reconstructs the scene geometry by predicting occupancy values $o \in [0, 1]$ for every 3D point in the scene. Fig. 2 shows an overview of our approach. Each input image I_i is processed by a 2D convolutional encoder Θ , extracting coarse and fine image features (Φ_i^c and Φ_i^f , respectively):

$$\Theta : I_i \in \mathbb{R}^{W \times H \times 3} \mapsto (\Phi_i^c, \Phi_i^f)$$

From these 2D image features, we construct a 3D feature grid in world space. To this end, we regularly sample grid points in 3D at a coarse resolution of every $v_c = 30$ cm and a fine resolution of $v_f = 10$ cm. For these coarse and fine sample points, we query corresponding 2D features in all N images and predict fused coarse ψ^c and fine 3D features ψ^f using transformer networks [40]:

$$\begin{aligned} \mathcal{T}_c : (\Phi_1^c, \dots, \Phi_N^c) &\mapsto (\psi^c, w^c) \\ \mathcal{T}_f : (\Phi_1^f, \dots, \Phi_N^f) &\mapsto (\psi^f, w^f) \end{aligned}$$

Note that we also store the intermediate attention weights w^c and w^f of the first transformer layers for efficient view selection, which is explained in Sec. 3.4.

To further improve the features in the 3D spatial domain, we apply 3D convolutional networks \mathcal{C}_c and \mathcal{C}_f , at the coarse and fine level, respectively:

$$\begin{aligned} \mathcal{C}_c : \{\psi^c\}_{C \times C \times C} &\mapsto \{\tilde{\psi}^c\}_{C \times C \times C} \\ \mathcal{C}_f : \{(\tilde{\psi}^c, \psi^f)\}_{F \times F \times F} &\mapsto \{\tilde{\psi}^f\}_{F \times F \times F} \end{aligned}$$

Finally, to predict the scene geometry occupancy for a point $\mathbf{p} \in \mathbb{R}^3$, the coarse $\tilde{\psi}^c$ and fine features $\tilde{\psi}^f$ are trilinearly interpolated and a multi-layer perceptron \mathcal{S} maps these features to occupancies:

$$\mathcal{S} : (\tilde{\psi}^c, \tilde{\psi}^f) \mapsto o \in [0, 1]$$

This extraction of surface occupancies is inspired by convolutional occupancy networks [32] and IFNets [5]. From this occupancy field we extract a surface mesh with Marching cubes [27]. Note that in addition to surface occupancy, we also predict occupancy masks for near-surface locations at the coarse and fine levels. These masks are used for coarse-to-fine surface filtering (see Sec. 3.2), which

improves reconstruction performance with a focus on the surface geometry prediction and enables interactive runtime.

We train our approach in end-to-end fashion by supervising the surface occupancy predictions using the following loss:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_f + \mathcal{L}_o,$$

where \mathcal{L}_c and \mathcal{L}_f denote binary cross-entropy (BCE) losses on occupancy mask predictions for near-surface locations at the coarse and fine levels, respectively (see Sec. 3.2), and \mathcal{L}_o denotes a BCE loss for surface occupancy prediction (see Sec. 3.3).

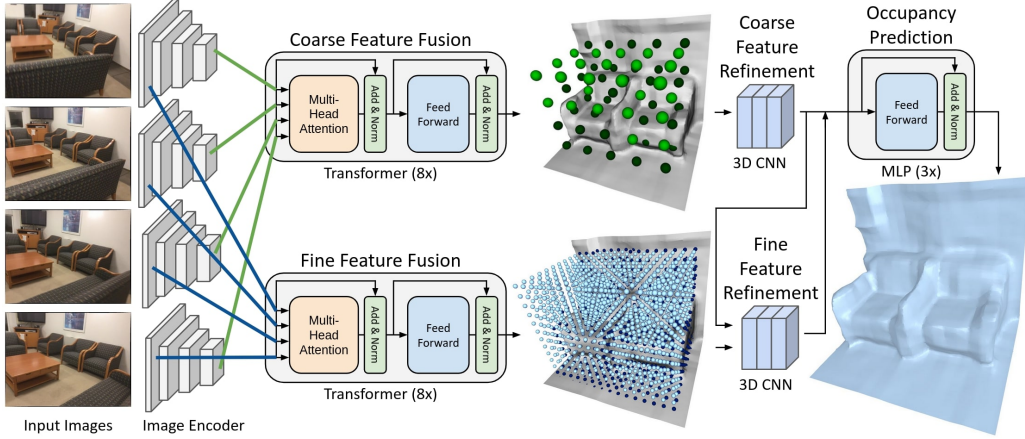


Figure 2: Method overview: given multiple input images, we compute coarse and fine level features. Using a transformer architecture, we separately fuse these coarse and fine features in a voxel grid. To improve the spatial features, we use a refinement network for both the coarse and the fine features. From these feature grids, we extract an occupancy field using a lightweight MLP.

3.1 Learning Temporal Feature Fusion via Transformers

For a spatial location $\mathbf{p} \in \mathbb{R}^3$ in the scene reconstruction, we learn to fuse coarse ψ^c and fine level features ψ^f from the N coarse and fine feature images (Φ_i^c and Φ_i^f , respectively), which are extracted by the 2D encoder Θ . Specifically, we train two instances of a transformer model, one for fusing coarse-level features ψ^c and one for fusing fine-level features ψ^f . Both transformers \mathcal{T}_c and \mathcal{T}_f share the same architecture. Thus, for simplicity, we omit the coarse and fine notation in the following.

Our transformer model \mathcal{T} is independently applied to each sample point in world space. For a point \mathbf{p} , the transformer network takes a series of 2D features ϕ_i as input that are bilinearly sampled from the feature maps Φ_i at the corresponding projective image location. The projective image location is computed via a full-perspective projection $\Pi_i(\mathbf{p}) = \pi(\mathbf{K}_i(\mathbf{R}_i\mathbf{p} + \mathbf{t}_i))$, assuming known camera intrinsics \mathbf{K}_i and extrinsics $\mathbf{P}_i = (\mathbf{R}_i, \mathbf{t}_i)$. To inform the transformer about invalid features (i.e., a sample point is projected outside an image), we also provide the pixel validity $v_i \in \{0, 1\}$ as input. In addition to these 2D features ϕ_i , we concatenate the projected depth $d_i = (\mathbf{R}_i\mathbf{p} + \mathbf{t}_i)_z$, and the viewing ray $\mathbf{r}_i = (\mathbf{p} - \mathbf{c}_i) / \|\mathbf{p} - \mathbf{c}_i\|_2$ to the input ($\mathbf{c}_i \in \mathbb{R}^3$ denoting the camera center of view i). These input features are converted to an embedding vector $\theta_i \in \mathbb{R}^D$ using a linear layer $\theta_i = \text{FCN}(\phi_i, d_i, v_i, \mathbf{r}_i)$, before feeding it into the transformer network that then predicts a fused feature $\psi \in \mathbb{R}^D$:

$$\mathcal{T} : (\theta_1, \dots, \theta_N) \mapsto (\psi, w)$$

As described above, w denotes the attention values of the initial attention layer, which are used for view selection to speed-up fusion (see Sec. 3.4).

Transformer architecture. We followed [12] when designing the transformer architecture \mathcal{T} . It consists of 8 modules of feed-forward and attention layers, using multi-head attention with 4 attention heads and embedding dimension $D = 256$. Feed-forward layers process the temporal inputs independently, and contain ReLU activation, linear layers with residual connection, and layer norm.

The model returns both fused feature $\psi \in \mathbb{R}^D$ and attention weights $w \in \mathbb{R}^N$ over all temporal inputs from the initial attention layer that are later used for selecting which views to maintain over longer sequences of input image views.

3.2 Spatial Feature Refinement

While the transformer network fuses 2D observations in the temporal domain, we additionally imbue explicit spatial reasoning by applying a 3D CNN to spatially refine the fused features $\{\psi^c\}_{C \times C \times C}$ and $\{\psi^f\}_{F \times F \times F}$ that are computed by the transformers \mathcal{T}_c and \mathcal{T}_f on the coarse and fine grid, respectively. The coarse features $\{\psi^c\}_{C \times C \times C}$ are refined by a 3D CNN \mathcal{C}_c consisting of 3 residual blocks that maintain the same spatial resolution and produce refined features $\{\tilde{\psi}^c\}_{C \times C \times C}$. These features are upsampled to a fine grid resolution using nearest-neighbor upsampling, and concatenated with fused features at fine level $\{\psi^f\}_{F \times F \times F}$. A fine-level 3D CNN \mathcal{C}_f is then applied to the concatenated features, resulting in refined fine features $\{\tilde{\psi}^f\}_{F \times F \times F}$. Both, coarse $\tilde{\psi}^c$ and fine features $\tilde{\psi}^f$ are used for surface occupancy prediction.

Coarse-to-fine surface filtering. The refined features are also used to predict occupancy masks for near-surface locations at both coarse and fine levels, thus, filtering out free-space regions and sparsifying the volume, such that the higher-resolution and computationally expensive fine-scale surface extraction is performed only in regions close to the surface. To achieve this, additional 3D CNN layers \mathcal{M}_c and \mathcal{M}_f are applied to the refined features, outputting a near-surface mask $m^c, m^f \in [0, 1]$ for every grid point:

$$\begin{aligned}\mathcal{M}_c &: \{\tilde{\psi}^c\}_{C \times C \times C} \mapsto \{m^c\}_{C \times C \times C} \\ \mathcal{M}_f &: \{\tilde{\psi}^f\}_{F \times F \times F} \mapsto \{m^f\}_{F \times F \times F}\end{aligned}$$

Only spatial regions where both m^c and m^f are larger than 0.5, i.e., close to the surface, are processed further to compute the final surface reconstruction; other regions are determined to be free space. This improves the overall reconstruction performance by focusing the capacity of the surface prediction network to close-to-the-surface regions and enables a significant runtime speed-up.

Intermediate supervision of near-surface masks m^c and m^f is employed using masks m_{gt}^c and m_{gt}^f generated from the ground truth scene reconstruction, denoting the grid point as near-surface if there exists ground truth surface in the radius of v_c or v_f from the point. Binary cross entropy losses $\mathcal{L}_c = \text{BCE}(m^c, m_{\text{gt}}^c)$ and $\mathcal{L}_f = \text{BCE}(m^f, m_{\text{gt}}^f)$ are applied.

3.3 Surface Occupancy Prediction

The final surface reconstruction is predicted by decoding the coarse and fine feature grids to occupancy values $o \in [0, 1]$, with values $o \geq 0.5$ representing occupied points and values $o < 0.5$ representing free-space points. For a point $\mathbf{p} \in \mathbb{R}^3$, we compute its feature representation by trilinearly interpolating coarse and fine grid features:

$$\begin{aligned}\psi_{\mathbf{p}}^c &= \text{Trilinear}(\mathbf{p}, \{\tilde{\psi}^c\}_{C \times C \times C}) \\ \psi_{\mathbf{p}}^f &= \text{Trilinear}(\mathbf{p}, \{\tilde{\psi}^f\}_{F \times F \times F})\end{aligned}$$

We concatenate the interpolated features and predict the point’s occupancy as $o = \mathcal{S}(\psi_{\mathbf{p}}^c, \psi_{\mathbf{p}}^f)$, where \mathcal{S} is a multi-layer perceptron (MLP) with 3 modules of feed-forward layers, containing ReLU activation, linear layer with residual connection, and layer norm.

Surface occupancy supervision. We train on $1.5 \times 1.5 \times 1.5$ m volumetric chunks of scenes for training efficiency. To supervise the surface occupancy loss, 1k points are sampled inside the chunk, with 80% of samples drawn from a truncation region at most 10 cm from the surface, and 20% sampled uniformly inside the chunk. Ground truth occupancy values o_{gt} are computed using the ScanNet RGB-D reconstructions [8]. For uniform samples it is straightforward to generate unoccupied point samples by sampling points in free space in front of the visible surface, but it is unknown whether a point sample is occupied when it lies behind seen surfaces. In order to prevent artifacts behind walls, we follow the data processing applied in [28] and additionally label point samples as occupied, if they are sampled in areas where an entire vertical column of voxels is occluded in the scene. A binary cross entropy loss $\mathcal{L}_o = \text{BCE}(o, o_{\text{gt}})$ is then applied to the occupancy predictions o .

3.4 View Selection for Online Scene Reconstruction

We aim to consider all N frames as input to our transformer for each 3D location in a scene; however, this becomes extremely computationally expensive with long videos or large-scale scenes, which prohibits online scene reconstruction. Instead, we proceed with the reconstruction incrementally, processing every video frame one-by-one, while keeping only a small number $K = 16$ of measurements for every 3D point. We visualize this online approach in Fig. 1.

During training, for efficiency, we use only K_t random images for each training volume. At test time, we leverage the attention weights w^c and w^f of the initial transformer layers to determine which views to keep in the set of K measurements. Specifically, for a new RGB frame, we extract its 2D features, and run feature fusion for every coarse and fine grid point inside the camera frustum. This returns the fused feature and also the attention weights over all currently accumulated input measurements. Whenever the maximum number of K measurements is reached, a selection is made by dropping out a measurement with lowest attention weight before adding new measurements in the latest frame. This guarantees a low number of input measurements, speeding up fusion processing times considerably. Furthermore, by using coarse-to-fine filtering, described in Sec. 3.2, we can further accelerate fusion by only considering higher resolution points in the area near the estimated surface. Together with incremental processing that results in high performance benefits, our approach performs per-frame feature fusion at about 7 FPS despite an unoptimized implementation.

3.5 Training Scheme

Our approach has been implemented using the PyTorch library [31]. The architecture details of the used networks are specified in the supplemental document. To train our approach we use ScanNet dataset [8], an RGB-D dataset of indoor apartments. We follow the established train-val-test split. For training, we randomly sample $1.5 \times 1.5 \times 1.5$ m volume chunks of the train scenes, sampling less chunks in free space and more samples in areas with non-structural objects, i.e. not only consisting of floor or walls. This results in ≈ 165 k training chunks. For each chunk, we randomly sample $K_t = 8$ RGB images among all frames that include the chunk in their camera frustums.

The 2D convolutional encoder Θ for image feature extraction is implemented as a ResNet-18 [14] network, pre-trained on ImageNet [24]. During training, a batch size of 4 chunks is used with an Adam [23] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and weight regularization of 10^{-4} . We use a learning rate of 10^{-4} with 5k warm-up steps at initialization, and square root learning rate decay afterwards. When computing the losses of coarse and fine surface filtering predictions, a higher weight of 2.0 is applied to near-surface voxels, to increase recall and improve overall robustness. Training takes about 30 hours using an Intel Xeon 6242R Processor and an Nvidia RTX 3090 GPU.

4 Experiments

Metrics. To evaluate our monocular scene reconstruction, we use several measures of reconstruction performance. We evaluate geometric accuracy and completion, with accuracy measuring the average point-to-point error from predicted to ground truth vertices, completion measuring the error in the opposite direction, and chamfer as the average of accuracy and completion (in cm). To account for possibly different mesh resolutions among methods, we uniformly sample 200k points over mesh faces of every reconstructed mesh. Additionally, we threshold these point-to-point errors and compute precision and recall by computing the ratio of point-to-point matches within distance ≤ 5 cm. Since it is easy to maximize either precision (by predicting only a few but accurate points) or recall (by over-completing reconstructions with noisy surface), we found the most reliable metric to be F-score, determined by both precision and recall.

Our ground truth reconstructions are obtained by automated 3D reconstruction [9] from RGB-D videos of real-world environments and, thus, they are often incomplete due to unobserved and occluded regions in the scene. To avoid penalizing methods for reconstructing a more complete scene w.r.t. the available ground truth, we apply an additional occlusion mask at evaluation.

As most state of the art, particularly for depth estimation, rely on a pre-sampled set of keyframes (based on sufficient translation or rotation difference between camera poses), we evaluate all approaches based on sequences of sampled keyframes, using the keyframe selection of [13].

Table 1: Quantitative comparison with baselines and ablations on test set of Scannet dataset [8].

Method	Acc ↓	Compl ↓	Chamfer ↓	Prec ↑	Recall ↑	F-score ↑
RevisitingSI [18]	14.29	16.19	15.24	0.346	0.293	0.314
MVDepthNet [42]	12.94	8.34	10.64	0.443	0.487	0.460
GPMVS [17]	12.90	8.02	10.46	0.453	0.510	0.477
ESTDepth [26]	12.71	7.54	10.12	0.456	0.542	0.491
DPSNet [19]	11.94	7.58	9.77	0.474	0.519	0.492
DELTAS [38]	11.95	7.46	9.71	0.478	0.533	0.501
DeepVideoMVS [13]	10.68	6.90	8.79	0.541	0.592	0.563
COLMAP [37]	10.22	11.88	11.05	0.509	0.474	0.489
NeuralRecon [39]	5.09	9.13	7.11	0.630	0.612	0.619
Atlas [28]	7.16	7.61	7.38	0.675	0.605	0.636
Ours: w/o TRSF, avg	7.23	9.74	8.48	0.635	0.501	0.557
Ours: w/o TRSF, weight	6.11	11.12	8.61	0.686	0.512	0.583
Ours: w/o TRSF, conv	6.56	9.84	8.20	0.661	0.524	0.582
Ours: w/o spatial ref.	10.46	16.91	13.68	0.479	0.295	0.361
Ours: w/o C2F filter	6.57	7.69	7.13	0.678	0.592	0.631
Ours: w/o proj. depth	8.06	10.02	9.04	0.594	0.475	0.525
Ours: w/o viewing ray	5.71	8.59	7.15	0.706	0.559	0.621
Ours: 30 cm voxel size	7.92	17.33	12.63	0.491	0.258	0.335
Ours: 15 cm voxel size	5.79	9.62	7.71	0.686	0.520	0.589
Ours: 4 images, RND	8.01	10.28	9.15	0.587	0.445	0.502
Ours: 4 images	6.80	8.40	7.60	0.661	0.524	0.581
Ours: 8 images, RND	6.74	8.55	7.64	0.665	0.544	0.596
Ours: 8 images	6.17	7.69	6.93	0.704	0.584	0.636
Ours: 16 images, RND	5.80	8.56	7.18	0.711	0.584	0.638
Ours	5.52	8.27	6.89	0.728	0.600	0.655

4.1 Comparison with State of the Art

In Tab. 1, we compare our approach with state-of-the-art methods. All methods are trained on the ScanNet dataset [8], using the official train/val/test split. We use the pre-trained models provided by the authors for MVDepthNet [42], GPMVS [17] and DPSNet [19] which are fine-tuned on ScanNet. For baselines that predict depth in a reference camera frame instead of directly reconstructing 3D surface, a volumetric fusion method [6] is used to fuse different depth maps into a 3D truncated signed distance field. The single-view depth prediction method RevisitingSI [18] suffers from the more challenging task formulation without the use of multiple views, leading to noisier depth predictions and inconsistencies between frames. Multi-view depth estimation methods leverage the additional view information for improved performance, with the LSTM-based approach of DeepVideoMVS [13] achieving the best performance among these approaches. Reconstruction quality further improves with methods that directly predict the 3D surface geometry, such as NeuralRecon [39] and Atlas [28]. Our transformer-based feature fusion approach enables more robust reconstruction and outperforms all existing methods in both chamfer distance and F-score. The performance improvement can also be clearly seen in the qualitative comparisons in Fig. 3.

4.2 Ablations

To demonstrate the effectiveness of our design choices, we conducted a quantitative ablation study which is shown in Tab. 1 and discussed in the following.

What is the impact of learning to fuse features from different views with transformers? We evaluate the effect of our learned feature fusion by replacing the transformer blocks with a multi-layer perceptron (MLP) that processes input image observations independently. The per-view outputs of this MLP are fused using an average (*w/o TRSF, avg*) or using a weighted average with weights

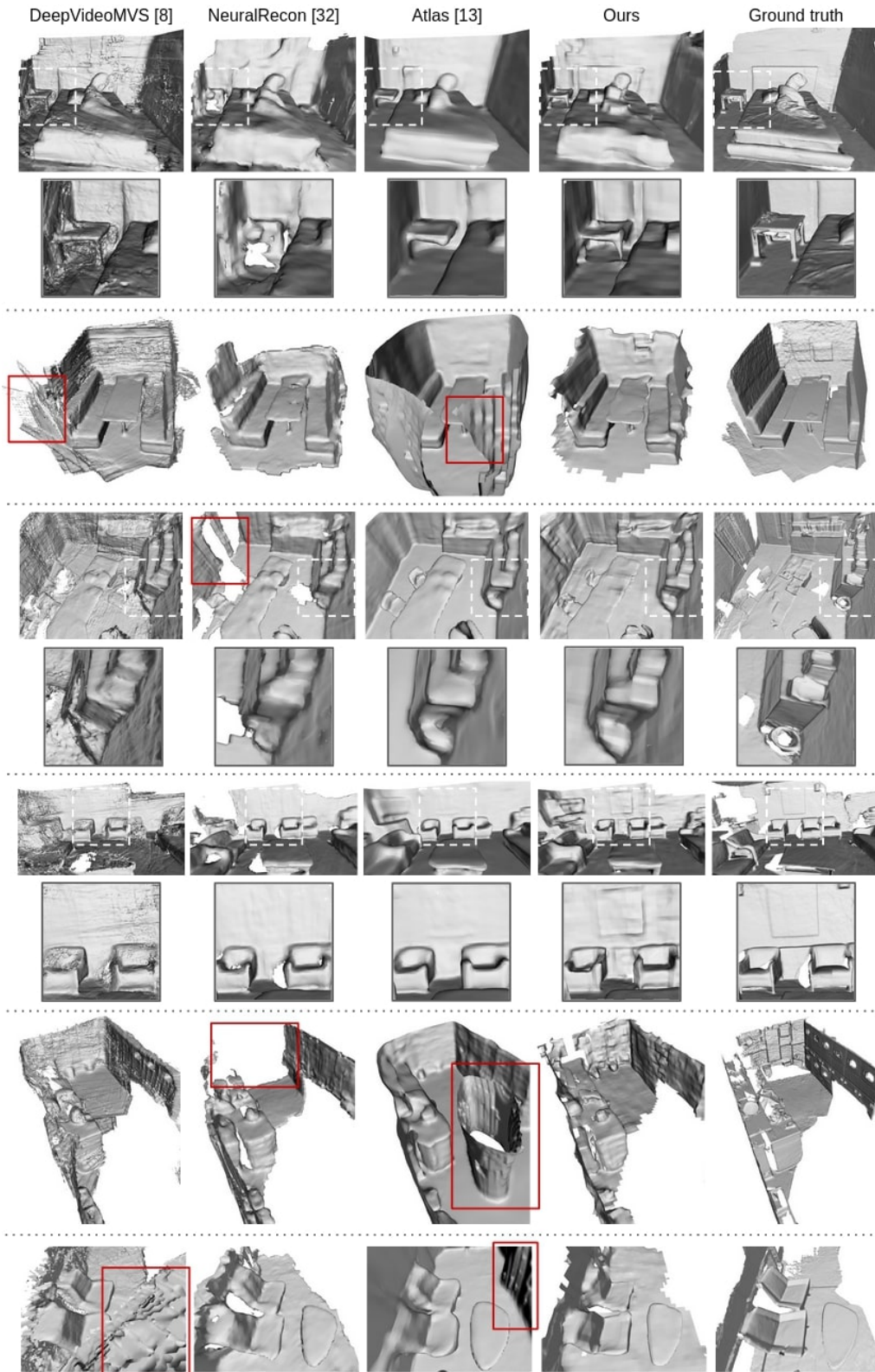


Figure 3: Qualitative comparison of scene reconstructions on test set of ScanNet dataset [8]; note that only RGB input is used by each method while the ground truth is reconstructed using the input depth.

predicted by the MLP (*w/o TRSF, weight*). Additionally, we implemented convolutional feature fusion, using a 1-dimensional CNN that processes features in temporal domain and predicts fused features (*w/o TRSF, conv*). We find that our transformer-based view fusion effectively learns to attend to the most informative views for a specific location, resulting in significantly improved performance over these feature fusion alternatives.

Does spatial feature refinement help reconstruction performance? Spatial feature refinement is indeed very important for reconstruction quality. It enables the model to aggregate feature information in spatial domain and produce more spatially consistent and complete reconstructions, without it (*w/o spatial ref.*) the geometry completion (and recall metric) are considerably worse.

How important is coarse-to-fine filtering? Predicting the coarse and fine near-surface masks provides an additional performance improvement compared to the model without it (*w/o C2F filter*), as it allows more focus on surface geometry. Furthermore, this enables a speed-up of the fusion runtime by a factor of approximately 3.5, resulting in processing times of 7 FPS (instead of 2 FPS).

Are additional inputs to the transformer networks needed? Existing reconstruction approaches [39, 28] aggregate 2D features using a simple average operation. In comparison, our approach uses a transformer to learn the feature fusion. That makes it possible to use additional inputs that don't support a straight-forward average operation, but could be very informative for the task of multi-view surface reconstruction, such as projected depth and viewing ray. In Tab. 1 we conducted an additional quantitative ablation study w.r.t. the input to the transformer networks. Both the projected depth as well as the view ray help the transformer to better fuse the features for the task of 3D reconstruction.

How does voxel size of feature grids influence reconstruction performance? We compared the reconstruction performance when using different voxel sizes for the feature grid. We only varied fine feature grid resolution, voxel size of coarse grid was always 30 cm. More specifically, we replaced the voxel size of 10 cm at the fine grid level with 30 cm and 15 cm. In both cases, the performance decreased considerably; i.e., the higher the resolution, the better the results. That is reflected also in qualitative comparison in the supplemental document.

How many views should be used for feature fusion? In our experiments, we use a limited number of $K = 16$ frame observations to inform the feature for every 3D grid location. We find that these views all contribute, with performance degrading somewhat with sparser sets of observations ($K = 8$ or $K = 4$). The number of frames is limited because of execution time and memory consumption for bigger scenes.

How effective is frame selection using attention weights? The K frames for each 3D grid feature are selected based on the computed attention weights and are updated during scanning. To evaluate this frame selection, we compare against a frame selection scheme that randomly selects frames that observe the 3D location (*RND*), which results in a noticeable drop in performance for both chamfer and F-score. The performance difference is even larger when using less views for fusion ($K = 8$ or $K = 4$), where view selection becomes even more important. In Fig. 1, we visualize the most important view for locations in the scene, selected by the highest attention weight. Relatively smooth transitions between selected views among neighboring 3D locations suggest that view selection is spatially consistent. To illustrate the frame selection, we also visualize all selected frames with corresponding attention weights for specific 3D locations in the supplemental document.

4.3 Limitations

Under severe occlusions and partial observation of the scene, our method can struggle to reconstruct details of certain objects, such as chair legs, monitor stands, or books on the shelves. Furthermore, transparent objects, such as glass windows without frames, are often inaccurately reconstructed as empty space. We show qualitative examples of these failure cases in Fig. 4. These challenging scenarios are often not properly reconstructed even when using ground truth RGB-D data, and we believe that using self-supervised losses [10] for monocular scene reconstruction could be an interesting future research direction. Additionally, higher resolution geometric fidelity could potentially be achieved by sparse operations in 3D or learning local geometric priors on detailed synthetic data [21].

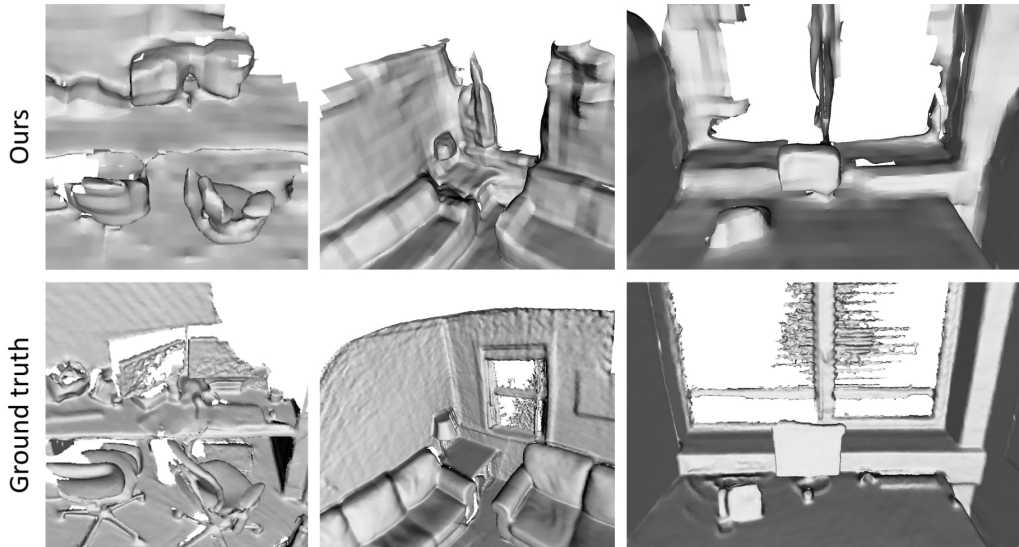


Figure 4: Limitations of our approach are the lack of detail at partially observed and occluded objects, and inaccurate reconstruction of transparent surfaces, such as glass windows.

5 Conclusion

We introduced TransformerFusion for monocular 3D scene reconstruction, leveraging a new transformer-based approach for online feature fusion from RGB input views. A coarse-to-fine formulation of our transformer-based feature fusion improves the effective reconstruction performance as well as the runtime. Our feature fusion learns to exploit the most informative image view features for geometric reconstruction, achieving state-of-the-art reconstruction performance. We believe that our interactive scanning approach provides exciting avenues for future research, and enables new possibilities in learning multi-view perception and 3D scene understanding.

Broader Impact

Our work proposes a novel monocular scene reconstruction approach that can be used for applications in the field of augmented and virtual reality, and also serves as a basis for 3D scene understanding from monocular RGB input, enabling navigation of autonomous agents in unknown environments. Being a building block for these applications, we need to be aware of the potential negative societal impacts of some applications, such as the improper use of autonomous robots in military, or labor market disruptions as a consequence of job automation. Since our approach is data-driven, using RGB-D data as supervision, we also need to be aware of related privacy concerns when capturing new datasets for 3D reconstruction.

Acknowledgments

This project is funded by the Bavarian State Ministry of Science and the Arts and coordinated by the Bavarian Research Institute for Digital Transformation (bidt), a TUM-IAS Rudolf Mößbauer Fellowship, the ERC Starting Grant Scan2CAD (804724), and the German Research Foundation (DFG) Grant Making Machine Learning on Static and Dynamic 3D Data Practical.

References

- [1] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11, pages 1–11, 2011.
- [2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam. *arXiv preprint arXiv:2007.11898*, 2020.
- [4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.
- [5] J. Chibane, T. Alldieck, and G. Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.
- [6] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.
- [7] A. Dai and M. Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 452–468, 2018.
- [8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.
- [9] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1, 2017.
- [10] A. Dai, C. Diller, and M. Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2020.
- [11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderoeder, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [13] A. Düzçeker, S. Galliani, C. Vogel, P. Speciale, M. Dusmanu, and M. Pollefeys. Deepvideomvs: Multi-view stereo on video with recurrent spatio-temporal fusion, 2020.
- [14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [15] J. Hou, A. Dai, and M. Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4421–4430, 2019.
- [16] J. Hou, A. Dai, and M. Nießner. Revealnet: Seeing behind objects in rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2098–2107, 2020.
- [17] Y. Hou, J. Kannala, and A. Solin. Multi-view stereo by temporal nonparametric fusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2651–2660, 2019.
- [18] J. Hu, M. Ozay, Y. Zhang, and T. Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. 2019.
- [19] S. Im, H.-G. Jeon, S. Lin, and I. S. Kweon. Dpsnet: End-to-end deep plane sweep stereo. *arXiv preprint arXiv:1905.00538*, 2019.
- [20] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang. Surfacer-net: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2307–2315, 2017.
- [21] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020.
- [22] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021.
- [23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

- [24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
- [25] C. Liu, J. Gu, K. Kim, S. G. Narasimhan, and J. Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10986–10995, 2019.
- [26] X. Long, L. Liu, W. Li, C. Theobalt, and W. Wang. Multi-view depth estimation using epipolar spatio-temporal network. *CVPR*, 2021.
- [27] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.
- [28] Z. Murez, T. van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich. Atlas: End-to-end 3d scene reconstruction from posed images. In *ECCV*, 2020. URL <https://arxiv.org/abs/2003.10432>.
- [29] Y. Nie, J. Hou, X. Han, and M. Nießner. Rfd-net: Point scene understanding by semantic instance reconstruction. *arXiv preprint arXiv:2011.14744*, 2020.
- [30] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.
- [31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.
- [32] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, Cham, Aug. 2020. Springer International Publishing.
- [33] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche. Monofusion: Real-time 3d reconstruction of small scenes with a single web camera. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 83–88. IEEE, 2013.
- [34] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
- [35] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.
- [36] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019.
- [37] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, 2016.
- [38] A. Sinha, Z. Murez, J. Bartolozzi, V. Badrinarayanan, and A. Rabinovich. Deltas: Depth estimation by learning triangulation and densification of sparse points. In *ECCV*, 2020. URL <https://arxiv.org/abs/2003.08933>.
- [39] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *CVPR*, 2021.
- [40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.
- [41] L. von Stumberg, V. Usenko, and D. Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. In *International Conference on Robotics and Automation (ICRA)*, May 2018.
- [42] K. Wang and S. Shen. Mvdepthnet: Real-time multiview depth estimation neural network. In *2018 International conference on 3d vision (3DV)*, pages 248–257. IEEE, 2018.
- [43] W. Wang, R. Yu, Q. Huang, and U. Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018.
- [44] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.
- [45] P. Zins, Y. Xu, E. Boyer, S. Wuhrer, and T. Tung. Learning implicit 3d representations of dressed humans from sparse views. *arXiv preprint arXiv:2104.08013*, 2021.

Who holds the Copyright on a NeurIPS paper

According to U.S. Copyright Office's page, **What is a Copyright**, when you create an original work you are the author and the owner and hold the copyright, unless you have an agreement to transfer the copyright to a third party such as the company or school you work for.

Authors do not transfer the copyright of their papers to NeurIPS. Instead, they grant NeurIPS a non-exclusive, perpetual, royalty-free, fully-paid, fully-assignable license to copy, distribute and publicly display all or part of the paper.

Figure C.4: Copyright authorization. Authorization for reuse of paper *TransformerFusion: Monocular RGB Scene Reconstruction using Transformers* in this dissertation.

D Differentiable Non-rigid Optimization

In this chapter we provide more information about the differentiable components of our Neural Non-rigid Tracker, or more specifically, we describe in more detail how to make the non-rigid tracking formulation from Sec. 4.4.3 differentiable. Given a deformation graph with an edge set \mathcal{E} , and a set of predicted correspondences \mathcal{C} , we want to estimate the deformation parameters \mathcal{T} . Our non-rigid optimization is based on the Gauss-Newton algorithm and minimizes an energy formulation that is based on three types of residual components: the 2D reprojection term, the depth term and the regularization term of the non-rigid deformation.

For a pixel $\mathbf{u} \in \Pi_s \subset \mathbb{R}^2$ and graph edge $(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E}$, we define such terms as:

$$\begin{aligned} r_{2D}^{\mathbf{u}}(\mathcal{T}) &= w_{\mathbf{u}}(\pi_{\mathcal{C}}(\mathbf{Q}(\mathbf{p}_{\mathbf{u}}, \mathcal{T})) - \mathbf{c}_{\mathbf{u}}) \\ r_{\text{depth}}^{\mathbf{u}}(\mathcal{T}) &= w_{\mathbf{u}}([\mathbf{Q}(\mathbf{p}_{\mathbf{u}}, \mathcal{T})]_z - [\mathbf{P}_t(\mathbf{c}_{\mathbf{u}})]_z) \\ r_{\text{reg}}^{\mathbf{v}_i, \mathbf{v}_j}(\mathcal{T}) &= e^{\hat{\omega}_{\mathbf{v}_i}}(\mathbf{v}_j - \mathbf{v}_i) + \mathbf{v}_i + \mathbf{t}_{\mathbf{v}_i} - (\mathbf{v}_j + \mathbf{t}_{\mathbf{v}_j}) \end{aligned}$$

where $\mathbf{c}_{\mathbf{u}} \in \mathbb{R}^2$ and $w_{\mathbf{u}} \in \mathbb{R}$ represent the predicted correspondence and the importance weight, respectively; and $\mathbf{p}_{\mathbf{u}} = \pi_{\mathcal{C}}^{-1}(\mathbf{u}, d_{\mathbf{u}})$ is a 3D point corresponding to the pixel \mathbf{u} with depth value $d_{\mathbf{u}}$. To construct the residual vector \mathbf{r} , we stack together 2D reprojection and depth terms for each correspondence $\mathbf{c}_{\mathbf{u}} \in \mathcal{C}$, and regularization terms for every graph edge $(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E}$.

The Gauss-Newton method is an iterative scheme. In every iteration n , we compute the Jacobian matrix \mathbf{J}_n and the residual vector \mathbf{r}_n , and get a solution increment $\Delta\mathcal{T}$ by solving the normal equations:

$$\mathbf{J}_n^T \mathbf{J}_n \Delta\mathcal{T} = -\mathbf{J}_n^T \mathbf{r}_n$$

In Sec. D.1 we describe the construction of the Jacobian matrix $\mathbf{J} \in \mathbb{R}^{(3|\mathcal{C}|+3|\mathcal{E}|) \times 6N}$, consisting of partial derivatives of the residual vector $\mathbf{r}^{3|\mathcal{C}|+3|\mathcal{E}|}$ with respect to deformation parameters $\mathcal{T} = (\omega_{\mathbf{v}_1}, \mathbf{t}_{\mathbf{v}_1}, \dots, \omega_{\mathbf{v}_N}, \mathbf{t}_{\mathbf{v}_N}) \in \mathbb{R}^{6N}$. The linear system is solved using LU decomposition. To enable differentiation through the entire Gauss-Newton solver, we have to ensure that the linear solve is differentiable. The differentiable linear solve operation is detailed in Sec D.2.

D.1 Partial Derivatives

As described in Sec. 4.3, decompose the rotation matrix into $e^{\hat{\omega}_{\mathbf{v}_i}} = e^{\hat{\boldsymbol{\epsilon}}_{\mathbf{v}_i}} \mathbf{R}_{\mathbf{v}_i}$ with $\boldsymbol{\epsilon}_{\mathbf{v}_i} = 0$, and instead of $\hat{\omega}_{\mathbf{v}_i}$ we optimize the rotation increment $\hat{\boldsymbol{\epsilon}}_{\mathbf{v}_i}$. In the following, we derive the partial derivatives of the residual vector \mathbf{r} with respect to $\boldsymbol{\epsilon}_{\mathbf{v}_i}$ and $\mathbf{t}_{\mathbf{v}_i}$ of every node \mathbf{v}_i , to construct the Jacobian matrix \mathbf{J} .

Appendix

To simplify notation, we define the rotation operator that takes as input an angular velocity vector $\boldsymbol{\epsilon} \in \mathbb{R}^3$, rotation matrix $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ and point $\mathbf{p} \in \mathbb{R}^3$ and outputs the rotated point:

$$\mathbf{R}(\boldsymbol{\epsilon}, \mathbf{R}, \mathbf{p}) = e^{\widehat{\boldsymbol{\epsilon}}} \mathbf{R} \mathbf{p}$$

To compute the partial derivative with respect to $\boldsymbol{\epsilon}$, we follow the derivation from [26]:

$$\left. \frac{\partial \mathbf{R}(\boldsymbol{\epsilon}, \mathbf{R}, \mathbf{p})}{\partial \boldsymbol{\epsilon}} \right|_{\boldsymbol{\epsilon}=\mathbf{0}} = -\widehat{\mathbf{R} \mathbf{p}}$$

Here, the $\widehat{\cdot}$ -operator creates a 3×3 skew-symmetric matrix from a 3-dimensional vector. For a vector $\mathbf{a} = (a_1, a_2, a_3)^T$, its skew symmetric matrix $\widehat{\mathbf{a}}$ is defined as:

$$\widehat{\mathbf{a}} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

The rotation operator $\mathbf{R}(\boldsymbol{\epsilon}, \mathbf{R}, \mathbf{p})$ is a core part of the warping operator $\mathbf{Q}(\mathbf{p}, \mathcal{T})$. It follows that partial derivatives of a warping operator $\mathbf{Q}(\mathbf{p}, \mathcal{T})$ with respect to $\boldsymbol{\epsilon}_{\mathbf{v}_i}$ and $\mathbf{t}_{\mathbf{v}_i}$ for every node \mathbf{v}_i can be computed as

$$\begin{aligned} \frac{\partial \mathbf{Q}(\mathbf{p}, \mathcal{T})}{\partial \boldsymbol{\epsilon}_{\mathbf{v}_i}} &= -\alpha_{\mathbf{v}_i}^{\mathbf{p}} \widehat{\mathbf{R}_{\mathbf{v}_i}(\mathbf{p} - \mathbf{v}_i)} \\ \frac{\partial \mathbf{Q}(\mathbf{p}, \mathcal{T})}{\partial \mathbf{t}_{\mathbf{v}_i}} &= \alpha_{\mathbf{v}_i}^{\mathbf{p}} \mathbf{I} \end{aligned}$$

Another building block of our optimization terms is the perspective projection $\pi_{\mathbf{c}}$ with intrinsic parameters $\mathbf{c} = (f_x, f_y, c_x, c_y)$:

$$\begin{aligned} \pi_{\mathbf{c}} : \mathbb{R}^3 &\rightarrow \mathbb{R}^2 \\ \pi_{\mathbf{c}} \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} \right) &= \begin{bmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{bmatrix} \end{aligned}$$

whose partial derivatives with respect to the point $\mathbf{p} = (x, y, z)^T$ are derived as:

$$\frac{\partial \pi_{\mathbf{c}}(\mathbf{p})}{\partial \mathbf{p}} = \begin{bmatrix} \frac{f_x}{z} & 0 & -\frac{f_x x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{f_y y}{z^2} \end{bmatrix}$$

By applying the chain rule, derivatives of all three optimization terms can be computed.

Derivative of 2D reprojection term. For a pixel $\mathbf{u} \in \Pi_s \subset \mathbb{R}^2$ and its corresponding 3D point \mathbf{p}_u , we derive partial derivatives of $r_{2D}^{\mathbf{u}}(\mathcal{T})$ as follows:

$$\frac{\partial r_{2D}^{\mathbf{u}}(\mathcal{T})}{\partial \boldsymbol{\epsilon}_{\mathbf{v}_i}} = -w_{\mathbf{u}} \alpha_{\mathbf{v}_i}^{\mathbf{p}_u} \begin{bmatrix} \frac{f_x}{\mathbf{p}_u^z} & 0 & -\frac{f_x \mathbf{p}_u^x}{(\mathbf{p}_u^z)^2} \\ 0 & \frac{f_y}{\mathbf{p}_u^z} & -\frac{f_y \mathbf{p}_u^y}{(\mathbf{p}_u^z)^2} \end{bmatrix} \overline{\mathbf{R}_{\mathbf{v}_i}(\mathbf{p}_u - \mathbf{v}_i)}$$

$$\frac{\partial r_{2D}^{\mathbf{u}}(\mathcal{T})}{\partial \mathbf{t}_{\mathbf{v}_i}} = w_{\mathbf{u}} \alpha_{\mathbf{v}_i}^{\mathbf{p}_u} \begin{bmatrix} \frac{f_x}{\mathbf{p}_u^z} & 0 & -\frac{f_x \mathbf{p}_u^x}{(\mathbf{p}_u^z)^2} \\ 0 & \frac{f_y}{\mathbf{p}_u^z} & -\frac{f_y \mathbf{p}_u^y}{(\mathbf{p}_u^z)^2} \end{bmatrix}$$

Derivative of depth term. When computing the partial derivatives of the depth term $r_{\text{depth}}^{\mathbf{u}}(\mathcal{T})$, we need to additionally apply the projection to the z -component in the chain rule:

$$\frac{\partial r_{\text{depth}}^{\mathbf{u}}(\mathcal{T})}{\partial \boldsymbol{\epsilon}_{\mathbf{v}_i}} = -w_{\mathbf{u}} \alpha_{\mathbf{v}_i}^{\mathbf{p}_u} [0 \ 0 \ 1] \overline{\mathbf{R}_{\mathbf{v}_i}(\mathbf{p}_u - \mathbf{v}_i)}$$

$$\frac{\partial r_{\text{depth}}^{\mathbf{u}}(\mathcal{T})}{\partial \mathbf{t}_{\mathbf{v}_i}} = w_{\mathbf{u}} \alpha_{\mathbf{v}_i}^{\mathbf{p}_u} [0 \ 0 \ 1]$$

Derivative of regularization term. For a graph edge $(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E}$, the partial derivatives of $r_{\text{reg}}^{\mathbf{v}_i, \mathbf{v}_j}(\mathcal{T})$ with respect to $\boldsymbol{\epsilon}_{\mathbf{v}_i}$, $\mathbf{t}_{\mathbf{v}_i}$, $\boldsymbol{\epsilon}_{\mathbf{v}_j}$, $\mathbf{t}_{\mathbf{v}_j}$ are computed as:

$$\frac{\partial r_{\text{reg}}^{\mathbf{v}_i, \mathbf{v}_j}(\mathcal{T})}{\partial \boldsymbol{\epsilon}_{\mathbf{v}_i}} = -\overline{\mathbf{R}_{\mathbf{v}_i}(\mathbf{v}_j - \mathbf{v}_i)} \quad \frac{\partial r_{\text{reg}}^{\mathbf{v}_i, \mathbf{v}_j}(\mathcal{T})}{\partial \boldsymbol{\epsilon}_{\mathbf{v}_j}} = \mathbf{0}$$

$$\frac{\partial r_{\text{reg}}^{\mathbf{v}_i, \mathbf{v}_j}(\mathcal{T})}{\partial \mathbf{t}_{\mathbf{v}_i}} = \mathbf{I} \quad \frac{\partial r_{\text{reg}}^{\mathbf{v}_i, \mathbf{v}_j}(\mathcal{T})}{\partial \mathbf{t}_{\mathbf{v}_j}} = -\mathbf{I}$$

D.2 Differentiable Linear Solve Operation

To simplify the notation, in the following we use $\mathbf{A} = \mathbf{J}_n^T \mathbf{J}_n$, $\mathbf{b} = -\mathbf{J}_n^T \mathbf{r}_n$ and $\mathbf{x} = \Delta \mathcal{T}$, which results in the linear system of the form:

$$\mathbf{A} \mathbf{x} = \mathbf{b}$$

For matrix $\mathbf{A} \in \mathbb{R}^{6N \times 6N}$ and vectors $\mathbf{b} \in \mathbb{R}^{6N}$ and $\mathbf{x} \in \mathbb{R}^{6N}$ we define the linear solve operation as:

$$\Lambda : \mathbb{R}^{6N \times 6N} \times \mathbb{R}^{6N} \rightarrow \mathbb{R}^{6N}, \quad (\mathbf{A}, \mathbf{b}) \mapsto \mathbf{A}^{-1} \mathbf{b} = \mathbf{x}$$

In order to compute the derivative of the linear solve operation, we follow the analytic derivative formulation of [23]. If we denote the partial derivative of the loss \mathcal{L} with respect to linear system solution \mathbf{x} as $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$, we can compute the partial derivatives with

Appendix

respect to matrix \mathbf{A} and vector \mathbf{b} as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \mathbf{A}^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{x}} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \left(\mathbf{A}^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{x}} \right) \mathbf{x}^T = -\frac{\partial \mathcal{L}}{\partial \mathbf{b}} \mathbf{x}^T$$

Thus, the computation of $\frac{\partial \mathcal{L}}{\partial \mathbf{b}}$ requires solving a linear system with matrix \mathbf{A} . To solve this system, we either run PCG solver again with different right-hand side, or if LU solver is used, we can re-use the LU decomposition from the forward pass.

Acronyms

1D, 2D, 3D, ...	n spatial dimentions.
Adam	Adaptive Moment Estimation.
AR	Augmented Reality.
ARAP	As-Rigid-As-Possible.
BCE	Binary Cross Entropy.
BN	Batch Normalization.
CE	Cross Entropy.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
ED	Embedded Deformation.
FCN	Fully Connected Network.
FN	False Negative.
FP	False Positive.
GN	Gauss-Newton.
GPU	Graphical Processing Unit.
GRU	Gated Recurrent Unit.
ICP	Iterative Closest Points.
IMU	Inertial Measurement Unit.
LiDAR	Light Detection and Ranging.
LSTM	Long Short-Term Memory.
MAE	Mean Absolute Error.
MLP	Multi-Layer Perception.
MSE	Mean Squared Error.
MVS	Multi-View Stereo.
PCA	Principal Component Analysis.

Appendix

PCG	Preconditioned Conjugate Gradient.
ReLU	Rectified Linear Unit.
RNN	Recurrent Neural Network.
SDF	Signed Distance Field.
SfM	Structure from Motion.
SGD	Stochastic Gradient Descent.
SIFT	Scale-Invariant Feature Transform.
TN	True Negative.
TP	True Positive.
TSDF	Truncated Signed Distance Field.
VR	Virtual Reality.

List of Tables

3.1	Quantitative matching comparison. We outperform baseline matching methods by a considerable margin. 2D/3D errors are average pixel/-point errors, and 2D/3D accuracy is the percentage of pixels/points with distance of at most 20 px/0.05 m.	45
3.2	Quantitative reconstruction comparison. Our learned correspondences significantly improve both tracking and reconstruction quality compared to the state-of-the-art approaches. We also provide ablation studies on training data type and different network parts.	47
4.1	Quantitative non-rigid tracking evaluation. We evaluate non-rigid tracking on the DeepDeform dataset [13], showing the benefit of end-to-end differentiable optimizer losses and self-supervised correspondence weighting. We denote correspondence prediction as Φ_c , Φ_{c+g} and Φ_{c+g+w} , depending on which losses $\mathcal{L}_{\text{corr}}$, $\mathcal{L}_{\text{graph}}$, $\mathcal{L}_{\text{warp}}$ are used, and correspondence weighting as $\Psi_{\text{supervised}}$ and $\Psi_{\text{self-supervised}}$, either using an additional supervised loss or not.	66
4.2	Quantitative non-rigid reconstruction evaluation. Our method achieves state-of-the-art non-rigid reconstruction results on the DeepDeform benchmark [13]. Both our end-to-end differentiable optimizer and the self-supervised correspondence weighting are necessary for optimal performance. Not only does our approach achieve lower deformation and geometry error compared to state of the art, our correspondence prediction is about $85\times$ faster.	68
5.1	Quantitative comparison. We show quantitative comparisons with state-of-the-art approaches, evaluating geometry using chamfer distance ($\times 10^{-4}$), and deformation using EPE3D ($\times 10^{-2}$). We also include an ablation study of different components of our method: GRAPH = coverage and interior losses, AO = affinity optimization with affinity consistency and sparsity, VC = viewpoint consistency, SC = surface consistency. . . .	84
6.1	Quantitative comparison. We compare with state-of-the-art methods and perform ablations on test set of Scannet dataset [17].	101
6.2	Feature fusion runtime. We analyse runtime of the per-frame feature fusion.	106
6.3	Mesh extraction runtime. We analyse runtime of the per-chunk mesh extraction.	106

List of Figures

1.1	Applications of non-rigid reconstruction. Possible use cases range from telepresence in VR/AR (<i>left</i> , from [3]) to safe human-robot interactions (<i>right</i>).	3
1.2	Non-rigid reconstruction pipeline. A seminal work of DynamicFusion [4] incrementally reconstructs a canonical shape and at the same time tracks deformations over input RGB-D frames.	5
2.1	Shape representations. We can model the surface of a 3D object as a mesh (<i>left</i>), a point cloud (<i>middle</i>) or a signed distance field (<i>right</i>).	12
2.2	Deformation graph construction. Given a shape geometry, we define the deformation graph nodes and edges either by using uniform sampling and connecting nearest-neighbors (<i>left</i>), or by optimizing over a spatially-varying node positions, influence and connectivity, optimal for the entire video sequence (<i>right</i>).	20
2.3	Iterative closest points. The template mesh (<i>red</i>) is iteratively aligned to the current point cloud (<i>blue</i>) by finding (new) closest point matches at each iteration.	22
2.4	Global correspondences. We can either estimate sparse correspondences for selected pixels via heatmap prediction (<i>left</i>), or densely predict correspondences for all pixels using an optical flow network (<i>right</i>).	24
2.5	Incremental non-rigid reconstruction. At each frame we extract the latest mesh from the grid-based SDF representation, compute correspondences with RGB-D frame and optimize the deformations, and finally integrate the newly observed RGB-D data to complete the SDF representation (figure is from [35]).	25
2.6	Volumetric fusion. We project a dense grid of voxels to the depth map, and compute projective depth distances, positive in front of the surface and negative behind it (figure is from [36]).	26
2.7	MLP-based reconstructions. Some examples of SIREN [39] reconstructions, where an MLP stores the SDF value for any continuously queried point.	28
2.8	Multi-MLP object representation. Each MLP stores a corresponding transformation, which maps from local MLP-specific space to the global world space. A point is transformed to every local space, evaluated by a small MLP, and the MLP contributions are aggregated via a (weighted) sum.	29

2.9	Feature-based MLP reconstruction. Shape is reconstructed using an (object-specific) feature grid and a (shared) MLP decoder. Additional networks such as a 3D CNN are often employed to improve generalization.	30
2.10	Overview of 3D-R2N2 [49]. A multi-view reconstruction approach that encodes RGB frames into global codes that are fused across frames via an LSTM network, without considering input camera poses.	30
2.11	Overview of NeuralRecon [55]. A 3D reconstruction approach that encodes multi-view RGB frames into coarse-to-fine feature grids, aligned using camera poses, and fused across time using GRU units.	31
3.1	Approach overview. We propose a semi-supervised strategy combining self-supervision with sparse annotations to build a large-scale RGB-D dataset of non-rigidly deforming scenes (400 scenes, 390,000 frames, 5,533 densely aligned frame pairs). With this data, we propose a new method for non-rigid matching, which we integrate into a non-rigid reconstruction approach.	36
3.2	Example frames of the collected dataset. Our large-scale dataset contains a large variety of dynamic sequences with segmentation masks and point correspondences between different RGB-D frames.	37
3.3	Network architecture. We devise an end-to-end architecture for RGB-D matching based on a Siamese network to find matches between a source and a target frame. Our network is based on two towers that share the encoder and have a decoder that predicts two probability heatmaps in the target frame that encode the likelihood of the location of the source point. Our network also predicts a depth value for the matched point and a visibility score that measures if the source point is visible in target frame.	40
3.4	Dense alignment constraints. We qualitatively show the effect of different optimization constraints on the dense alignment of the given RGB-D frame pair.	44
3.5	Qualitative heatmap prediction results. Our matching approach works well even for highly non-rigid motions.	46
3.6	Qualitative comparison to DynamicFusion [4]. We use our re-implementation of the framework.	48
3.7	Qualitative comparison to MonoFVV [64]. Reconstruction results were kindly provided by the authors.	49
3.8	Qualitative comparison to KillingFusion [62]. Reconstruction results were kindly provided by the authors.	50
3.9	Reconstruction visualizations. Our approach obtains high-quality warped model (in the current frame) and canonical shape (in the initial frame).	51
3.10	Visibility detection. The visibility score is in the range $[0, 1]$, it is high for visible correspondences and low for occluded parts. We filter out all correspondences with visibility score less than 0.50.	52

3.11	Ablation on data generation. Comparison of correspondence prediction for reference frame (left) using self-supervised training data (middle) and semi-supervised dense data (right).	52
3.12	Limitations. Integration of background can cause wrong deformation graph connectivity, which can lead to non-rigid tracking failure.	54
4.1	Neural Non-Rigid Tracking. Based on RGB-D input data of a source and a target frame, our learned non-rigid tracker estimates the non-rigid deformations to align the source to the target frame. We propose an end-to-end approach, enabling correspondences and their importance weights to be informed by the non-rigid solver. Similar to robust optimization, this provides robust tracking, and the resulting deformation field can then be used to integrate the depth observations in a canonical volumetric 3D grid that implicitly represents the surface of the object (final reconstruction).	56
4.2	Deformation graph construction. Given an object in the source RGB-D frame, we define a deformation graph \mathcal{G} over the former. Nodes \mathcal{V} (red spheres) are uniformly subsampled over the source RGB-D frame. Edges \mathcal{E} (green lines) are computed between nodes based on geodesic connectivity among the latter.	58
4.3	Overview of neural non-rigid tracker. Given a pair of source and target images, \mathcal{I}_s and \mathcal{I}_t , a dense correspondence map \mathcal{C} between the frames is estimated via a convolutional neural network Φ . Importance weights \mathcal{W} for these correspondences are computed through a function Ψ . Together with a graph \mathcal{G} defined over the source RGB-D frame \mathcal{P}_s , both \mathcal{C} and \mathcal{W} are input to a differentiable solver Ω . The solver outputs the graph motion \mathcal{T} , i.e., the non-rigid alignment between source and target frames. Our approach is optimized end-to-end, with losses on the final alignment using $\mathcal{L}_{\text{graph}}$ and $\mathcal{L}_{\text{warp}}$, and an intermediate loss on the correspondence map $\mathcal{L}_{\text{corr}}$	60
4.4	Appearance reconstruction. We compute shape textures by aggregating color images over 100 frames of motion into a voxel grid.	64
4.5	Qualitative comparison with DynamicFusion [4] and DeepDeform [13]. We compare reconstruction results on test sequences from [13]. The rows show different time steps of the sequence.	67
4.6	Qualitative comparison with MonoFVV [64]. The results on test sequences from [13] were kindly provided by the authors.	70
4.7	Qualitative comparison with KillingFusion [62]. The results on test sequences from [13] were kindly provided by the authors.	71
4.8	Generalization to a different sensor. We show qualitative reconstruction results on VolumeDeform [35] sequences, captured by a Microsoft Kinect sensor.	72

- 5.1 **Neural Deformation Graphs.** Given range input data, represented as a signed distance field, our method predicts globally-consistent deformation graph that is used to reconstruct the non-rigidly deforming surface of an object. The surface of the object is represented as a set of implicit functions centered around the deformation graph nodes. Our global optimization provides consistent surface and deformation prediction, enabling robust tracking of an observed input sequence and even multiple disjoint captures of the same object (as we do not assume sequential input data). 74
- 5.2 **Method overview.** A Neural Deformation Graph encodes a 64^3 SDF grid to a graph embedding with graph node positions \mathbf{V} , rotations \mathbf{R} and importance weights \mathbf{W} . To compute an SDF value for a sample point $(X, Y, Z) \in \mathbb{R}^3$, the point is transformed to local coordinates around each node, and passed through locally embedded implicit functions that are represented as MLPs; the global SDF value is computed by interpolating the local MLP predictions using the node radii \mathbf{r} and importance weights \mathbf{W} . For graph regularization, a set of affinity matrices $\mathbf{A}_i \in \mathbb{R}^{N \times N}$ and a node-to-node distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ are globally optimized. 77
- 5.3 **Point sample visualization.** We visualize the uniform samples P_{un} (*green*), near-surface samples P_{ns} (*red*) and the surface samples P_s (*blue*) for a slice of samples with $z \in [-0.01, 0.01]$ at one frame of the character sequence shown in the Fig. 5.1. 82
- 5.4 **Qualitative comparison on synthetic sequences.** We qualitatively compare our method to the baseline methods on synthetic data. Each point is given a color value w.r.t. its location in the bounding box of the first frame. With perfect tracking and reconstruction, a specific point on the surface will have the same color throughout the entire sequence, while errors in tracking result in changing surface colors. Our approach outperforms state of the art in both reconstruction and deformation tracking quality. 83
- 5.5 **Qualitative comparison on real sequences.** We show qualitative comparisons of our method with multi-view DynamicFusion [65] on real sequences captured by four Kinect Azure sensors. Colors represent corresponding locations in the first frame of the sequence to visualize the tracking quality and consistency. 85
- 5.6 **Capture setup.** We capture non-rigid motion using 4 Kinect Azure sensors, which are pre-calibrated and hardware-synced to guarantee spatial and temporal coherence of depth captures. We capture depth images at resolution of 1024×1024 , using the highest available frame-rate of 15 FPS. 87
- 5.7 **Influence of pose conditioning.** We compare our implicit reconstruction with pose conditioning (*middle*) to without pose conditioning (*left*). Pose conditioning clearly improves reconstruction performance in regions of very strong deformation. 88

5.8	Neural deformation graph examples. We visualize a few examples of neural deformation graphs, optimized on real sequences (<i>top</i>) and synthetic data (<i>bottom</i>) using self-supervision.	88
6.1	TransformerFusion. We present an online scene reconstruction method that takes a monocular RGB video as input. The features extracted from each observed image are fused incrementally with a transformer architecture. This fusion approach learns to attend to the most relevant image frames for each 3D location (see view attention color maps of the most relevant frame) achieving state-of-the-art reconstruction results.	92
6.2	Method overview. Given multiple input images, we compute coarse and fine level features. Using a transformer architecture, we separately fuse these coarse and fine features in a voxel grid. To improve the spatial features, we use a refinement network for both the coarse and the fine features. From these feature grids, we extract an occupancy field using a lightweight MLP.	95
6.3	View attention visualization. Visualization of selected camera views with self-supervised attention weights (lower weights are visualized as <i>blue</i> and higher as <i>red</i>) for specific 3D locations (highlighted as <i>green</i>) in the scene.	98
6.4	Qualitative comparison to state of the art. We compare scene reconstructions on test set of ScanNet dataset [17]; note that only RGB input is used by each method while the ground truth is reconstructed using the input depth.	102
6.5	Qualitative comparison of ablations of our approach. We perform ablations on test set of ScanNet dataset [17]; note that only RGB input is used by each method while the ground truth is reconstructed using the input depth.	103
6.6	Qualitative comparison of ablations on feature voxel size. We replaced the original voxel size of 10 cm at the fine grid level with 30 cm and 15 cm.	105
6.7	Qualitative reconstruction results. We visualize reconstructions of representative scenes from the test-set of the ScanNet dataset [17].	107
6.8	Limitations. Our approach can lack detail at partially observed and occluded objects, and result in inaccurate reconstruction of transparent surfaces, such as glass windows.	108
C.1	Copyright authorization. Authorization for reuse of paper <i>DeepDeform: Learning Non-rigid RGB-D Reconstruction With Semi-Supervised Data</i> in this dissertation.	152
C.2	Copyright authorization. Authorization for reuse of paper <i>Neural Non-rigid Tracking</i> in this dissertation.	165

Appendix

C.3	Copyright authorization. Authorization for reuse of paper <i>Neural Deformation Graphs for Globally-consistent Non-rigid Reconstruction</i> in this dissertation.	177
C.4	Copyright authorization. Authorization for reuse of paper <i>TransformerFusion: Monocular RGB Scene Reconstruction using Transformers</i> in this dissertation.	191