

DISS. ETH NO. 29876

On the Reconstruction, Understanding and Editing of 3D Scenes for Augmented Reality

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES from ETH Zürich

(Dr. sc. ETH Zürich)

presented by

Silvan Adrian Weder

MSc in Robotics, Systems, and Control
ETH Zürich, Schweiz

born on 19th of October, 1993

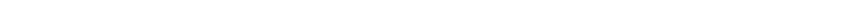
accepted on the recommendation of

Prof. Dr. Marc Pollefeys
Prof. Dr. Martin R. Oswald
Prof. Dr. Bastian Leibe

2023

Abstract

We stand on the cusp of a new technological era, where technology seamlessly integrates into our daily lives. Entering this brave new world requires the convergence of always-on artificial intelligence and augmented reality. However, we still must overcome numerous challenges to realize this vision. This thesis addresses three pivotal challenges that still remain: 3D reconstruction, 3D scene understanding, and 3D scene editing. Augmented reality applications demand a reconstruction of the world that is continuously updated with new information. Therefore, we start with tackling the challenge of incrementally fusing noisy and outlier contaminated data in an online system. We approach the challenge from a data-driven perspective, utilizing a learned scene representation, to enhance existing methods' efficiency with the power of machine learning. However, spatial awareness alone is not sufficient. Hence, we move on to 3D scene understanding, where we confront the high costs of annotating datasets for 3D semantic segmentation models. We introduce an automated semantic annotation pipeline that matches human annotation quality, unifying the predictions of state-of-the-art models into a shared label space that are further improved through 3D lifting. Additionally, we extend the online reconstruction pipeline to semantic mapping, overcoming limited receptive fields with a spatio-temporal attention mechanism that efficiently combines information from 2D and 3D with past information. In the final part, we explore the use of neural radiance fields for 3D scene editing. Thus, we propose a method that leverages priors encoded in powerful 2D inpainting method for removing objects from scenes. This requires the design of a confidence-based view-selection mechanism during the optimization stage that enforces multi-view consistency in the final reconstruction.



Zusammenfassung

In dieser Arbeit wurden drei wesentliche Säulen der erweiterten Realität beleuchtet: die Rekonstruktion, die Perzeption und die Bearbeitung von 3D Szenen. Die Rekonstruktion bildet das Fundament vieler AR-Anwendungen und liefert die notwendige räumliche Wahrnehmung, die für immersive Nutzererlebnisse erforderlich ist. In Bezug auf die 3D Rekonstruktion konzentrierten wir uns auf die Herausforderung, Szenen in Echtzeit mithilfe eines kontinuierlichen Stroms von Sensordaten zu aktualisieren. Wir haben gezeigt, wie maschinelles Lernen zur Verbesserung der Genauigkeit der rekonstruierten Geometrie eingesetzt werden kann. Dabei haben wir insbesondere die Bewältigung von Ausreißern, welche die rekonstruierte Geometrie beeinträchtigen, behandelt. Dazu wurde der Fusionsprozess in eine gelernte Darstellung verlagert und dieser latenten Raum anschließend in die finale Geometrie übersetzt, wodurch Ausreißer effizient entfernt wurden. Im Bereich der 3D Perzeption sind wir zwei zentrale Herausforderungen angegangen. Erstens haben wir ein automatisiertes Verfahren für die Annotation von Daten entwickelt, um den Aufwand für die Erstellung von 3D Segmentierungsdatensätzen zu reduzieren. Zweitens haben wir die inkrementelle semantische Modellierung mit ausschliesslich lokalen Informationen verbessert, indem wir ein neuronales Expertennetzwerk präsentierten, das Informationen aus dem 2D- und 3D-Raum kombiniert. Schließlich haben wir im Bereich der 3D Bearbeitung eine Methode zur Entfernung von Objekten aus neuronalen Strahlenfeldern vorgestellt. Diese Methode verwendet Signale aus vortrainierten 2D Modellen, um Objekte nahtlos aus den Szenen zu entfernen. Um Inkonsistenzen zwischen den kompletten Bildern aufzulösen, haben wir ein Optimierungsverfahren entwickelt und so die Qualität der Entfernung von Objekten verbessert.



Acknowledgements

First, I wish to express my gratitude to Marc for affording me the invaluable opportunity to embark on my PhD journey within the Computer Vision and Geometry group at ETH Zürich. His unwavering support and trust provided me with the essential resources and the freedom to follow my curiosity, enabling me to explore diverse avenues within the dynamic field of computer vision.

I extend my sincere thanks to Martin, who played an instrumental role in shaping my academic journey. It was during my master's studies that Martin introduced me to the captivating field of 3D computer vision, igniting a passion that has guided my path for the last four years. His guidance and steadfast presence have been a source of inspiration and motivation throughout this significant chapter of my life.

I extend my special thanks to Akihito, a remarkable mentor, despite the vast distances that separated us. Over the course of my studies, his guidance proved to be an invaluable compass on my academic journey. Our journey reached its pinnacle with an unforgettable visit to Japan, a memory I will forever cherish.

Thank you Johannes, a long-term collaborator since the inception of our depth fusion project to the culmination of my PhD thesis. His expertise in paper writing has consistently elevated our work, ensuring clarity and impact in our research publications.

Thank you Francis, thank you Hermann. Working alongside Francis and Hermann, the final project of my PhD was a truly enriching experience.

I had the distinct pleasure of embarking on two exciting internships during my PhD journey, first at Meta and then at Niantic. My time at Meta introduced me to a remarkable group of individuals, spanning locations in Redmond, Zurich,

and Sausalito. Although the internship was necessitated to be remote, I was fortunate to have Shuo Chen as an exceptional manager. I also had the privilege of collaborating with a stellar team that included Audrey, Alex, Christoph, Fabian, and Katrin. Their expertise and camaraderie enriched my internship experience at Meta.

My internship at Niantic stands as one of the defining highlights of my PhD journey. Gabe, in particular, emerged as an exemplary role model in the realms of leadership and dedication. His unwavering commitment to my growth and development was manifest in our remarkable weekly meetings, where he consistently pushed the boundaries of my thinking and approaches. Furthermore, I had the privilege of collaborating with extraordinary individuals like Michael, Aron, Guillermo, and Sara, who redefined the essence of international collaboration. Their support and collective efforts were instrumental in shaping my Niantic experience. Along this remarkable journey, I crossed paths with several remarkable individuals, including Jamie Wynn, Jamie Watson, Mohammed, and Zadar, all of whom generously offered their guidance and help, enriching my experience and expanding my horizons. I am deeply grateful to each of them for their contributions to my academic and professional growth.

A PhD journey is not solely defined by research; it is equally about the people who surround and support you along the way. Over the years, I have had the privilege of meeting remarkable colleagues, some of whom have become cherished friends. I extend my heartfelt thanks to Mihai, with whom I have spent discussing everything from technology and research to politics and societal matters. His insights and camaraderie have been invaluable. To Paul-Edouard and Rémi, who began their journeys alongside me, I owe countless adventures and spirited debates. From remote work in a house in Costa Rica to shared AirBnBs in Vancouver during CVPR, we have shared remarkable experiences. To Songyou, with whom I had many thought-provoking discussions about career paths and aspirations. To Iago, a dear Spanish friend who exemplifies openness and kindness, he visited our lab as an academic guest but he also persuaded me to explore the breathtaking beauty of Galicia during vacation, and invited me to partake in the celebration of his extraordinary wedding in Santiago. To Victor and Rebecka, who truly

remarkable individuals redefining the social aspects of a PhD, and I always relish the opportunities to rendezvous with them, whether it is for a lively party or a delightful dinner accompanied by drinks, somewhere around the world. To Philipp, who added an extra spark to the Swiss-Austrian rivalry in skiing, making it all the more memorable. To Shaohui, a person with a great sense of humor, and our conversations are consistently intriguing and enjoyable. With Petr, I discovered a fellow cycling fan, and our discussions about recent races were always enjoyable. Taein's candid discussions about relationships and anything else during train rides to ski retreats added depth and camaraderie to our journeys. To Julia, a fellow "All in" podcast enthusiast, with whom I had enlightening discussions during our time in Sicily, I appreciate our shared interests and insightful conversations. Thanks to Linfei, Luca, and Sandro (and of course Rémi), who shared the same office with me during my PhD. While our office was very focused most of the time, even more special were the humorous and entertaining interruptions that added laughter and camaraderie to our daily routine. Their presence brought both productivity and joy to our shared workspace, and I am thankful for the memorable moments we created together.

And of course all other current members of our group. To Boyang, Fangjinhua, Zuria, Lubor, Katarina, Marcel, Daniel T., Daniel B., Iro, Ian, Jonas, Zador, Marko with whom I had the pleasure to share some of my time. And of course all the fantastic people that I met in the field at conferences and summer schools. These friendships have illuminated my PhD years and added a layer of richness to the journey that I will forever cherish. It was a truly amazing ride.

Last but certainly not least, my heartfelt thanks go to my parents Gerda and Adrian. Their unwavering support and encouragement have been the bedrock upon which my journey has been built. It is their guidance and the values instilled in me, including curiosity, a relentless pursuit of knowledge, and a steadfast ambition, that have propelled me to where I am today. Their love and belief in me have been instrumental, and I am profoundly grateful for their role in shaping my path.

In closing, my heart goes to my partner, Carmen. Throughout the arduous journey of pursuing this PhD, I can confidently say that I could not have asked for a more supportive and inspiring partner to stand by my side. As anyone who

has embarked on a doctoral journey knows, it is a path paved with exhilarating highs and daunting lows. Carmen has been the unwavering pillar of strength and motivation that has propelled me forward, consistently pushing me beyond my comfort zone and wholeheartedly championing my academic pursuits. Whether it was the immersive weeks spent in Costa Rica or the relentless demands of impending deadlines, Carmen's steadfast support and unwavering understanding have been the cornerstone of my resilience. Even during our cherished vacation days, when paper submissions tugged at my attention, she never once uttered a word of complaint. Her enduring encouragement and unwavering belief in my journey have been invaluable beyond measure. I am profoundly grateful for her constant presence, which has illuminated every page of this remarkable chapter in my life.

Silvan Adrian Weder
Zürich, October 2023

Contents

Abstract	i
Zusammenfassung	iii
Acknowledgements	v
List of Figures	xv
List of Tables	xix
I Preamble	1
1 Introduction	3
1.1 Scope of this work	9
1.1.1 Contributions	11
1.2 Outline	12
II 3D Reconstruction	13
2 Introduction	15
3 Background	19
3.1 From Measurements to 3D Model	19
3.1.1 Obtaining the Measurements	20

3.1.2	Aggregating the Measurements	22
3.2	Scene Representations for 3D Reconstruction	23
3.2.1	Non-learned Representations	23
3.2.2	Learned Representations	25
3.3	Global Depth Fusion	28
3.4	Online Depth Fusion	29
3.4.1	Surfel-based Fusion Methods.	30
3.4.2	Probabilistic Depth Map Fusion.	30
3.4.3	Learned Depth Map Fusion	31
3.4.4	Learned RGB fusion	31
3.5	Appearance Reconstruction	32
3.5.1	Classical Texture Mapping.	32
3.5.2	Online Appearance Aggregation.	33
4	Learning-based Depth Map Fusion	35
4.1	Method	39
4.1.1	Review of Standard TSDF Fusion	39
4.1.2	Pipeline Overview	40
4.1.3	Depth Routing	40
4.1.4	TSDF Extraction	42
4.1.5	Depth Fusion	43
4.1.6	TSDF Update Integration	44
4.1.7	Outlier Filtering	44
4.1.8	Loss Function and Training Procedure	44
4.2	Experiments	45
4.2.1	Implementation Details	45
4.2.2	Results	46
4.2.3	Synthetic Data	47
4.2.4	Real-World Data	50
4.2.5	Ablation Studies	55
4.3	Discussion	55
4.3.1	Limitations	55
4.3.2	Summary	57

5	Moving the fusion to a latent space	59
5.1	Method	61
5.1.1	Overview	61
5.1.2	Feature Extraction	63
5.1.3	Feature Fusion	63
5.1.4	Feature Integration	64
5.1.5	Feature Translation	64
5.1.6	Training Procedure and Loss Function.	65
5.2	Experiments	66
5.2.1	Implementation Details.	66
5.2.2	Evaluation Metrics.	66
5.2.3	Results on Synthetic Data	67
5.2.4	Ablation Study	69
5.2.5	Loss Ablation.	70
5.2.6	Real-World Data	71
5.3	Discussion	72
5.3.1	Limitations	72
5.3.2	Summary	73
6	Learning-based Appearance Fusion	83
6.1	DeepSurfels 3D Scene Representation	87
6.1.1	Data Structure	87
6.1.2	Surface Fitting	89
6.2	Online Appearance Fusion Pipeline	89
6.2.1	Differentiable Projection Π	91
6.2.2	Fusion Network	91
6.2.3	Inverse Projection Π^{-1}	92
6.2.4	Appearance Rendering Module	92
6.2.5	Loss and Optimization	93
6.3	Evaluation	93
6.3.1	Datasets	95
6.3.2	Metrics	95
6.3.3	Novel View Synthesis	95
6.3.4	Generalization	96
6.3.5	Ablation Studies	96
6.3.6	Real-world data	97

6.3.7	Runtime	97
6.4	Discussion	98
6.4.1	Limitations	98
6.4.2	Summary	98
III	3D Scene Understanding	105
7	Introduction	107
8	Background	111
8.1	Overview about Scene Understanding	111
8.1.1	Recognition	111
8.1.2	Image Segmentation	112
8.1.3	Semantic Segmentation	113
8.1.4	3D Semantic Segmentation	114
8.2	Datasets for Scene Understanding	115
8.2.1	2D Datasets	115
8.2.2	3D Datasets	116
8.2.3	Annotating Datasets for Scene Understanding	117
8.3	3D Semantic Segmentation	117
8.3.1	Offline vs. Online Processing.	118
8.3.2	Offline 3D Semantic Segmentation	119
8.3.3	Online 3D Semantic Segmentation.	120
9	Automatic Annotation for 3D Semantic Segmentation	123
9.1	Method	125
9.1.1	Base Models	125
9.1.2	Translation between Label Spaces	127
9.1.3	Model Consensus	129
9.1.4	3D Lifting	131
9.1.5	Relabeling ScanNet Scenes	132
9.2	Experiments	133
9.2.1	Implementation Details	133
9.2.2	Datasets	133
9.2.3	Baselines	135
9.2.4	Comparison to State-of-the-Art	135

9.2.5	Ablation Studies	140
9.2.6	Experiments on ARKitScenes	141
9.3	Discussion	141
9.3.1	Limitations	141
9.3.2	Summary	142
10	Online Semantic 3D Reconstruction	143
10.1	Method	146
10.1.1	Overview	146
10.1.2	Scene Representation	146
10.1.3	2D Encoder	147
10.1.4	3D Encoder	149
10.1.5	Spatio-Temporal Expert	149
10.1.6	Loss Function and Training Details	151
10.2	Experiments	154
10.2.1	Implementation Details	154
10.2.2	Online Methods in Comparison	154
10.2.3	Datasets and Metrics	156
10.2.4	3D Semantic Segmentation	157
10.2.5	Ablation Studies	161
10.3	Discussion	164
10.3.1	Discussion of Baseline Comparison	164
10.3.2	Why is there no qualitative comparison to baselines?	164
10.3.3	Limitations	165
10.3.4	Summary	166
IV	3D Scene Editing	167
11	Introduction	169
12	Background	173
12.1	Overview	173
12.1.1	Image Inpainting	173
12.1.2	Video Inpainting	174
12.1.3	3D Editing	176
12.2	Novel View Synthesis	177

12.2.1	Neural Radiance Fields	177
12.2.2	Generative Models for Novel View Synthesis	178
12.2.3	Inpainting in Novel View Synthesis	179
13	Removing Objects from Scenes	181
13.1	Method	184
13.1.1	RGB and Depth Inpainting Network	186
13.1.2	Background on NeRFs	186
13.1.3	Confidence-based View Selection	188
13.1.4	Implementation Details	191
13.2	Experiments	193
13.2.1	Datasets	193
13.2.2	Metrics	195
13.2.3	Ablations and Comparison with Baselines	196
13.3	Discussion	198
13.3.1	Limitations	198
13.3.2	Summary	199
V	Conclusion	203
14	Summary	205
15	Future Work	209
15.1	3D Scene Reconstruction	209
15.2	3D Scene Understanding	210
15.3	3D Scene Editing	211
	Appendices	213
A	Additional Results	215
A.1	Learning-based Depth Map Fusion	215
A.2	Moving the Fusion to a Learned Space	224
A.3	Learning-based Appearance Fusion	227
B	Dataset for Object Removal	231
	Bibliography	239

List of Figures

4.1	Standard TSDF fusion vs. our learned depth map fusion approach (on Kinect data [Shotton et al., 2013].	36
4.2	System overview for integrating depth maps into a global TSDF volume.	38
4.3	Proposed network architecture.	41
4.4	Qualitative Results on ShapeNet [Chang et al., 2015].	48
4.5	Evaluation of different noise levels σ	50
4.6	Qualitative results of our method on the Roadsign dataset [Ummenhofer and Brox, 2013].	51
4.7	Qualitative comparison on the heads scene of RGB-D Dataset 7-Scenes [Shotton et al., 2013].	53
4.8	Qualitative comparison on the Burghers of Calais scene [Zhou and Koltun, 2013].	54
4.9	Intersection over Union on Modelnet [Zhirong Wu et al., 2015] test data for different numbers of samples S	56
5.1	Results of our end-to-end depth fusion on real-world MVS data.	60
5.2	NeuralFusion pipeline overview.	62
5.3	Feature fusion network.	74
5.4	Translator network.	74
5.5	Quantitative and qualitative results on ShapeNet [Chang et al., 2015].	75
5.6	Mesh Accuracy (M.A.) visualization on ShapeNet meshes.	77

5.7	Reconstruction from noisy depth maps.	77
5.8	Reconstruction from outlier-contaminated data.	78
5.9	Performance of iterative fusion over time.	79
5.10	Random frame order permutations.	79
5.11	Visualization of our learned latent space encoding.	79
5.12	Ablation of loss terms.	80
5.13	Depth map fusion results on Scene3D [Zhou and Koltun, 2013].	81
5.14	Results on Tanks and Temples [Knapitsch et al., 2017].	82
6.1	Overview of our online appearance fusion pipeline and the Deep-Surfel scene representation.	84
6.2	DeepSurfel surface fitting.	88
6.3	Overview of our learned appearance fusion pipeline.	90
6.4	Qualitative and quantitative comparison on novel view synthesis with DeepSurfels.	94
6.5	Novel view synthesis for Replica [Straub et al., 2019] indoor scenes.	100
6.6	Qualitative results of our model on unseen scenes from ShapeNet [Chang et al., 2015].	101
6.7	Comparison of SRNs [Sitzmann et al., 2019b] and DeepVoxels [Sitzmann et al., 2019a].	102
6.8	Novel-view synthesis on unseen real-world data [Maier et al., 2017b].	104
9.1	Label space translation.	127
9.2	Resolving the translation from low-resolution to high-resolution label space.	128
9.3	LabelMaker pipeline overview.	130
9.4	Qualitative results on ScanNetv2 [Dai et al., 2017a] in 2D.	136
9.5	Dense 3D labels for ScanNetv2 [Dai et al., 2017a].	137
9.6	Automatic dense labeling of ARKitScenes.	139
10.1	Online semantic 3D reconstruction pipeline.	144
10.2	ALSTER pipeline overview.	148
10.3	Temporal expert network.	151
10.4	Temporal expert attention maps	152

10.5	Qualitative results of our proposed method and comparison between the different stages.	156
10.6	Additional qualitative results on the ScanNetv2 validation set — Part 1 of 2.	160
10.7	Additional qualitative results on the ScanNetv2 validation set — Part 2 of 2.	161
10.8	Our model on limited data.	163
10.9	Runtime analysis for different 2D backbones.	164
10.10	Confusion matrix of ScanNet validation set results.	166
13.1	Removal of unsightly objects.	182
13.2	Per-frame inpainting.	184
13.3	An overview of our method.	185
13.4	Full network architecture.	190
13.5	Mask refinement.	192
13.6	Results on ARKitScenes [Baruch et al., 2021a].	195
13.7	Qualitative comparisons with baseline.	201
13.8	Failure cases and limitations.	202
A.1	More qualitative results of standard TSDF and RoutedFusion on scene 3D data — Part 1 of 2.	216
A.2	More qualitative results of standard TSDF and RoutedFusion on scene 3D data — Part 2 of 2..	217
A.3	More qualitative results on ShapeNet test data — Part 1 of 6.	218
A.4	More qualitative results on ShapeNet test data — Part 2 of 6.	219
A.5	More qualitative results on ShapeNet test data — Part 3 of 6.	220
A.6	More qualitative results on ShapeNet test data — Part 4 of 6.	221
A.7	More qualitative results on ShapeNet test data — Part 5 of 6.	222
A.8	More qualitative results on ShapeNet test data — Part 5 of 6.	223
A.9	More qualitative results for different outlier fractions on ModelNet [Wu et al., 2015] examples.	225
A.10	Additional results on Scene3D [Zhou and Koltun, 2013].	226
A.11	Qualitative results of our model on unseen ShapeNet [Chang et al., 2015] car scenes for different DeepSurfel parameters — Part 1 of 2.	228

A.12	Qualitative results of our model on unseen ShapeNet [Chang et al., 2015] car scenes for different DeepSurfel parameters – Part 2 of 2.	229
B.1	Our real objects dataset – Part 1 of 3.	232
B.2	Our real objects dataset – Part 2 of 3.	233
B.3	Our real objects dataset – Part 3 of 3.	234
B.4	Our synthetic objects dataset – Part 1 of 3.	235
B.5	Our synthetic objects dataset – Part 2 of 3.	236
B.6	Our synthetic objects dataset – Part 3 of 3.	237

List of Tables

4.1	Quantitative results on ShapeNet [Chang et al., 2015].	49
4.2	Quantitative evaluation of our method on 3D Scene Data [Zhou and Koltun, 2013].	52
5.1	Ablation study on feature dimensions.	69
5.2	Our model trained on limited training data.	70
5.3	Quantitative evaluation on Scene3D [Zhou and Koltun, 2013].	71
6.1	Ablation study on ShapeNet [Chang et al., 2015] cars.	103
6.2	Varying number of feature channels.	103
9.1	Comparison of the label quality of the ScanNet labels.	134
9.2	Ablation of all base models in LabelMaker on our 5 labelled ScanNet [Dai et al., 2017a] scenes and Replica [Straub et al., 2019].	138
10.1	3D Semantic Segmentation on ScanNet [Dai et al., 2017a] test set	155
10.2	Ablating different aspects of our pipeline on ScanNet [Dai et al., 2017a] validation set.	158
10.3	3D Semantic Segmentation on ScanNet and SceneNN.	159
13.1	Comparison with baselines and state of the art methods.	194
13.2	Ablation on view selection methods.	198



Part I

Preamble

Introduction

In our daily lives, the act of interacting with the world is so ingrained that we often take it for granted. We seamlessly move through our morning routines like brewing a cup of coffee, navigate crowded train stations during our daily commute, complete household chores such as dish washing after dinner, and cap off the day by assembling our newly ordered closet in the evening. All these actions are underpinned by our subconscious mental models and an intuitive grasp of our environment.

However, as we enter an era dominated by technology, the way we interact with the world is poised for a significant transformation, and augmented reality (AR) stands at the forefront of this revolution. Augmented reality seamlessly integrates digital information with the physical world, offering the potential to enhance and augment our everyday experiences.

Consider the prospect of wearing AR glasses that transform your kitchen into a dynamic cooking assistant, guiding you through recipes with interactive step-by-step instructions. Envision navigating a bustling train station with real-time digital path planning cues overlaid on your view, simplifying your commute and reducing stress. Imagine using AR to provide step-by-step visual instructions for assembling furniture, making the process more intuitive and reducing the potential for errors and thereby frustration. To make this future a reality, we have to overcome many technological challenges.

For an augmented reality application to function seamlessly, it requires a 3D reconstruction of the world it is deployed to. This reconstruction serves several

purposes. Firstly, to naturally position virtual content within the real world, ensuring that virtual content does not intersect with physical objects. Additionally, it empowers the augmented reality system to adeptly manage occlusions, allowing them to discern whether the physical world is in front of or behind virtual content, even in dynamic settings bustling with moving people and objects. Furthermore, 3D reconstruction plays a pivotal role in assessing the compatibility of virtual content with its physical surroundings. For instance, envision the scenario where one seeks to virtually place furniture within an apartment and determine if it harmonizes with the space. Here, the accuracy of the reconstruction is paramount, as it holds the key to creating an immersive and authentic augmented reality experience.

In addition to robust 3D reconstruction, the understanding is paramount for the effectiveness of augmented reality. Beyond the spatial placement of virtual content, true value emerges when augmented reality systems possess an intricate understanding of the user's context. This contextual awareness is vital for seamlessly situating content within the user's environment, aligning the content precisely with their needs and activities. After all, it makes little sense to display cooking instructions upon entering the bedroom. To be genuinely helpful, AR devices must possess the capability to discern our location, identify surrounding objects, and interpret our actions accurately. Only this holistic understanding allows for a meaningful and personalized augmented reality experience, where digital elements seamlessly blend with the real world to enhance our daily lives.

Moreover, there are instances where the ability to peer behind physical objects proves immensely valuable. Consider the scenario where one wishes to envision a room devoid of its existing furniture, a necessity when contemplating a living room renovation with high-end designer pieces. While a select few possess the innate ability to conjure such imaginative feats, most individuals rely on professionals who craft renderings of their living spaces with new furniture. However, the rise of augmented reality and machine learning can democratize this capability. This fusion of technology empowers individuals to make informed decisions with unprecedented ease and accuracy.

The technology enabling these use cases require three primary technologies.

These technologies are more specifically - 3D scene reconstruction, 3D scene understanding, and 3D scene editing, which we will discuss in this thesis.

3D Reconstruction. 3D scene reconstruction is the process of creating a digital representation of the three-dimensional world from sensory input data. The ultimate goal of this endeavor is to achieve a highly accurate and faithful reconstruction of the physical environment. However, this task is not without its challenges, which span various aspects of the process. First, we need to address how to capture the necessary 3D measurements. These measurements can be derived from input images through techniques like photogrammetry or obtained directly from specialized 3D sensing technologies such as LiDAR or depth sensors. Once the measurements are acquired, the next consideration is what kind of information to store in the digital representation. Some applications may require solely geometric information, while others necessitate preserving the visual appearance of the scene. The inclusion of appearance data becomes crucial when the reconstructed scene is intended for visualization or rendering. Another pivotal aspect is the choice of an appropriate data structure to digitally represent the 3D scene. This choice depends on several factors, including the intended downstream application, the scale and complexity of the scene, the type of information being stored, and the specific reconstruction algorithm employed. Therefore, a comprehensive understanding of the entire reconstruction pipeline is essential. Furthermore, the fusion algorithms play a crucial role in aggregating the captured measurements into a globally consistent 3D model. The design of these algorithms is significantly influenced by whether the data processing occurs offline, where all data is collected before aggregation, or online, where data is acquired concurrently with the reconstruction process. This decision also ties back to the previous considerations regarding data structure, as certain structures may be better suited for offline processing, while others excel in an online processing scenario.

Devices, to which augmented reality applications are deployed to, usually reconstruct the world in an incremental process from sensorial input data. This incremental reconstruction is the main focus of the first part of this thesis. Particularly, we address the challenges of handling noise and outliers in the input

data, using machine learning for the reconstruction process of both geometry and appearance, and propose a novel scene representation for the incremental mapping of appearance mapping.

The methods to obtain the 3D information are far from perfect. The data has noise that becomes more severe the further away the 3D structure is. Further, especially when computing 3D geometry from images, contains many outliers. These outliers contaminate the 3D model if not handled adequately and negatively impact the downstream applications such as occlusion detection and rendering. Thus, this needs to be properly handled during the aggregation phase. Lastly, the sensor measurements are oftentimes incomplete, *i.e.* some parts are missing due to limitations of the acquisition method. *E.g.*, stereo methods usually have difficulties to reconstruct texture less areas while direct 3D sensing technologies have problems to sense structure that absorb all the emitted light and do not send any signal back to the sensor. Hence, the algorithms used to fuse the measurements need to be able to handle sensor noise, outliers, and ideally also be able to perform some form of completion of local structure.

Further, the problem of efficient aggregation of the measurements into a globally consistent 3D model is still unsolved. We address this issue from a machine learning perspective and propose a fusion pipeline that combines the advantages of existing methods with the benefits of data-driven machine learning. This is particularly challenging when it comes to online fusion of sensor measurements. The main challenge is that the machine learning algorithm effectively updates the scene representation in light of new information. This requires to adjust the reconstruction according to the new measurements but also keep valuable information from the past. We explore this avenue for pure geometric fusion but also appearance fusion. In regard to appearance fusion, we additionally explore a more efficient data structure that effectively combines the advantages of explicit surfel clouds with learned features.

3D Scene Understanding. A 3D model of the world alone is not sufficient for many applications; we also require an understanding of it. Therefore, it is essential to combine reconstruction with understanding. The ultimate objective is to gain insights into the world on various levels: at the scene level, object level, and even

at the primitive level, which encompasses points, voxels, or vertices. Typically, the fundamental form of understanding starts at the primitive level. This process is commonly referred to as segmentation, wherein each pixel (in 2D) or primitive (in 3D) is classified into a specific semantic category. Attaining an understanding of a scene at this level provides a foundation for addressing higher-level understanding tasks like object recognition, room delineation, or even building classification. After settling on the level of understanding, we need to choose an appropriate algorithm for processing the data and generating predictions for semantic and possibly instance labels. Similar to the reconstruction process, we also need to decide whether the scene is processed in an offline or online manner. In other words, we must determine whether we analyze all the data simultaneously or if the data arrives as a continuous stream. In the context of augmented reality, it is usually required that the process is online and incremental as the device is capturing more and more data about the user's environment.

When it comes to the challenges of 3D scene understanding, there are two primary aspects that we address in this thesis.

Firstly, a newer challenge has arisen in that, until now, understanding has been often reduced to classifying primitives into a fixed set of semantic classes, e.g., the NYU40 [Nathan Silberman and Fergus, 2012] classes. However, the world is vast and diverse, and the number of different object classes is virtually unlimited. Therefore, constraining the task to a fixed number of predefined classes is inherently limiting. To address this limitation, we need to approach the problem from a fresh perspective and expand the number of available classes to better capture the richness and diversity of the real world. As the number of labels grows, the need for data to train the understanding models on increases in parallel. Since most understanding models still follow the supervised learning paradigm, large-scale datasets have to be annotated. However, densely annotating large-scale datasets with hundreds of semantic classes is costly and does not scale well. Additionally, when crowd-sourcing this labelling effort, the overall annotation quality deteriorates and this deterioration in label quality leads to problems in model training and deployment. Thus, we propose a labelling pipeline that automatically annotates RGB-D trajectories.

Secondly, we must tackle how to effectively do incremental mapping and segmentation of a scene. The goal is to obtain and continuously update a semantic prediction for every point in the scene. It is crucial that these predictions incorporate both existing knowledge and new information as it emerge in the input data. This necessitates the incorporation of an appropriate receptive field into the algorithm. Many classification tasks require a degree of higher-level understanding. For example, distinguishing between a curtain in a bathroom (likely a shower curtain) and a curtain in a living room (usually a regular curtain) depends on the context of the room where the object is located. This context needs to be sufficiently large in the decision-making process. However, this expanded context comes at the cost of increased memory usage and runtime. Hence, there is a need to minimize the re-computation of large receptive fields in predictions by reusing existing context information. To this end, we propose a incremental semantic segmentation pipeline that incorporates the required receptive fields through by combining 2D and 3D information using a spatio-temporal expert network.

3D Scene Editing. The final essential component in the augmented reality pipeline is 3D scene editing, a critical task encompassing actions like adding, removing, and altering content within the scene and manipulating its appearance. While 2D image editing tools like Photoshop [[Adobe Systems, Inc., 1990](#)] have long facilitated these operations in two dimensions, the realm of immersive augmented reality demands the automation of these tasks in three dimensions. In this context, we may need to artificially eliminate real-world objects from the scene, modify them, or introduce entirely new virtual objects seamlessly.

Achieving these objectives involves a series of intricate steps. Firstly, we must identify the object in question and delineate its boundaries, often relying on prior semantic or instance segmentation. In the case of object removal, this identification is followed by precise removal and the subsequent filling of the resulting hole with appropriate structure. For adding new objects, considerations extend to estimating scene lighting, ensuring that the rendering of virtual content aligns realistically with the scene. This involves determining the source, color, and interaction of lighting with the added object. Moreover, operations such as moving, transforming, and altering existing objects can be distilled to the fundamental

processes of removal and addition, albeit with alterations in their position or form.

In this thesis, we focus on the removal of objects from 3D scenes. This involves the complete removal of the object, consistent inpainting of the created hole in the scene, and rendering novel views from the reconstruction, which is especially important for AR applications.

1.1 Scope of this work

In this work, we approach all three parts mentioned that are required for immersive augmented reality applications. We address 3D reconstruction, 3D scene understanding, and 3D scene editing.

In the case of 3D scene reconstruction, the main challenge is how to do online depth fusion from RGB-D imagery using learned components. *I.e.*, we explore how we can train and use neural networks for the task of online depth fusion in combination with learned scene representations. The fusion process needs to be reasonably fast and generate an accurate reconstruction of the scene given the incoming data. Thus, they need to be handle the various issues, *e.g.* noise, outliers, and missing data, caused by the different 3D acquisition methods such as 3D from stereo or direct 3D sensing.

Secondly, we address challenges in 3D scene understanding. In this part, we address the issue of having enough data that can be used to train and evaluate 3D scene understanding. Existing methods all rely on human annotated data. However, they are limited in scale and annotation quality. Therefore, we propose a method that allows to automatically annotate 3D scenes by combining multiple state-of-the-art models and aggregate their predictions into a single consensus. We show that the generated labels are on par with human annotations while being fully automatic. This allows scaling to an unseen scale in dataset size for dense annotation. In addition to the data problem, we also address the challenge of online semantic mapping. Existing methods either operate in 2D only and lift these predictions into 3D, or operate in 3D yet require a large-receptive field. We propose to combine 2D and 3D information in a local neighbourhood using an attention fusion mechanism. We show state-of-the-art results compared to other

local methods.

Thirdly, we explore 3D scene editing. More specifically, we address the challenge of removing objects from neural radiance fields. We show that exploiting inpainting capabilities of state-of-the-art 2D inpainting methods is a powerful prior for removing objects in 3D. We remove the object in all 2D images, inpaint the created whole using a 2D model, and then lift these inpaintings into 3D during NeRF training. The main problem to solve are the inconsistent 2D inpaintings between different views. If we do not handle them, we get blurry results and the 3D reconstruction and the renderings are contaminated with artifacts. Therefore, we introduce a uncertainty attenuation during NeRF optimization that captures disagreement between different inpaintings and votes for the most consistent inpaintings.

1.1.1 Contributions

This thesis is a combination of the following peer-reviewed publications and preprints.

[Weder et al., 2020]

RoutedFusion: Learning real-time depth map fusion

S. Weder, J.L. Schönberger, M. Pollefeys, M.R. Oswald

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition

2020

[Weder et al., 2021]

NeuralFusion: Online Depth Fusion in Latent Space

S. Weder, J.L. Schönberger, M. Pollefeys, M.R. Oswald

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition

2021

[Mihajlovic et al., 2021]

DeepSurfels: Learning online appearance fusion

M. Mihajlovic, S. Weder, M. Pollefeys, M.R. Oswald

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition

2021

[Weder et al., 2023a]

LabelMaker: Automatic Semantic Label Generation from RGB-D Trajectories

S. Weder, H. Blum, F. Engelmann, M. Pollefeys

Preprint

2023

[Weder et al., 2023b]

ALSTER: A Local Spatio-Temporal Expert for Online 3D Semantic Reconstruction

S. Weder, F. Engelmann, J.L. Schönberger, A. Seki, M. Pollefeys, M.R. Oswald

Preprint

2023

[Weder et al., 2023c]

Removing Objects from Neural Radiance Fields

S. Weder, G. Garcia-Hernando, A. Monzpart, M. Pollefeys, G. Brownstow, M. Firman, S. Vicente

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition

2023

Further, I have contributed to the following publication

[[Sandström et al., 2022](#)] **Learning online multi-sensor depth fusion**
E. Sandström, M. R. Oswald, S. Kumar, S. Weder, F. Yu, C. Sminichisescu, L. van Gool
European Conference on Computer Vision
2022

1.2 Outline

This thesis is structured into three parts. In part [II](#), we will present the contributions towards 3D reconstruction with a focus on online processing. We will motivate the methods in detail, review the relevant related work and present our contributions and their results. In part [III](#), we will dive into the topic of 3D scene understanding, where we again will give an overview over the field and discuss relevant related work. Then, we will discuss our automatic annotation pipeline and present the online semantic mapping approach. In part [IV](#), we will present our work on 3D scene editing with a focus on neural radiance fields. Finally, we will draw the conclusions from all presented research and give an outlook into the future of this field and all the challenges that are waiting for us.

Part II

3D Reconstruction

Introduction

As we have discussed in the introductory part of this thesis, seamless applications of augmented reality require some form of spatial awareness. This spatial awareness serves several purposes such as occlusion detection, virtual object positioning, and visualization. To enable this capability, we can draw inspiration how humans solve the problem of spatial awareness.

The world around us is 3D. Yet, we perceive its 2D projection and mentally build a spatial model of it. This mental model is required to perform any daily task such as grasping plates while doing the dishes, avoiding to hit your toes when walking around your table, or catching the ball that is thrown towards you. However, not only casual tasks as above require spatial perception, but also professional tasks such as carpentry. Humans build the spatial 3D model by relying on our multi-modal sensory input like vision, audio, and touch. Vision and audio is used to perceive large-scale scenes while we rely on touch to perceive fine details such as engraved names of loved ones in your jewelry. Further, we can rely on measuring techniques that allow us to build a model of a room to perform tasks such as constructing a house, fitting furniture, or calculating how much paint you need for repainting. As with everything in the world, all these models are only temporary, they get created, they get changed, and sometimes they get destroyed again.

So, how can computers accomplish the same task in our quest for smooth augmented reality experiences and its every day applications to make it valuable for humanity? This is the task of 3D modeling in computer vision. We aim to

develop 3D models of the world using a set of measurements \mathcal{M} , which can be 2D or 3D. Once these measurements are acquired, we assemble them into a consistent 3D model akin to assembling a Lego structure, where each piece must fit precisely. Which measurement fits where? How do we connect them? And maybe some parts are missing as Lego pieces might go missing. The created model can afterwards be used for downstream applications including visualizing your living room with new furniture or registering a digital model to the real machine a technician is repairing.

The first question we have to address is how to acquire the 3D measurements of the scene that we subsequently assemble into the globally consistent model. This is of utmost importance as depending on the type of measurement and the type of acquisition procedure the later parts of the 3D reconstruction pipeline are affected. Once we have acquired these measurements, we can ask ourselves how we represent the 3d scene internally. There are different options and all of them have their advantages and disadvantages for the next questions we aim to answer. Once we know the answer to these three questions, we can finally think about how to build the scene from the measurements. This requires to think about the quality of the data. Is there a lot of noise corrupting it? Do we expect to have outliers? Or might there even be some missing data? We need to know how to handle these challenges in order to get an as accurate as possible 3D reconstruction of the world. We know that the world is not static and ever changing. Thus, we have to finally ask the question on how we can adjust our 3D reconstruction to changes in the world. This is not only necessary when the world is changing but also if the measurements become available in a sequence and we cannot not look at them all at once.

This is also the specific problem that we focus on in this part of the thesis. How do we aggregate measurements into a 3D model of the world such that we can adapt and update the model to new data available? While previous methods proposed hand-crafted algorithms [Curless and Levoy, 1996a] or probabilistically [Dong et al., 2018] modelled the aforementioned noise and outlier distributions, we approach the challenge from a data-driven perspective. Hand-crafted methods suffer from several shortcomings such as surface thickening in presence

of sensor noise. We explore how neural networks can be leveraged for the task of depth map and RGB-D fusion and mitigate these issues. Further, we explore two different scene representation for online fusion to overcome limitations of explicitly storing the geometry of the scene during fusion. A feature-based scene representation in order to better model outlier handling during the fusion process as well as a surfel-based scene representation that combines the advantages of surfels [Pfister et al., 2000] with learned features to improve rendering quality. With these contributions, we aim to close the loop to gap between hand-crafted online reconstruction method and accurate RGB-D reconstructions that are required for seamless experiences in augmented reality.

The rest of this part is organized as following. We will give all necessary background information regarding 3D reconstruction to help you understand the big picture and introduce all necessary concepts. We will also review the related work in this field with a focus on the key topic of this part - online 3D reconstruction. Then, we will advance to answer the three key questions of this part: How do we effectively use machine learning for online 3D reconstruction? How do we exploit learned scene representation for online 3D reconstruction? And finally, how do we online appearance reconstruction?

Background

We have learned about the importance generating a digital 3D model of the world. Thus, this field has been central in computer vision for decades. In this chapter, we give an overview about the necessary background as well as review all relevant related work for this chapter.

3.1 From Measurements to 3D Model

When we build a spatial model of the 3D world, we start with some set of measurements $\mathcal{M} = \{m_1, m_2, \dots, m_n\}$ that are assembled into a 3D model of the world \mathcal{S} . This set of measurements can be anything from images to 3D point clouds. For the remainder of this thesis, we will use \mathcal{M} if the measurement type is kept general. Otherwise, we will define the type of measurement, *e.g.* images, RGB-D data, or (partial) point clouds. This reconstruction of the scene \mathcal{S} might only represent the geometry of the scene or also represent the appearance of the scene. This depends on the application the reconstruction is used for. *E.g.*, if we are only interested in occlusion detection for augmented reality, geometry is sufficient. But, if we want to stream our environment in a virtual reality call to our friends, we also need to represent the appearance of the world.

3.1.1 Obtaining the Measurements

Before we assemble the 3D model of the world, we need to collect information about the 3D world through sensor measurements. So, how do we obtain these measurements and the corresponding 3D information? There are different ways to obtain them. The key distinction is whether we directly measure the 3D structure of the world, or whether we obtain the 3D structure from images through geometric computations or neural network estimation.

From Images to 3D Measurements. We can obtain 3D measurements from images. This can be either by using geometric computations or a learned neural network that predicts dense depth maps. For geometric computations, the resulting form of 3D measurements can be sparse point clouds or dense depth maps obtained from two- or multi-view stereo. These different outputs are mostly computed using the same underlying geometric principles. In general, the underlying principle is to find the projection of one 3D point in two or more images and then triangulate the 3D point given the known camera positions. The main challenge is to find these corresponding 2D projections in the different images. Therefore, the general pipeline detects and describes features in all images. These features can either be sparse or dense depending on the method. Then, these features are matched between images to find the corresponding 2D projections of a 3D point. Finally, the corresponding 2D points are triangulated in order to obtain the final 3D point. Obviously, there are many relevant details along the way such as uncertainty estimation, handling outlier matches, and jointly estimating the camera pose. However, in the scope of this thesis we only explain how the general principle works. For more details, the following papers give an good introduction in this specific field [Schönberger and Frahm, 2016, Agarwal et al., 2011].

For dense two-, or multi-view stereo, the high-level idea is similar, however it is usually approached by computing a matching cost volume or the matching cost for local patches and minimize it to find good disparity values. Historically popular approaches are the plane-sweep algorithm [Gallup et al., 2007] and PatchMatch [Bleyer et al., 2011]. A good review over this field can be found in [Stathopoulou and Remondino, 2023]. More recently, these methods have been combined with neural networks, *e.g.* [Duzceker et al., 2021, Sayed et al., 2022],

where the feature extraction, the matching, or the cost volume computation is replaced by a learned variant.

Nowadays, two-view stereo algorithms are available through off-the-shelf sensors such as Stereolabs ZED 2 [Stereolabs, 2019] and Intel RealSense [Intel, 2014]. These sensors provide convenient access to stereo depth maps as an end-to-end solution combining camera and stereo algorithm in one device. The performance can be easily improved by coupling the stereo with an infrared pattern and therefore enabling active stereo vision.

Alternatively to the the geometric approach, there is an entire line of work, that directly estimates dense depth maps from images or videos using a neural network [Fu et al., 2018a, Eigen and Fergus, 2015, Li and Snavely, 2018, Ranftl et al., 2022, Ranftl et al., 2021]. These methods are usually called monocular depth estimation. However, they usually predict relative depth and require additional information and processing to be exploited in metric 3D reconstruction.

Directly Sensing the 3D Structure. In contrast to humans, machines can also directly sense the 3D structure of the world. This can be done using different types of 3D sensors. One common type of 3D sensor are LiDARs that measure the time of flight of a laser ray. For that, they actively emit a laser beam and measure the time-of-flight until the signal returns to the sensor. From this time-of-flight, they compute the distance between the sensor and the reflecting structure. LiDAR are usually accurate yet also expensive. Examples for these products are [Geosystems, 2016, NavVis, 2021, Velodyne, 2019] With the introduction of LiDAR in Apple iPhones, LiDARs have been commoditized into consumer devices and made conveniently available to dense reconstruction applications through the ARKit [Apple, 2017] framework. The main limitation of LiDARs are their high cost and oftentimes sparse measurements requiring additional densification through post-processing.

A cheaper alternative are structured light or time-of-flight sensors such as the different versions of Microsoft Kinect or Asus Xtion. Structure light sensor work by projecting a pattern onto the 3D world. This pattern is deformed by the 3D geometry of the scene. The deformed pattern is then used to measure disparity and compute the associated depth. *I.e.*, structured light sensors are active stereo

methods that also work on areas that have no texture. However, they suffer in direct sunlight as the sun's light is interfering with the infrared pattern. Examples for structured light sensors are PrimeSense Carmine [[PrimeSense](#),] and Microsoft Kinect v1. Much like LiDARs, time-of-flight sensors emit a signal and calculate the time it takes for the signal to travel to an object and return to the sensor. This yields accurate measurements of the 3D structure of the world. However, they suffer under various conditions such as when scanning outdoors or when capturing reflective surfaces. Microsoft Azure Kinect [[Microsoft, 2020](#)] and ASUS Xtion 2 are prominent examples of time-of-flight sensors.

3.1.2 Aggregating the Measurements

Once the measurements about the 3D world have been obtained, they have to be aggregated into a consistent three-dimensional model. This is the key focus of this part of the thesis. All of the sensing methods mentioned above have different challenges. Stereo methods usually suffer from outlier contamination while direct 3D sensors have a depth-dependent noise pattern. And almost all of them, fail to sense the complete structure and produce incomplete measurements. *E.g.*, stereo methods have problems to reconstruct texture less areas and time-of-flight sensors fail to capture computer screens or reflective surfaces. Therefore, an optimal aggregation or fusion algorithm handles all these challenges. Further, the aggregation can take place in two different forms. Offline aggregation looks at all data at once after the data has been collected. Online aggregation handles a stream of incoming data in scenarios, where we can not wait until the entire scene has been scanned. We focus on online aggregation as most scenario in augmented reality require an incremental mapping of the world around us. However, before we aggregate the measurements, we have to select an adequate scene representation for the aggregation. We discuss this next.

The remainder of this chapter is organized as following. We first review the different scene representations for 3D reconstruction. Then, we discuss the differences between offline and online methods and review relevant related work. Afterwards, we review existing work on online reconstruction and its different flavours. Finally, we review the literature about joint reconstruction of geometry

and appearance.

3.2 Scene Representations for 3D Reconstruction

The choice of scene representation is of utmost importance. It defines the amount of memory the reconstruction needs. It also defines how easily we can change the reconstruction when new information becomes available. Thus, this choice has to reflect the downstream applications and use-cases that the pipeline is designed for. In the following, we discuss the advantages and disadvantages of different scene representations. In general, we distinguish between non-learned and learned scene representations.

3.2.1 Non-learned Representations

The major advantage of explicit, non-learned geometric representations is their direct interpretability. This allows to have full control over what is stored in the scene representation. The most common types of explicit scene representations are point clouds, meshes, voxel-grids, and surface elements (surfels).

Point Clouds. Point clouds are often the raw output of many 3D scanners such as LiDARs as well as sparse reconstruction pipelines such as COLMAP [Schönberger and Frahm, 2016]. Point clouds are a discrete set of points represented by their 3D coordinates x , y , and z . In addition to their 3D coordinates, each point is can be associated with specific point-wise features such as color, normals, or even learned features. Point clouds offer a distinct advantage due to their lightweight representation, which simplifies processing and manipulation. However, this lightweight structure comes at the cost of limited scalability and increased run-time of simple post-processing steps. One major limitation is the lack of topology and connectivity information. *I.e.*, if you want to figure out the local neighbourhood of a specific 3D point, you need to run expensive nearest neighbour search as the neighbourhood, unlike other representations, is not implicitly defined. This limitation makes it hard to extract watertight surfaces from raw point clouds without expensive post-processing [Hoppe et al., 1992, Kazhdan et al., 2006, Berger et al., 2014]. Therefore, point clouds are not a standard representation in

modern dense 3D reconstruction pipelines.

Triangle Meshes. Triangle meshes are a common representation in computer graphics. Triangle meshes store both vertex points and triangles. Each triangle stores the three vertex indices it is constructed from. *I.e.*, the three indices of the triangle can be used to obtain the world positions of its corners from the set of vertices. Moreover, additional geometry or appearance features can be associated with each vertex or face. Triangle meshes are suitable for rendering applications as basic rendering is simply intersecting camera rays with the mesh and render back the appearance information at these locations. Extensive research has been invested into this direction and improvements have been realized through rasterization and hardware acceleration. Further, triangle meshes scale well to large scenes and texture mapping is convenient that is necessary for subsequent rendering. While triangle meshes dominate use cases, where the focus is on rendering, they are rarely used in reconstruction. Usually, meshes are extracted from point clouds, voxel-grids, or surface elements in a post-processing step, *e.g.* using marching-cubes [Lorenson and Cline, 1987]. However, they are less suitable as a representation for online reconstruction due to their fixed topology. This is because the topology of a scene might change during online reconstruction as new data becomes available and these changes need to be incorporated into the representation. Therefore, triangle meshes are less suitable for online reconstruction.

Voxel-Grids. Voxel-grids naturally extend the notion of a 2D pixel to the 3D world. They simply represent the world by square cubes that store information about the scene - similar to building a 3D model from LEGO. Voxels can store information about both geometry and appearance. The geometry is usually represented explicitly using occupancy o , where $o = 1$ indicates that the voxel is occupied and $o = 0$ indicates that it is free. It can also be represented implicitly using a signed distance value s , which represents the distance to the closest surface. A value $s < 0$ means it is inside the geometry and a value $s > 0$ means a specific voxel is outside the geometry. If the value $s = 0$, then the voxel is exactly on the surface. Further, they can additionally store information about normals if it is necessary for downstream applications. Furthermore, they can store surface

normals when required for downstream applications. Appearance is typically represented by storing the color in the voxels. Like point clouds, voxel-grids are easy to handle using standard tooling. However, their scalability is limited, leading to significant research efforts aimed at enhancing the scalability of voxel representations. A notable work in this direction is voxel hashing [Nießner et al., 2013], which proposes to store only voxels close to a reconstructed surface in a hash table.

Surface Elements (Surfels). Surface elements (surfels) are an extension of point clouds that is adapted to rendering. Surfels are non-connected point primitives yet store all essentials that are needed for efficient rendering. They were initially proposed in [Pfister et al., 2000]. They have been adapted to online reconstruction in SLAM systems by [Whelan et al., 2016], [Wang et al., 2019b], and [Schöps et al., 2019]. Due to their properties they are both very efficient for reconstruction and rendering and thus ideal for online appearance reconstruction as there the goal is to capture and reconstruct both appearance and geometry and be able to subsequently render images from the scene.

3.2.2 Learned Representations

In addition to explicit scene representation, learned scene representations have been proposed in recent years. The general idea is to represent the geometry and appearance of the scene implicitly as a learned function approximator or as a combination of an explicit scene representation (*e.g.* voxel-grids or point clouds) with learned features. In the case of a learned function approximator, the input is a point coordinate and the output is the occupancy, density, or signed distance at the given location. In the case of explicit learned features, a learned function usually decodes them into an interpretable geometry such as signed distance, occupancy, or density.

In recent years, a plethora of work emerged proposing different learned scene representations. [Ladicky et al., 2017] directly estimate an iso-surface from a point cloud and learned local point features using a random forest. Alternatively, there exist multiple proposals for methods that learn 3D reconstruction in an implicit space using a neural network [Mescheder et al., 2019, Park et al., 2019, Chen

and Zhang, 2019, Michalkiewicz et al., 2019, Xu et al., 2019a]. In contrast to [Ladicky et al., 2017], these works encode the geometry into the weights of a neural network and the geometry can be queried by passing the 3D coordinates through the network retrieving the local geometry represented by either occupancy or a signed distance field at that location. These methods show promising results, but they operate only on a unit cube and are thus limited to single objects or small scenes and they are not suited for online reconstruction. Several methods have explored different conditioning and supervision strategies for the learned implicit representation. [Kar et al., 2017, Huang et al., 2018, Saito et al., 2019, Saito et al., 2020, Choy et al., 2016, Xu et al., 2019a] propose to condition the network on a single or multiple input images for shape reconstruction. Recently, several works proposed a more local scene representation [Genova et al., 2019a, Genova et al., 2019b, Badki et al., 2020] that allows larger scale scenes and multiple objects. The issue of only operating on a unit cube has been also addressed by [Chabra et al., 2020, Chibane et al., 2020, Jiang et al., 2020, Peng et al., 2020] which use multiple features to encode parts of the scene and scale the implicit representation to large-scale scenes like rooms and apartments or even entire buildings. These approaches struggle to scale to larger scenes and to capture high-frequency details as they tend to learn low-frequency functions, which often results in over-smoothed geometry [Rahaman et al., 2019].

Early works on learned scene representations like [Mescheder et al., 2019, Park et al., 2019, Chen and Zhang, 2019, Michalkiewicz et al., 2019, Xu et al., 2019a, Chabra et al., 2020, Chibane et al., 2020, Peng et al., 2020] all require 3D supervision. The next step was to supervise the scene representation using 2D information such as depth maps and RGB images. Therefore, [Liu et al., 2020a, Niemeyer et al., 2020, Liu et al., 2019a] learn implicit representations with 2D supervision via differentiable rendering. This is also important if one not only wants to represent the geometry but also the appearance of the scene.

Learned Representations for Geometry and Appearance. Similar to representing geometry, learned representations have achieved state-of-the-art results for representing appearance. They encode visual information into learned features and store them in voxel-grids [Flynn et al., 2019, Mildenhall et al., 2019, Penner and

Zhang, 2017, Srinivasan et al., 2019, Lombardi et al., 2019, Rematas and Ferrari, 2020], point clouds [Aliev et al., 2020], or meshes [Thies et al., 2019, Riegler and Koltun, 2020, Zhang et al., 2021] which are rendered using neural networks.

[Oechsle et al., 2019, Oechsle et al., 2020] use a neural network conditioned on geometry to generate a learned texture representation. [Niemeyer et al., 2020] combines geometry and appearance to generate a joint implicit representations of the scene. Worrall *et al.* [Worrall et al., 2017] learn a disentangled representation of object pose, appearance, illumination, and other properties to interpret and manipulate learned feature-based scene representations. Other works [Saito et al., 2019, Saito et al., 2020] take advantage of local features for higher representation power, while [Huang et al., 2020, Thies et al., 2020] uses appropriate loss terms to correct for geometric misalignment. Recent trends and applications of neural renderers are summarized in [Tewari et al., 2020].

The global volumetric appearance reconstruction approach [Bi et al., 2020] additionally separates albedo, roughness, and lighting. [Liu et al., 2019] present a learned approach for shape and texture reconstruction that linearly fuses shape and color information in a voxel grid as in [Curless and Levoy, 1996a] and post-process the grid with a multi-resolution neural network. However, pure post-processing methods may not be able to revert errors of an incorrect earlier linear fusion. [Sitzmann et al., 2019a] represent the scene as learned-features stored in a voxel grid and renders images for 2D supervision. This has been extended by Scene Representation Networks [Sitzmann et al., 2019b] that replaces the voxel-grid by a neural network that is conditioned on a hyper-network. Then, a major revolution happened in the field. [Mildenhall et al., 2020a] pushed the idea of scene representations one step further by using volume rendering and adding positional encoding to the coordinate processing. The simplicity of the method in combination with the impressive results kicked off an entire new field of research in the direction of neural radiance fields. While all these methods propose an interesting direction for representing geometry and appearance, they are not suitable for online processing that is required for many tasks in the real-world.

In this thesis, we focus on online processing as it is a key requirement for many applications in the real-world such as enabling augmented reality on consumer

devices. Thus, we explore how we can use learned scene representations for online fusion of depth and RGB-D data. To this end, we explore how to use a feature-based scene representation for depth fusion in chapter 5 and propose a novel surfel-based scene representation that stores learned features for appearance mapping in chapter 6.

3.3 Global Depth Fusion

Once we have obtained individual 3D measurements of the scenes and have decided on a scene representation that we discussed before, we fuse the measurements into a globally consistent 3D model. The main challenges during this task are to deal with imperfect data (noise, outliers) and missing data. Methods that do global depth fusion assume to have all measurements available and no new measurements are obtained afterwards.

While online approaches only process one depth map at a time, global approaches use all information at once and typically apply additional smoothness priors like total variation [Zach et al., 2007, Kolev et al., 2009], its variants including semantic information [Häne et al., 2013, Cherabier et al., 2016, Häne et al., 2017, Savinov et al., 2015, Savinov et al., 2016], or refine surface details using color information [Zollhöfer et al., 2015a]. Consequently, their high compute and memory requirements prevent their application in online scenarios unlike ours.

Several learning-based methods have been proposed to fuse, estimate, or improve geometry. Octnet [Riegler et al., 2017b] and its follow-up OctnetFusion [Riegler et al., 2017a] fuse depth maps using TSDF fusion into an octree and then post-processes the fused geometry using machine learning. RayNet [Paschali-dou et al., 2018] uses a learned Markov random field and a view-invariant feature representation to model view dependencies. SurfaceNet [Ji et al., 2017] jointly estimates multi-view stereo depth maps and the fused geometry, but requires to store a volumetric grid for each input depth map, which is unnecessarily memory demanding. 3DMV [Dai and Nießner, 2018] combines 2D view information with a pre-fused TSDF scene to jointly optimize for shape and semantics. In [Leroy et al., 2018], multi-view consistency is learned upon classical TSDF fusion.

Moreover, hierarchical volumetric deep learning-based approaches [Cao et al., 2018, Cherabier et al., 2018, Dai et al., 2018] tackle the effects of noisy measurements, outliers, and missing data. All these approaches operate on a voxel-grid with high memory demands and are not real-time capable. Further, there are several works that learn to predict 3D meshes based on input images [Groueix et al., 2018, Gkioxari et al., 2019, Wen et al., 2019]. In contrast to global depth fusion methods, we focus on online depth fusion.

3.4 Online Depth Fusion

Online depth fusion addresses the shortcomings of global depth fusion methods that prevents them from being applied for online processing. Thus, we now dive into the field of online 3D reconstruction. This section is organized along the type of fusion and scene representation. We start with volumetric depth map fusion, move to surfel-based and probabilistic depth map fusion, and finish with learning-based depth map fusion methods.

With their seminal work, Curless and Levoy [Curless and Levoy, 1996a] proposed an elegant way for fusing noisy depth maps, which later got adopted by numerous works like KinectFusion [Izadi et al., 2011], more scalable generalizations like voxel hashing [Nießner et al., 2013, Marniok and Goldluecke, 2018], or hierarchical scene representations, such as voxel octrees [Fuhrmann and Goesele, 2011, Steinbrücker et al., 2013, Marniok et al., 2017] and hierarchical hashing [Kähler et al., 2016]. Especially for SLAM pipelines like InfiniTAM [Kähler et al., 2015], volumetric fusion became a standard approach due to its real-time capability. In this context, it was also extended to become more accurate and robust [Choi et al., 2015] as well as improve SLAM with additional surface registration of scene parts to account for pose drift as proposed in [Whelan et al., 2016, Maier et al., 2017b, Dai et al., 2017b]. Approaches with additional median filtering [Rothermel et al., 2016, Marniok et al., 2017, Marniok and Goldluecke, 2018] improve the robustness and are still real-time capable but with limited effectiveness. All these methods handle noisy measurements by updating a wider band of voxels around the measured depth leading to thickening artifacts on thin

geometry. Moreover, the output of these methods usually contains typical noise artifacts, such as surface thickening and outlier blobs, which can be tracked back to the original formulation of [Curless and Levoy, 1996a].

3.4.1 Surfel-based Fusion Methods.

Surfel-based methods approximate the surface with local point samples, which can further encode additional local properties such as normal or texture information. Multiple methods have been proposed, *e.g.* MRSMMap [Stückler and Behnke, 2014] uses an octree to store multi-resolution surfel data. Point-based fusion methods [Keller et al., 2013, Lefloch et al., 2015] combine a surfel representation with probabilistic fusion discussed in the next paragraph. ElasticFusion [Whelan et al., 2016] handles real-time loop closures and corrects all surface estimates online. [Schöps et al., 2019] proposed a depth fusion approach with real-time mesh construction. A disadvantage of surfel-based methods is the missing connectivity information among surfels. The unstructured neighborhood relationships can only be established with a nearest neighbor search or simplified with space partitioning data structures. Therefore, we decided to rely on volumetric representation in chapters 4 and 5. However, extending our approach to unstructured settings is an interesting avenue of future work.

3.4.2 Probabilistic Depth Map Fusion.

To account for varying noise levels in the input depth maps and along different line-of-sight directions, the fusion problem can also be cast as probability density estimation [Duan et al., 2012] while typically assuming a Gaussian noise model. Keller *et al.* [Keller et al., 2013] propose a point-based fusion approach which directly updates a point cloud rather than a voxel grid. Lefloch *et al.* [Lefloch et al., 2015] extended this idea to anisotropic point-based fusion in order to account for different noise levels when a surface is observed from different incident angles. The mesh-based fusion approach by [Zienkiewicz et al., 2016] allows for depth fusion across various mesh resolutions for known fixed topology. The probabilistic fusion method by [Woodford and Vogiatzis, 2012] incorporates long range visibility constraints. Similar ray-based visibility constraints were also used

in [Ulusoy et al., 2015, Ulusoy et al., 2016], but these methods are not real-time capable due to the complex optimization of ray potentials. Anisotropic depth map fusion methods additionally keep track of fusion covariances [Ylimäki et al., 2018]. Similarly, PSDF Fusion [Dong et al., 2018] explicitly models directional dependent sensor noise. However, all these approaches assume particular noise distributions, primarily Gaussians, which often do not model the real sensor observations correctly. We aim to address this shortcoming in this thesis by approaching depth fusion from a data-driven perspective.

3.4.3 Learned Depth Map Fusion

In the context of simultaneous localization and mapping, CodeSLAM [Bloesch et al., 2018], SceneCode [Zhi et al., 2019] and DeepFactors [Czarnowski et al., 2020] learn a 2.5D depth representation and its probabilistic fusion rather than fusing into a full 3D model. The DeepTAM [Zhou et al., 2020] mapping algorithm builds upon traditional cost volume computation with hand-crafted photoconsistency measures, which are fed into a neural network to estimate depth, but full 3D model fusion is not considered. DeFuSR [Donné and Geiger, 2019] refines depth maps by improving cross-view consistency via reprojection, but it is not real-time capable. More recently, [Huang et al., 2021a] proposed to combine implicit models with online depth fusion using test-time optimization.

3.4.4 Learned RGB fusion

The recent ATLAS [Murez et al., 2020] method fuses features from RGB input into a voxel grid and then regresses a TSDF volume. While our method learns the fusion of features, they use simple weighted averaging. Their large ResNet50 backbone limits real-time capabilities. This challenge of directly mapping 3D structure from a stream of RGB data either through feedforward predictions or test-time optimization has been approached in an end-to-end framework by several methods. Works like [Sun et al., 2021, Sayed et al., 2022, Stier et al., 2021, Bozic et al., 2021] directly reconstruct a 3D mesh from a stream of RGB input data given known poses using different neural network architectures. [Sucar et al., 2021] and

[Zhu et al., 2022] combine advances test-time optimization for 3D reconstruction from RGB data with pose optimization into a full SLAM pipeline.

3.5 Appearance Reconstruction

So far, we have only considered geometric 3D reconstruction of the world around us. However, it is oftentimes required to also obtain a reconstruction of the appearance - *i.e.*, we reconstruct how the world looks like in colours. There is a long line of research that aims to address this challenge. Similar to geometric reconstruction, most work has focused on global appearance mapping. However, online appearance mapping is essential for many tasks, where online geometric mapping is required. Thus, we will first review the relevant literature on classical (global) texture mapping and then look at approaches that tackle online appearance reconstruction.

3.5.1 Classical Texture Mapping.

The classical way of coloring a surface from a set of input images with known camera pose is to un-project the image information onto the surface and perform a selection or blending operation to fuse the color information [Debevec et al., 1996, Wood et al., 2000, Allène et al., 2008]. Due to errors in the camera alignment or in the surface geometry, blurry textures or patch seams affect results and additional texture alignment procedures have been proposed [Lensch et al., 2001, Bernardini et al., 2001, Theobalt et al., 2007, Eisemann et al., 2008, Gal et al., 2010, Waechter et al., 2014, Lempitsky and Ivanov, 2007, Takai et al., 2010, Fu et al., 2020] to tackle these problems. Better texture mapping results have been achieved with an optical flow-like correction in texture space [Eisemann et al., 2008, Waechter et al., 2014, Fu et al., 2018b], patch-based optimization [Bi et al., 2017], or via 2D perspective warp techniques [Lee et al., 2020]. With significantly more computation effort, it is also possible to better leverage the redundancy of multiple surface observations from different views and to compute super-resolved texture maps via energy minimization [Goldlücke et al., 2014, Tsiminaki et al., 2014, Fu et al., 2018b, Tsiminaki et al., 2019] or with deep learning techniques [Li

et al., 2019, Richard et al., 2019]. All previously mentioned methods share the strategy of aggregating appearance information in patches or texture atlases with corresponding coordinates onto a mesh-based surface, while other works use voxel grids [Newcombe et al., 2011a, Zollhöfer et al., 2015b, Maier et al., 2017b, Kutulakos and Seitz, 1999, Seitz and Dyer, 1999, Szeliski and Golland, 1998], or mesh colors [Yuksel et al., 2010, Armando et al., 2019]. An overview of texture mapping methods with different representations is given in [Tarini et al., 2017, Yuksel et al., 2019].

3.5.2 Online Appearance Aggregation.

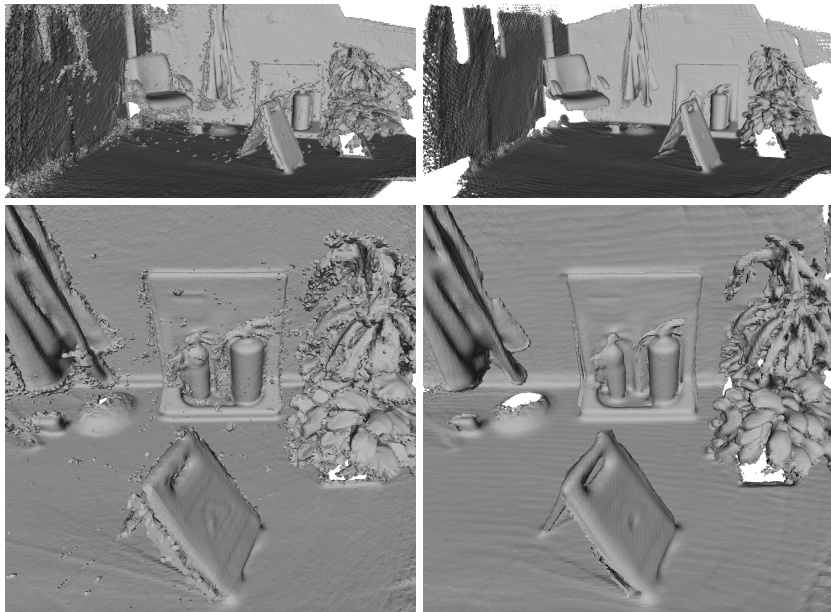
The previously discussed texture mapping methods process all input images in a batch-based way after the geometry estimation step and are implemented as a separate post-processing step, whereas only a minority addresses the problem of online appearance reconstruction. A popular work is KinectFusion [Newcombe et al., 2011a] and related works [Zollhöfer et al., 2015a, Maier et al., 2017b, Maier et al., 2017a, Lee et al., 2020], which estimate surface and appearance information from a stream of RGB-D images. Other works fuse both geometry and appearance information directly into an oriented surfel cloud [Schöps et al., 2019, Whelan et al., 2016, Wang et al., 2019b]. The vast majority of these approaches directly fuse RGB-D images for which Zollhöfer *et al.* [Zollhöfer et al., 2018] provide a recent survey.

The major drawback of these methods is limited capacity to store high-frequency appearance along the surface. This limited capacity to store high-frequency appearance leads to blurry results when rendering images from the scene representation decreasing the overall quality of the rendered images. Therefore, we propose an efficient online appearance estimation pipeline in chapter 6 mitigating these limitations.

Learning-based Depth Map Fusion

As we have learned, reconstructing the 3D world around us is essential for overlaying the real with the virtual world. Devices that enable augmented reality experiences typically require a constantly updated model of the 3D world around us. This is why the reconstruction has to happen on the fly and needs to be adjustable to new information that is captured using different types of sensors. While we have reviewed different methods to obtain 3D measurements, we focus on leveraging 3D sensors that directly measure the structure of the world in this and the next chapter. In other words, we address the problem of online depth fusion. The problem of fusing depth maps from multiple camera viewpoints has been addressed in recent 3D reconstruction pipelines [Zach et al., 2007, Zach, 2008, Kolev et al., 2009, Bláha et al., 2016, Savinov et al., 2015, Savinov et al., 2016, Dai et al., 2018, Dai and Nießner, 2018], especially for real-time applications [Izadi et al., 2011, Nießner et al., 2013, Whelan et al., 2016, Dai et al., 2017b].

In this chapter, we revisit the problem of 3D reconstruction via depth map fusion from a machine learning perspective. The major difficulty of this task is to deal with various amounts of noise, outliers, and missing data. The classical approach [Curless and Levoy, 1996a, Izadi et al., 2011] to fusing noisy depth maps is to average truncated signed distance functions (TSDF). This approach



Standard TSDF Fusion
[Curless and Levoy, 1996a]

Ours

Figure 4.1: Standard TSDF fusion vs. our learned depth map fusion approach (on Kinect data [Shotton et al., 2013]). Due to a more informed decision process, our approach better handles noise and fine geometric details.

has many advantages: First, the updates are local (truncated) and can be done in constant time for a fixed number of depth values. The high memory usage of voxel grids can be easily reduced with voxel hashing [Nießner et al., 2013] or octrees [Steinbrücker et al., 2013]. Second, online updates are simple to implement and noisy measurements are fused into a single surface with few operations. Third, the approach is computationally cheap and highly parallelizable due to locally independent updates.

However, the approach also has a number of shortcomings: First, the average is only the optimal estimate for zero-mean Gaussian noise, but the real error distribution is typically non-Gaussian, non-centered and depth-dependent. Second, the updates are linear and a minimal thickness assumption of surfaces has to be made according to the expected noise level. Therefore, thickening artifacts become

apparent along surface edges and for thin object structures. Third, this issue is even more severe when depth measurements of a thin object are made from opposite directions. In this case, the surface vanishes since the linear TSDF updates cancel each other out. Moreover, the linear fusion weights do not properly account for view-directional dependencies during TSDF aggregation. The noise level along the viewing direction is typically very different from the one in orthogonal directions. Further, the fusion approach is unable to handle gross outliers. The depth map has to be pre-filtered or incorrect measurements will clutter the scene. Finally, the fusion parameters must be tuned for specific scenes and sensors and it is often difficult to find a good trade-off between runtime and different aspects of reconstruction quality.

We aim to tackle the disadvantages mentioned above while maintaining all the advantages of traditional approaches with a reasonable amount of additional computation time to still meet real-time requirements. To this end, we propose a learned approach, which we term RoutedFusion, that fuses noisy and outlier contaminated measurements into a single surface, performs non-linear updates to better deal with object boundaries and thin structures, and is fast enough for real-time applications. Figure 4.1 shows example outputs of our approach.

In summary, the contributions of this chapter are as follows:

- We present a learning-based method for real-time depth map fusion. Due to its compact architecture it requires only little training data, and is not prone to over-fitting.
- We propose a scalable and real-time capable neural architecture that is independent of the scene size. Therefore, it is applicable to a large set of real-world scenarios.
- We show a significant improvement of standard TSDF fusion’s shortcomings: 1) It better handles the fusion of anisotropic noise distributions that naturally arise from the multi-view setting, and 2) It mitigates the surface thickening effect on thin objects and surface boundaries by avoiding inconsistent updates.

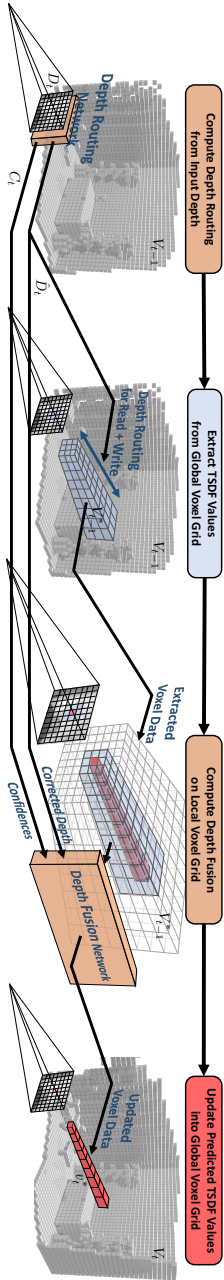


Figure 4.2: System overview for integrating depth maps into a global TSDF volume. A *2D Depth Routing Network* takes depth input and decides on the update location for every ray within the TSDF volume. The network corrects for noise, outliers and missing values and further estimates per-ray confidence values. Then, for each ray, we extract a depth and view-dependent local voxel grid (light blue) which also includes neighboring rays. We sample S values along each ray, centered around the surface. A *Depth Fusion Network* then takes the local grid of existing TSDF values, the depth and confidences to predict adequate updates. The predicted TSDF values (red) are then written back into the global volume. Our method learns a robust weighting of input depths and performs non-linear updates to better handle noise, outliers, and thin objects.

4.1 Method

Before we look at the problem of depth map fusion from a machine learning perspective, we review the essentials of standard TSDF fusion to provide context and introduce the necessary notation.

4.1.1 Review of Standard TSDF Fusion

Standard TSDF fusion integrates given depth maps $D_{t=1,\dots,T} \in \mathbb{R}^{W \times H}$ from known viewpoints $P_t \in \text{SE}(3)$ with camera intrinsics K_t into a discretized signed distance function $V_t \in \mathbb{R}^{X \times Y \times Z}$ and weight function $W_t \in \mathbb{R}^{X \times Y \times Z}$ defined over the entire scene. The fusion process is incremental, *i.e.* each depth map is integrated after one another for location \mathbf{x} using the update equations introduced by Curless and Levoy [Curless and Levoy, 1996a] as

$$V_t(\mathbf{x}) = \frac{W_{t-1}(\mathbf{x}) \cdot V_{t-1}(\mathbf{x}) + w_t(\mathbf{x}) \cdot v_t(\mathbf{x})}{W_{t-1}(\mathbf{x}) + w_t(\mathbf{x})} \quad (4.1)$$

$$W_t(\mathbf{x}) = W_{t-1}(\mathbf{x}) + w_t(\mathbf{x}) \quad , \quad (4.2)$$

starting from zero-initialized volumes V_0 and W_0 . The signed distance update v_t and its corresponding weight w_t integrate the depth measurements of the next depth map D_t at time step t into the TSDF volume. These update functions are traditionally truncated before and after the surface in order to ensure efficient runtimes and robust reconstruction of fine-structured surfaces given noisy depth measurements.

The choice of the truncation distance parameter typically requires cumbersome hand-tuning to adapt to a specific scene and depth sensor as well as accounting for runtime. If the truncation distance is chosen too large, the reconstruction of thin structures becomes more difficult due to larger thickening artifacts and the fusion process gets slower since more voxels have to be updated for each ray. Contrary, a small truncation distance results in time efficient updates but cannot deal with larger noise in the depth measurements.

In this paper, we overcome this limitation by learning the function v_t automatically from data. Our system is based on the same update equations described

above and our learned functions have only little computational overhead compared to traditional TSDF fusion. As such, our method facilitates real-time depth map fusion and can be readily integrated into existing reconstruction systems. In the following, we describe our proposed method in more detail.

4.1.2 Pipeline Overview

Our method contains *two* network components: a **depth routing network** and a **depth fusion network**. The pipeline consists of the following *four* essential processing steps which are also illustrated in Figure 4.2:

1. **Depth Routing:** The depth routing network takes a raw depth map D_t and estimates a denoised and outlier-corrected depth map \hat{D}_t , and further estimates a corresponding confidence map C_t . This network *routes* the depth location for reading and writing TSDF values along each viewing ray. Moreover, the denoised depth estimate allows to extract a smaller window in the next step for improved efficiency.
2. **TSDF Extraction:** Given the routed depth values \hat{D}_t , we extract a local camera-aligned voxel grid with TSDF data V_{t-1}^* and weight W_{t-1}^* via trilinear interpolation from the corresponding global voxel grids V_{t-1} , W_{t-1} .
3. **Depth Fusion:** The depth fusion network takes the results of the previous processing steps ($\hat{D}_t, C_t, W_{t-1}^*, V_{t-1}^*$) and computes the local TSDF update v_t^* .
4. **TSDF Update Integration:** The predicted TSDF update v_t^* is transferred back into the global coordinate frame to get v_t which is then integrated into the global TSDF volumes V_t, W_t using the TSDF updates in Eqs. (4.1), (4.2).

These processing steps are detailed in the next subsections.

4.1.3 Depth Routing

Using the depth routing network, we pre-process the depth maps before passing them to the depth fusion network with the main motivation of denoising and outlier

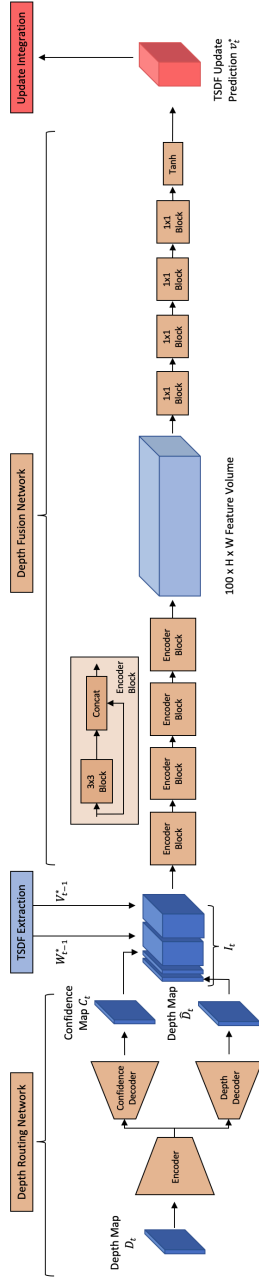


Figure 4.3: Proposed network architecture. Our depth routing network consists of a U-Net (depth one) with two separate decoders predicting a corrected depth map and a corresponding confidence map. The depth fusion network extracts in a series of encoding blocks 100 features along each ray. These features are then used to predict the TSDF updates along the ray.

correction. Towards this end, the network predicts denoised depth maps and also per-pixel confidence maps $C_{t=1,\dots,T} \in \mathbb{R}^{W \times H}$. Figure 4.3 illustrates our network architecture, which is using a fully-convolutional U-Net [Ronneberger et al., 2015] with a joint encoder and separate decoders for confidence and depth prediction. Further, we do not use normalization layers since it negatively influences the depth prediction performance by adding a depth-dependent bias to the result. The depth map and the confidence map are processed by two separate decoders to which the output of the bottleneck layers serves as an input.

4.1.4 TSDF Extraction

Instead of processing each ray of a view t independently as in standard TSDF fusion, we deliberately choose to compute the TSDF updates based on the information stored in a larger neighborhood in order to make a more informed decision about the surface location. Further, the 2D input data also holds valuable information about surface locations as often indicated by depth discontinuities. We argue that the fusion network can best benefit from both 2D and 3D data sources when they are already in correspondence and therefore propose a view-aligned local neighborhood extraction. Then, the 3D TSDF data and the 2D input data can be easily concatenated and fed into the network. Hence, for efficient real-time updates of the global data V_{t-1} , W_{t-1} , we extract a local, view-dependent TSDF volume and corresponding weights $V_{t-1}^*, W_{t-1}^* \in \mathbb{R}^{W \times H \times S}$. The first two volume dimensions W, H correspond to the width and height of the depth map whereas the third dimension S represents the local depth-sampling dimension of the window sampled along the ray. This number S closely relates to the truncation distance in standard TSDF fusion. For each ray independently, the local windows are centered at their respective depth values \hat{D}_t and discretely sampled into a fixed number of S values from the volume V_{t-1} . We choose the step size of the sampling according to the resolution of the scene and use trilinear interpolation to mitigate sampling artifacts.

The input I_t to the subsequent depth fusion is then a combination of all available local information. More specifically, this the corrected depth map \hat{D}_t , the confidence map C_t as well as the extracted TSDF values V_{t-1}^* and TSDF

weights \mathbf{W}_{t-1}^*

$$\mathbf{I}_t = \left[\hat{\mathbf{D}}_t \quad \mathbf{C}_t \quad \mathbf{W}_{t-1}^* \quad \mathbf{V}_{t-1}^* \right] \in \mathbb{R}^{W \times H \times (2S+2)}. \quad (4.3)$$

Before the subsequent update prediction step, we explicitly filter gross outliers where $\mathbf{C}_t < C_{\text{thr}}$ and set their corresponding feature values in \mathbf{I}_t to zero.

4.1.5 Depth Fusion

Our depth fusion network takes the local 3D feature volume \mathbf{I}_t as input and predicts the local TSDF update $v_t^* \in \mathbb{R}^{W \times H \times S}$. The architecture is fully convolutional in two dimensions and the channel dimension is along the camera viewing direction. Our network is compact and thereby facilitates real-time computation.

Our depth fusion network operates in a two-stage approach, as shown in Figure 4.3. The first stage encodes local and global information in the viewing frustum. We sequentially pass the input 3D feature volume through encoding blocks of two consecutive convolutional layers with interleaved batch normalization, non-linear activation using leaky ReLUs, and a dropout layer. The output of every block is concatenated with its input and passed through the next block. With every block, the receptive field of the neural network increases. This sequential feature extraction results in a 100-dimensional feature vector for each ray in the viewing frustum.

The second network part takes the feature volume and predicts the TSDF updates along each ray. The number of features is sequentially reduced by passing them through convolutional blocks with two 1×1 convolutional layers interleaved with leaky ReLUs, batch normalization, and dropout layers. In the last block, we directly reduce from 40 features to 20 in the first layer and then to S TSDF values in the last convolutional layer, where we apply a tanh-activation on the output mapping it to the range $[-1, 1]$.

Note that predicted TSDF update values v_t^* can take any value. The network can decide to not update the TSDF at all, *e.g.*, in case of an outlier. Conversely, it can reduce the influence of existing TSDF values if they contained outliers.

4.1.6 TSDF Update Integration

In order to compute the updated global TSDF volume V_t we transform the predicted local TSDF updates v_t^* back into the global coordinate frame v_t . To this end, we apply the inverse operation of the previous extraction step, that is, we redistribute the values using the same trilinear interpolation weights. In fact, we actually repurpose the update weights w_t for this task, where W_t accumulates the splatting weights for each voxel in the scene. Moreover, we also use W_t for post-filtering extreme outliers, which we discuss next.

4.1.7 Outlier Filtering

In order to reduce the amount of outliers in the scene, we have chosen to introduce outlier post-filtering according to the accumulated update weights during TSDF integration. Therefore, after every 100 frames integrated, we re-initialize all voxels, where the accumulated weights are smaller than 3. This is motivated by the fact that non-outlier geometry is frequently observed and only outliers rarely updated as they are almost randomly distributed in space.

4.1.8 Loss Function and Training Procedure

The two networks in our pipeline are trained in two steps. First, we train the depth routing network and then use the pre-trained routing output to train the fusion network.

Depth Routing Network. We train the depth prediction head in a supervised manner by computing the L1 loss on absolute depth values as well as on the depth map gradient, as proposed in [Donné and Geiger, 2019]. For training the confidence head, we chose a self-supervised approach [Kendall and Gal, 2017]. Therefore, the final loss function has the form

$$\mathcal{L}_{2D} = \sum_i c_i \mathcal{L}_1(y_i, \hat{y}_i) + c_i \mathcal{L}_1(\nabla y_i, \nabla \hat{y}_i) - \lambda \log c_i \quad (4.4)$$

where y_i, \hat{y}_i are the predicted and ground-truth depth values at pixel i respectively and $c_i \in C_t$ is the confidence value. The hyperparameter λ is empirically

set to 0.015.

Depth Fusion Network. Despite the pre-processing of the routing network, the filtered depth map might still contain noise and outliers which should be further handled by the depth fusion network. Each global TSDF update step should a) integrate new information about the true geometry and b) not destroy valuable, previously fused surface information. We train the fusion network in a supervised manner by randomly choosing an update step t during the fusion and penalize differences between the updated local volume $\mathbf{V}_t^* = \frac{\mathbf{W}_{t-1}^* \cdot \mathbf{V}_{t-1}^* + w_t^* \cdot v_t^*}{\mathbf{W}_{t-1}^* + w_t^*} \in \mathbb{R}^{W \times H \times S}$ and the local ground-truth $\hat{\mathbf{V}}_i^* \in \mathbb{R}^{W \times H \times S}$. Therefore, we define the loss function over all rays i as

$$\mathcal{L}_{3D} = \sum_i \lambda_1 \mathcal{L}_1(\mathbf{V}_{ti}^*, \hat{\mathbf{V}}_i^*) + \lambda_C D_C(\mathbf{V}_{ti}^*, \hat{\mathbf{V}}_i^*) \quad (4.5)$$

Here, \mathcal{L}_1 denotes the L1 loss over raw TSDF values and D_C denotes the cosine distance between the signs of the TSDF values computed along each ray i . The goal of the first term is to preserve fine surface details (through means of \mathcal{L}_1), while, the term D_C ensures that the surface is located at the zero-crossing of the signed distance field. The weights $\lambda_1 = 1$ and $\lambda_C = 0.1$ have been empirically found.

4.2 Experiments

In this section, we first present additional implementation details and our experimental setup. Next, we evaluate and discuss the efficacy of our approach on both synthetic and real-world data. We demonstrate that our approach outperforms traditional TSDF fusion and state-of-the-art learning-based approaches in terms of reconstruction accuracy with only little computational overhead.

4.2.1 Implementation Details

All networks were implemented in PyTorch and trained on an NVIDIA TITAN Xp GPU. We trained both networks using the RMSProp optimization algorithm with momentum 0.9 and initial learning rate $1e-5$ for the depth routing network

and $1e-3$ for the depth fusion network. The dropout layers were set to a probability of 0.2. For all experiments, we trained our neural networks in a sequential process, where we first pre-trained the depth routing and then the depth fusion network. A joint end-to-end refinement did not lead to an improvement of the overall performance of the system. To train the depth routing network, we use 10K frames sampled from 100 ModelNet [Zhirong Wu et al., 2015] or ShapeNet [Chang et al., 2015] objects and perturb them with artificial speckle noise. The data is packed into batches of size 4 and the gradient is accumulated across 8 batches before updating the routing network weights. Because of the incremental nature of the TSDF update equation, we must train our depth fusion network using a batch size of 1. However, each batch updates a very large number of voxels in the volume over which the loss is defined and, together with batch normalization, we obtain robust convergence during training. Since our network has only very few parameters, it is hard to overfit and only little training data is required. In fact, we can train our entire network (given a pre-trained depth routing network) on only ten models from ModelNet [Zhirong Wu et al., 2015] or ShapeNet [Chang et al., 2015] with a total of 1000 depth maps and it already generalizes robustly to other scenes. Furthermore, we can train the network from scratch in only 20 epochs (each epoch passes once over all 1000 frames). Unless otherwise specified, we used $S = 9$ and $C_{thr} = 0.9$ across all experiments. For all experiments, we used a voxel size $0.008m$, corresponding to a grid resolution of 128^3 for ShapeNet and ModelNet.

Runtime. A forward pass through the depth routing network and the depth fusion network for one depth map ($W = 320, H = 240$) takes 0.9 ms and 1.8 ms, respectively while the full pipeline runs at 15 fps. These numbers can be improved with a more efficient implementation, but already meet real-time requirements.

4.2.2 Results

We evaluate our method on synthetic and real-world data comparing to traditional TSDF fusion [Curless and Levoy, 1996a] as a baseline as well as to the state-of-the-art PSDF fusion method presented by Dong *et al.* [Dong et al., 2018]. Moreover, we compare to state-of-the-art learning-based 3D reconstruction

methods OccupancyNetworks [Mescheder et al., 2019] and DeepSDF [Park et al., 2019].

Evaluation Metrics. For quantifying the performance of our method, we compute the following four metrics by comparing the estimated TSDF against the ground-truth.

- **MAD:** The mean absolute distance is computed over all TSDF voxels and measures the reconstruction performance on fine surface details.
- **MSE:** The mean squared error loss is computed over all TSDF voxels and measures the reconstruction performance on large surface deviations.
- **Accuracy:** We compare the actual reconstruction accuracy on the occupancy grid. We extract the occupancy grid in the ground-truth and the estimated TSDF by extracting all voxels with negative TSDF values.
- **Intersection over Union (IoU):** We compute the intersection-over-union on the occupancy grid, which is an alternative performance measure to the accuracy.

These metrics not only quantify how well our pipeline fuses depth maps into a TSDF, but also how well it performs in classifying the occupancy and reconstructing the geometry.

4.2.3 Synthetic Data

To evaluate our method’s performance in fusing noisy synthetic data, we train and test it on the ModelNet [Zhirong Wu et al., 2015] and ShapeNet [Chang et al., 2015] datasets using rendered ground-truth depth maps that are perturbed with an artificial depth-dependent multiplicative noise distribution. For both, ModelNet and ShapeNet, we randomly sample our training and test data from the official train-test split.

ShapeNet. The model trained on ShapeNet is then used to evaluate the performance of our method in comparison with other approaches. Therefore, we fuse noisy depth maps of 60 objects (10 per test class - plane, sofa, lamp, table, car, chair) from the test set, which have not been seen during training. For comparison, we use the provided pre-trained model for point cloud completion in the case of

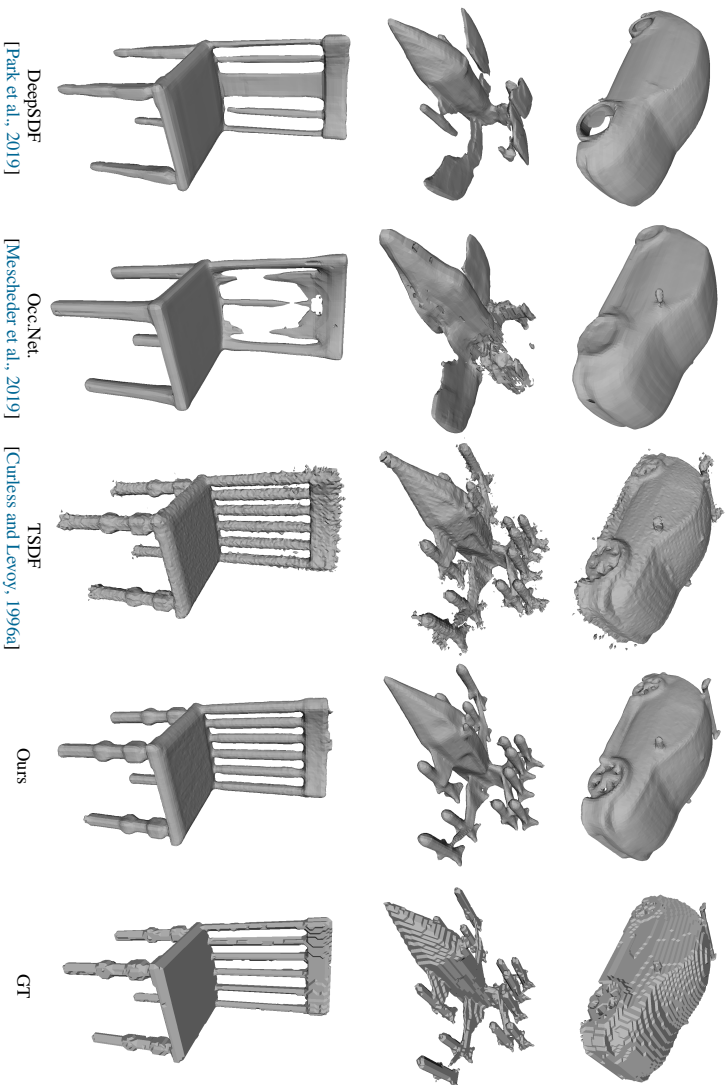


Figure 4.4: Qualitative Results on ShapeNet [Chang et al., 2015]. Our method is superior to all other methods in reconstructing fine details (see car wheels and spoiler) and produces smoother surfaces (input noise level $\sigma = 0.005$).

Method	MSE [e-05]	MAD	Acc. [%]	IoU [0, 1]
DeepSDF [Park et al., 2019]	464.0	0.0499	66.48	0.538
OccupancyNetworks [Mescheder et al., 2019]	56.8	0.0166	85.66	0.484
TSDF Fusion [Curless and Levoy, 1996a]	11.0	0.0078	88.06	0.659
TSDF Fusion + Routing	27.0	0.0084	87.48	0.650
Ours w/o Routing	5.9	0.0051	93.91	0.765
Ours	5.9	0.0050	94.77	0.785

Table 4.1: Quantitative Results on ShapeNet [Chang et al., 2015]. Our method outperforms TSDF fusion and other learning-based approaches in fusing noisy ($\sigma = 0.005$) depth-maps rendered from ShapeNet objects. The benefit of the routing network increases with higher noise levels (see Fig. 4.5).

OccupancyNetworks. In the case of DeepSDF, we trained the model from scratch using the code provided by the authors and using ShapeNet as training data. The quantitative results of this evaluation are shown in Table 4.1. Our method consistently outperforms standard TSDF fusion as well as the pure learning-based approaches OccupancyNetworks [Mescheder et al., 2019] and DeepSDF [Park et al., 2019] on all metrics. Our method significantly improves the accuracy of the fused implicit mesh as well as their IoU, MAD and MSE scores. The results also indicate the potential of our routing network. However, the full benefit of our routing network only becomes obvious when looking at the real-world data experiments and Figure 4.5.

Figure 4.4 illustrates the strengths of our method in dealing with noise and in reconstructing thin structures. Flat surfaces in the ground-truth appear smoother in our results as compared to standard TSDF fusion. Furthermore, thin structures are better reconstructed and contain less thickening artifacts. The thickening artifacts are also visible on the rim of the car, where our method yields accurate results and DeepSDF and OccupancyNetworks both fail. Both DeepSDF and OccupancyNetworks tend to over-smooth surface details less common in the training data, *e.g.*, the spoiler of the car or the details on the legs of the chair.

ModelNet. In order to test the robustness of our method against noise we trained and evaluated it on various noise levels $\sigma \in \{0.01, 0.03, 0.05\}$ and compared it to standard TSDF fusion. We also analyze the effect of the depth routing network on the fusion result by omitting it in our pipeline and by testing it in combination with standard TSDF fusion. Figure 4.5 illustrates that our pipeline outperforms

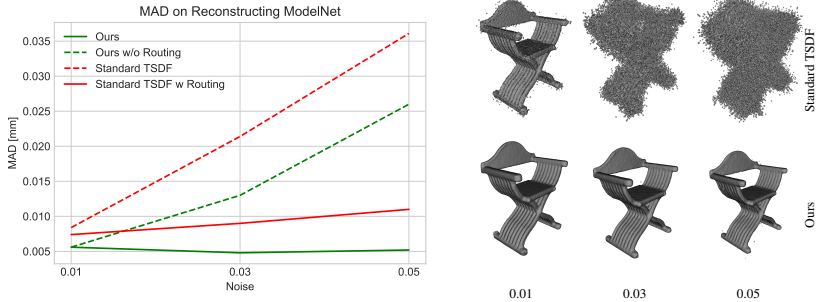


Figure 4.5: Evaluation of different noise levels σ . The left plot shows MAD for different noise levels $\sigma \in \{0.01, 0.03, 0.05\}$. Our routing network stabilizes both, our method as well as standard TSDF fusion, for high noise levels. On the right, we show corresponding qualitative results on ModelNet test data for Standard TSDF and our method. The figures show the denoising capability of our method, where standard TSDF fusion completely fails.

standard TSDF fusion for all tested noise levels. It also shows that our depth routing network stabilizes the fusion of data corrupted with extreme noise levels. When used for data pre-processing, our depth routing network also improves the results of standard TSDF fusion.

4.2.4 Real-World Data

We also evaluate on real-world datasets and compare to other state-of-the-art fusion methods. Due to lack of ground-truth data, we use the model trained on synthetic ModelNet data using an artificial and empirically chosen depth-dependent noise distribution with $\sigma = 0.01$. As such, we also show that our method must not necessarily be trained on real-world data but generalizes robustly to the real domain from being trained on noisy synthetic data only.

3D Scene Data [Zhou and Koltun, 2013]. To quantify the improvement of the reconstruction result, we evaluate our method compared to standard TSDF fusion on scenes provided by Zhou *et al.* [Zhou and Koltun, 2013]. Since there is no volumetric ground-truth available for these scenes, we fuse all frames of each scene using standard TSDF fusion and denoised the meshes. Then, we only fuse every 10th frame using standard TSDF fusion as well as our method for

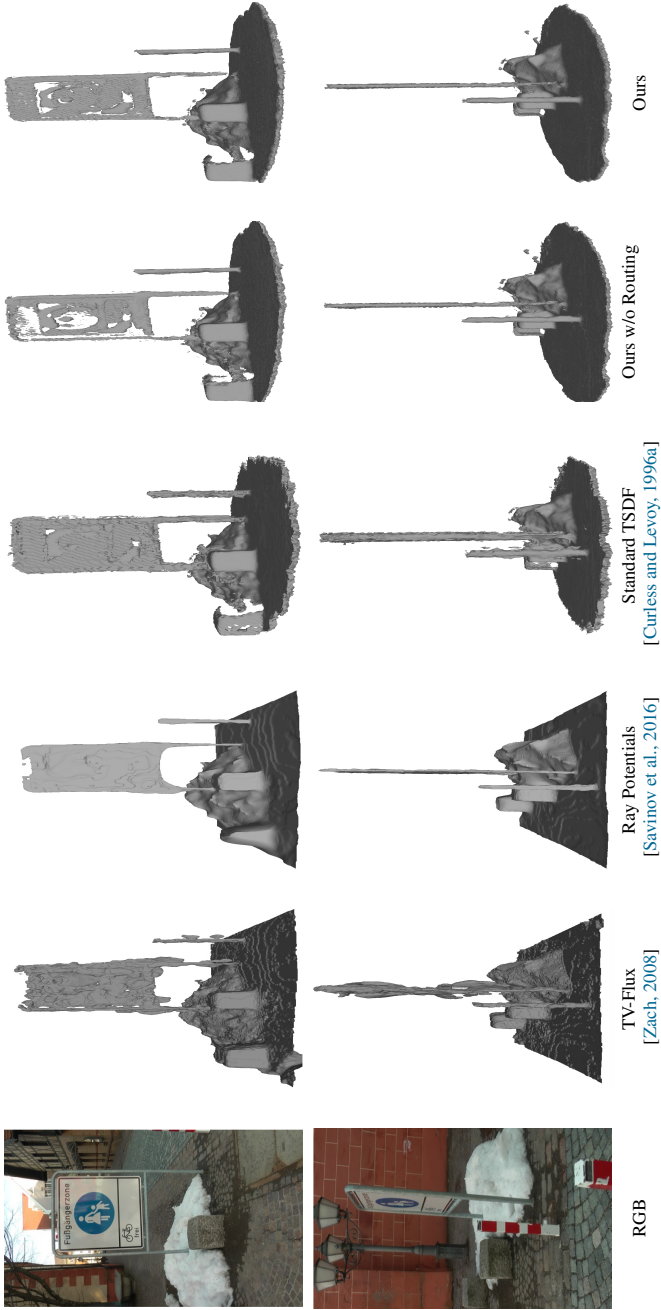


Figure 4.6: Qualitative results of our method on the Roadsign dataset [Ummenhofer and Brox, 2013]. Our method compares favorably to standard TSDF fusion as well as TV-Flux [Zach, 2008] in reconstructing thin surfaces while showing comparable performance with ray potentials [Savinov et al., 2016]. Our method generalizes reasonably well since it was trained on ModelNet and never saw the noise and outlier statistics of stereo depth maps nor the shape statistics of this scene and therefore has a less complete output. ($C_{thr} = 0.5$)

Method	Lounge	Copyroom	Stonewall	Cactusgarden	Burghers
TSDF	0.0095	0.0110	0.0117	0.0104	0.0126
RoutedFusion w/o routing	0.0055	0.0057	0.0047	0.0055	0.0071
RoutedFusion	0.0051	0.0051	0.0043	0.0052	0.0067

Table 4.2: Quantitative evaluation (MAD [mm]) of our method on 3D Scene Data [Zhou and Koltun, 2013]. Our method is consistently better than standard TSDF fusion on 3D Scene Data. These experiment also shows the benefit of our routing network when applied to real-world data.

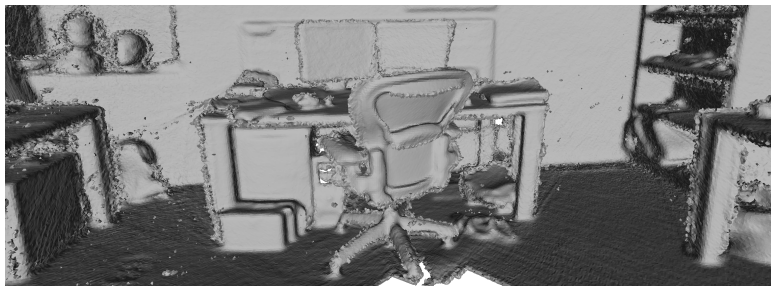
evaluation.

Table 4.2 shows the quantitative reconstruction results from fusing 5 scenes of the 3D scene dataset [Zhou and Koltun, 2013]. Our method significantly outperforms standard TSDF fusion on all scenes without being trained on real-world data.

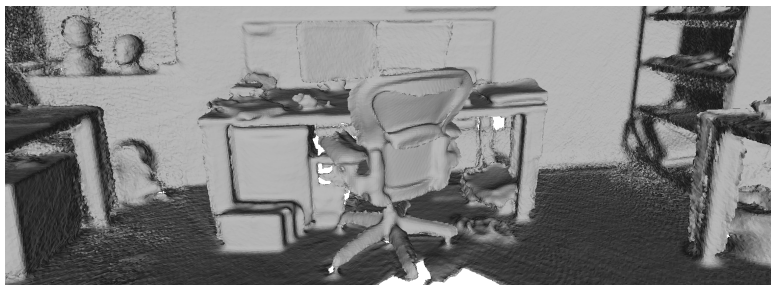
We further show a qualitative comparison to standard TSDF as well as PSDF fusion [Dong et al., 2018] on the Burghers of Calais scene in Figure 4.8. The results illustrate that our method better reconstructs fine geometric details (hands, fingers and face) and produces smoother surfaces than standard TSDF fusion and PSDF fusion [Dong et al., 2018].

Street Sign Dataset [Ummenhofer and Brox, 2013]. To evaluate the performance of our method on thin structures, we also evaluate on the street sign dataset, again without fine-tuning the network. This dataset consists of 50 RGB frames and we use the COLMAP SfM pipeline [Schönberger and Frahm, 2016, Schönberger et al., 2016] to compute camera poses and depth maps. Qualitative results on this scene for different state-of-the-art methods are shown in Figure 4.6. Our method clearly outperforms TV-Flux [Zach, 2008] and standard TSDF, while producing comparable results with ray potentials [Savinov et al., 2016]. The results also make the benefit of our routing network apparent. With routing, our method reconstructs with better completeness and less noise artifacts than without. Note that both TV-Flux and ray potentials involve an offline optimization with a smoothness prior to reduce noise and complete missing data. This prevents real-time application for these approaches, since ray potentials on this small scene runs for many hours on a cluster.

RGB-D Dataset 7-Scenes [Shotton et al., 2013]. For qualitatively evaluating our method on Kinect data, we fuse the 7-Scenes [Shotton et al., 2013] RGB-D

Standard TSDF
[Curless and Levoy, 1996a]

Ours w/o Routing



Ours

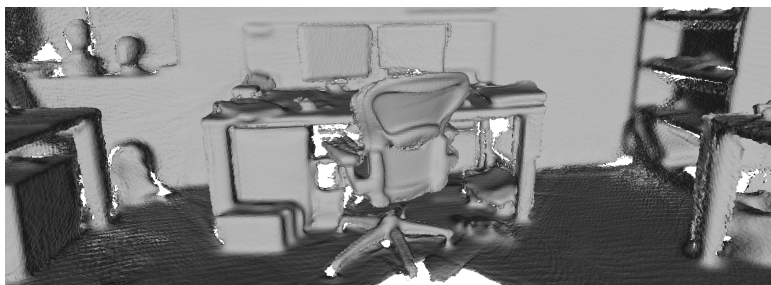


Figure 4.7: Qualitative comparison on the heads scene of RGB-D Dataset 7-Scenes [Shotton et al., 2013]. Our method significantly reduces noise artifacts and thickening effects - especially on the thin geometry of the chair’s leg.

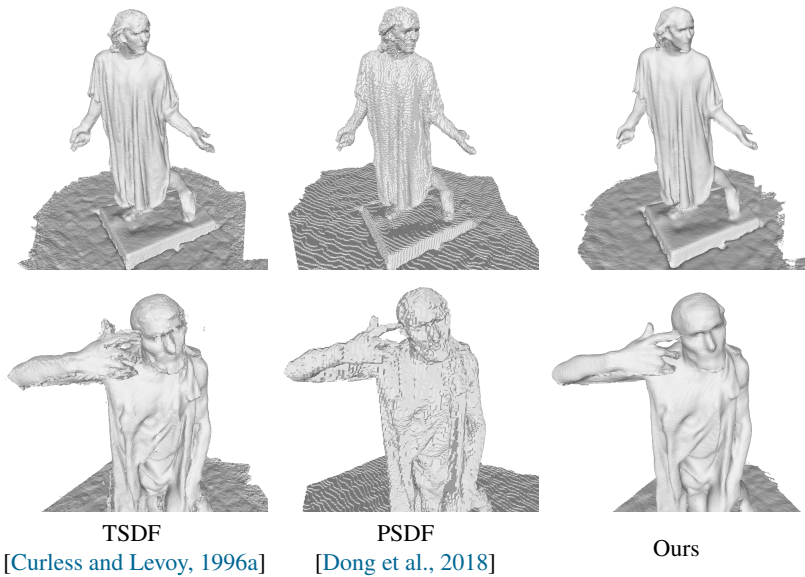


Figure 4.8: Qualitative comparison on the Burghers of Calais scene [Zhou and Koltun, 2013]. Our method reconstructs hands and face geometry with much higher degree of detail than standard TSDf fusion and PSDF fusion.

dataset. For each scene, we have chosen the first trajectory and fused it using our pipeline as well as standard TSDF fusion. In Figure 4.7, we show that our method significantly reduces noise and mitigates the surface thickening effect compared to standard TSDF fusion. Notably, the chair leg and table edges are reconstructed with higher fidelity than it is done by standard TSDF fusion. Moreover, our method shows strong performance in denoising and removing outliers from the scene.

4.2.5 Ablation Studies

Number of Samples S . First, we discuss our choice for the number of samples S . With figure 4.9, we show that sampling 9 values inside the local window centered around the surface leads to the best reconstruction performance. The number of samples S is closely related to the truncation distance in standard TSDF fusion [Curless and Levoy, 1996a]. Since the spacing between samples in the window is fixed to the scene’s resolution, the size of the local window is dependent on S . Therefore, an increase in S leads to an increase of the local window size. By increasing S and the window size, we feed more information along the ray to the depth fusion network and we can account for larger noise levels. However, if we increase the number of samples beyond 9, the performance decreases again, which is experimentally shown in figure 4.9. Having empirically evaluated the influence of S on our depth map fusion pipeline, we decided to keep the number of samples $S = 9$ constant across all experiments.

4.3 Discussion

4.3.1 Limitations

While RoutedFusion mitigates the shortcomings of standard TSDF fusion regarding thickening artifacts and surface vanishing due to its learned update function, it still lacks adequate outlier handling. As the fusion process is performed in an explicit TSDF grid, the network struggles from differentiating between observing a new surface and an outlier since their local signal can be similar. To obtain a

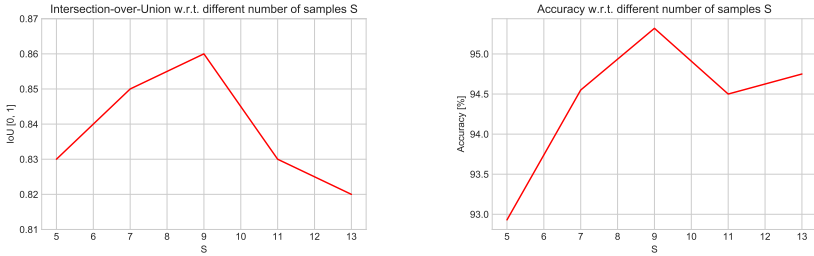


Figure 4.9: Intersection over Union on Modelnet [Zhirong Wu et al., 2015] test data for different numbers of samples S . When sampling 9 SDF values inside the local window, our pipeline shows the best performance in reconstructing models from noisy depth measurements.

clean and outlier-free reconstruction, the measurements have to be pre-processed by aggressively filtering low-confidence depth values or the reconstruction has to be post-processed by removing all geometry with a low observation count. Both strategies lead to a decrease in the completeness as these hand-crafted heuristics do not discriminate between outliers and actual geometry as both the confidence and the observation count are only proxies for the outlier decision.

Further, our results show that the RoutedFusion reconstructs fine geometry with higher accuracy than existing baselines. Yet, in the presence of severe noise - *e.g.* far away from the sensor - the reconstruction might become oversmoothed. This can be alleviated by filtering far away measurements in the input. However, this comes again at the cost of reduced completeness that manifests itself by holes and missing geometry in the reconstruction.

The main driver for these two limitations is that we train the pipeline on synthetic data with simulated sensor noise due to lack of high-quality real ground-truth data. This is of course only an approximation of the real conditions. While this works well for standard settings, it covers not all test conditions. In order to solve this problem, we would need to train the pipeline on real-world data. However, this is challenging for the following reason. There are no RGB-D datasets with high-quality ground-truth. Most datasets, *e.g.* Scannet [Dai et al., 2017a], create a "ground-truth" 3D geometry by fusing the RGB-D using standard TSDF fusion. This leads to the problem that this ground-truth is biased towards

the shortcomings of standard TSDF fusion and training a neural network of depth map fusion on this data would introduce these biases into the network itself. Thus, this limitation has to be overcome by either creating more accurate groundtruth for real-world data or designing a supervision signal that does not require 3D groundtruth.

4.3.2 Summary

We have presented a novel real-time capable depth map fusion method tackling the common limitations of standard TSDF fusion [Curless and Levoy, 1996a]. Due to learned non-linear TSDF updates – rather than hand-crafted linear updates – our method mitigates inconsistent reconstruction results that occur at object edges and thin structures. The proposed split of our network architecture into a 2D depth routing network and a 3D depth fusion network allows to effectively handle noise and outliers at different processing stages. Moreover, sensor-specific noise distributions can be learned from small amounts of training data. Our approach outperforms competing methods on both synthetic and real data experiments. Due to its low computational requirements and compact architecture, our method has the potential to replace standard TSDF fusion in a variety of tasks and applications.

Moving the fusion to a latent space

While we have addressed the challenge of online depth map fusion from a machine learning perspective in the previous chapter, we move it one step further in this chapter. The previously presented method, RoutedFusion, uses neural networks for depth map fusion. Yet, it represents the scene as a traditional signed distance field stored in a 3D voxel-grid. While this scene representations has the advantage of being explicitly interpretable and the underlying geometry can be easily extracted, this explicitness is a disadvantage when it comes to outlier handling. As discussed in the limitations section of chapter 4, existing methods, *e.g.* TSDF Fusion [Curless and Levoy, 1996a] and RoutedFusion, use various heuristics to filter outliers in decoupled pre- or post-processing steps to tackle this fundamental limitation. Such filtering techniques entail the usual trade-off in terms of balancing accuracy against completeness. Especially in an online fusion system, striking this balance is extremely challenging in the pre-filtering stage, because it is difficult to distinguish between a first surface measurement and an outlier. Consequently, to achieve complete surface reconstructions, one must use conservative pre-filtering, which in turn requires careful post-filtering of outliers by non-local reasoning on the TSDF volume or the final mesh.

In this chapter, we explore the potential of a learned scene representation for online depth fusion in order to tackle the challenge of outlier handling. Therefore,

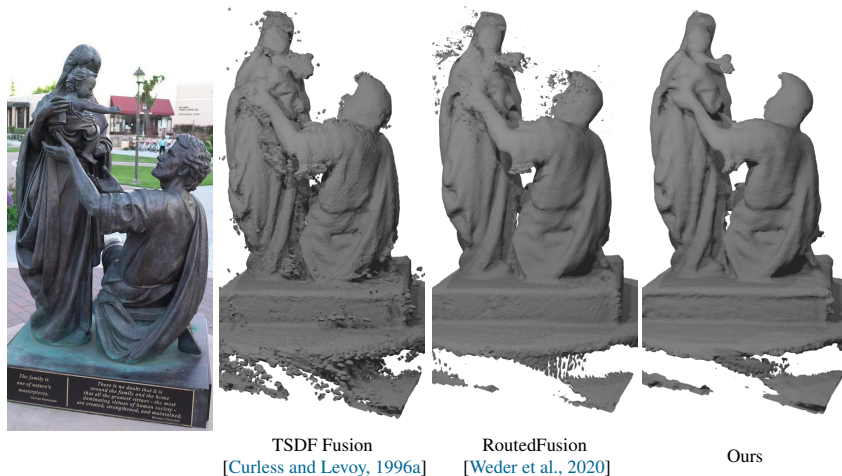


Figure 5.1: Results of our end-to-end depth fusion on real-world MVS data [Knapitsch et al., 2017]. Our method learns to separate outliers and true geometry without the need of filtering heuristics.

we represent - in contrast to RoutedFusion - the scene by learned features that are stored in a voxel grid similar to [Sitzmann et al., 2019a]. This allows to move the decision whether a certain measurement is an outlier or new geometry to a later stage when translating the learned feature representation into an interpretable representation of geometry. This allows to fuse all available information to improve completeness and by learning to decide what measurements are beneficial and what are not to have a clean reconstruction. In contrast to previous methods, we perform the fusion step in this learned scene representation that implicitly learned to encode features like confidence information or local scene information. A final translation step simultaneously filters and decodes this learned scene representation into the final output relevant to downstream applications.

In summary, we make the following contributions in this chapter:

- We propose a novel, end-to-end trainable network architecture, called NeuralFusion, which separates the scene representations for depth fusion and the final output into two different modules.
- The proposed latent representation yields more accurate and complete fusion re-

sults for larger feature dimensions allowing to balance accuracy against resource demands.

- Our network architecture allows for end-to-end learnable outlier filtering within a translation step that significantly improves outlier handling.
- Although fully trainable, our approach still only performs very localized updates to the global map which maintains the online capability of the overall approach.

5.1 Method

5.1.1 Overview

Given a stream of input depth maps $D^t : \mathbb{R}^2 \rightarrow \mathbb{R}$ with known camera calibration for each time step $t \in \mathbb{N}$, we aim to fuse all surface information into a globally consistent scene representation $g : \mathbb{R}^3 \rightarrow \mathbb{R}^N$ while removing noise and outliers as well as complete potentially missing observations. The final output of our method is a TSDF map $s : \mathbb{R}^3 \rightarrow \mathbb{R}$, which can be processed into a mesh with standard iso-surface extraction methods as well as an occupancy map $o : \mathbb{R}^3 \rightarrow [0, 1]$. Figure 5.2 provides an overview of our method. The key idea is to decouple the scene representation for geometry fusion from the output scene representation. This decoupling is motivated by the difficulty for existing methods to handle outliers within an online fusion method. Therefore, we propose to fuse geometric information into a latent feature space without any preliminary outlier pre-filtering. A subsequent translator network then decodes the latent feature space into the output scene representation (*e.g.*, a TSDF grid). This approach allows for better and end-to-end trainable handling of outliers and avoids any handcrafted post-filtering, which is inherently difficult to tune and typically decreases the completeness of the reconstruction. Furthermore, the learned latent representation also enables to capture complex and higher resolution shape information, leading to more accurate reconstruction results.

Our feature fusion pipeline consists of four key stages depicted as networks in Figure 5.2. The first stage extracts the current state of the global feature volume into a local, view-aligned feature volume using an affine mapping defined by

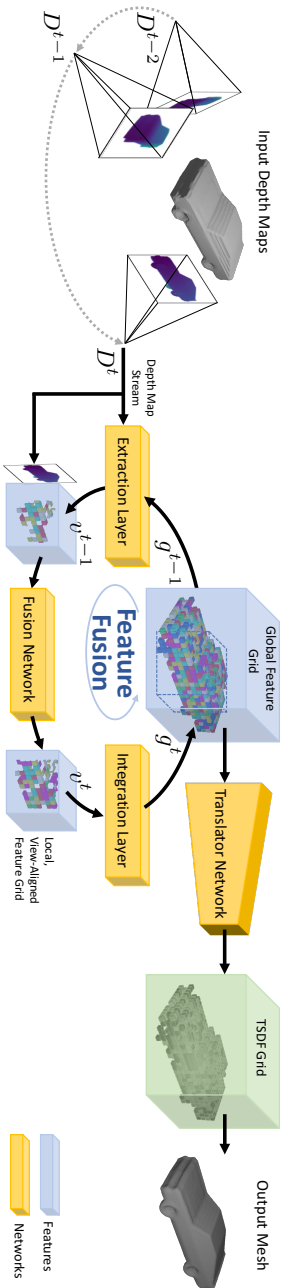


Figure 5.2: Proposed online reconstruction approach. Our pipeline consists of two main parts: **1)** A fusion network with its extraction and integration layers, and **2)** A translator network that translates the feature representation into an interpretable TSDF representation. For any new depth map D^t a local, view-aligned feature grid v^{t-1} is extracted from the previous global feature grid g^{t-1} . The fusion network updates the local feature grid v^t which is then integrated back into an updated global feature grid g^t . The translator network is independent of the fusion process and can be used asynchronously for an efficient fusion process.

the given camera parameters. After the extraction, this local feature volume is passed together with the new depth measurement and the ray directions through a feature fusion network. This feature fusion network predicts optimal updates for the local feature volume, given the new measurement and its old state. The updates are integrated back into the global feature volume using the inverse affine mapping defined in the first stage. These three stages form the core of the fusion pipeline and are executed iteratively on the input depth map stream. An additional fourth stage translates the feature volume into an application-specific scene representation, such as a TSDF volume, from which one can finally render a mesh for visualization. We detail our pipeline in the following.

5.1.2 Feature Extraction

The goal of iteratively fusing depth measurements is to **(a)** fuse information about previously unknown geometry, **(b)** increase the confidence about already fused geometry, and **(c)** to correct wrong or erroneous entries in the scene. Towards these goals, the fusion process takes the new measurements to update the previous scene state g^{t-1} , which encodes all previously seen geometry. For a fast depth integration, we extract a local view-aligned feature subvolume v^{t-1} with one ray per depth measurement centered at the measured depth via nearest neighbor search in the grid positions. Each ray of features in the local feature volume is concatenated with the ray direction and the new depth measurement. This feature volume is then passed to the fusion network.

5.1.3 Feature Fusion

The fusion network fuses the new depth measurements D^t into the existing local feature representation v^{t-1} . Therefore, we pass the feature volume through four convolutional blocks to encode neighborhood information from a larger receptive field. Each of these encoding blocks, consists of two convolutional layers with a kernel size of three. These layers are followed by layer normalization and tanh activation function. We found layer normalization to be crucial for training convergence. The output of each block is concatenated with its input, thereby generating a successively larger feature volume with increasing receptive field.

The decoder then takes the output of the four encoding blocks to predict feature updates. The decoder consists of four blocks with two convolutional layers and interleaved layer normalization and tanh activation. The output of the final layer is passed through a single linear layer. Finally, the predicted feature updates are normalized and passed as v^t to the feature integration.

5.1.4 Feature Integration

The updated feature state is integrated back into the global feature grid by using the inverse global-local grid correspondences of the extraction mapping. Similar to the extraction, we write the mapped features into the nearest neighbor grid location. Since this mapping is not unique, we aggregate colliding updates using an average pooling operation. Finally, the pooled features are combined with old ones using a per-voxel running average operation, where we use the update counts as weights. This residual update operation ensures stable training and a homogeneous latent space, as compared to direct prediction of the global features. Both the feature extraction and integration steps are inspired by RoutedFusion introduced in the previous chapter, but they use tri-linear interpolation instead of nearest-neighbor sampling. When extracting and integrating features instead of SDF values, we empirically found that nearest-neighbor interpolation produces better results and leads to more stable convergence during training.

5.1.5 Feature Translation

In the final and possibly asynchronous step, we translate the latent scene representation g^t into a representation usable for visualization of the scene (*e.g.*, signed distance field or occupancy grid). The network architecture in this step is inspired by IM-Net [Chen and Zhang, 2019]. For efficient and complete translation, we sample a regular grid of world coordinates. Then, at each of these sampled points p_i , the translator aggregates the information stored in the features of the local neighborhood and predicts the TSDF $s(p_i)$ as well as occupancy $o(p_i)$ for this specific grid location. To this end, the translator concatenates the feature vectors of the $5 \times 5 \times 5$ neighborhood and compresses them into a single feature vector using a linear layer followed by tanh activation. Next, the so combined

features are concatenated with the query point feature $g^t(p_i)$ and passed through the remaining translation network, which consists of four linear layers interleaved with tanh activations and channel-wise dropout preventing the network from overfitting to a single feature channel. According to the desired output ranges, the TSDF head is activated using tanh, while the occupancy head uses a sigmoid activation. After each layer, we concatenate the output with the original query point feature $g^t(p_i)$.

5.1.6 Training Procedure and Loss Function.

All networks are jointly trained end-to-end. In each training epoch, we randomly shuffle the input depth maps and iteratively fuse them one by one into the corresponding latent feature grid $g : \mathbb{R}^3 \rightarrow \mathbb{R}^N$. After integrating the depth map into the latent feature grid, we query the translator network, where the latent feature grid was just updated, and render the TSDF $s : \mathbb{R}^3 \rightarrow \mathbb{R}$ and occupancy $o : \mathbb{R}^3 \rightarrow [0, 1]$. The entire pipeline is optimized using the following loss function:

$$\begin{aligned} \mathcal{L} = & \frac{1}{n} \sum_i \lambda_1 \mathcal{L}_1(s_i, \hat{s}_i) + \lambda_2 \mathcal{L}_2(s_i, \hat{s}_i) \\ & + \lambda_o \mathcal{L}_o(o_i, \hat{o}_i) + \lambda_g \overline{\sigma_{ch}^2(g)} \ , \end{aligned} \quad (5.1)$$

where \mathcal{L}_1 and \mathcal{L}_2 denote the L_1 and L_2 norms, and \mathcal{L}_o is the binary cross-entropy on the predicted occupancy. The \mathcal{L}_2 loss is helpful with outliers, whereas the \mathcal{L}_1 loss improves the reconstruction of fine details. In each step, n denotes the number of all updated feature grid locations. When training with outlier contaminated data, we found that setting n equal to all visited feature grid locations yields the best results. Therefore, n is a crucial hyperparameter when training the pipeline. Moreover, \hat{s}_i and \hat{o}_i denote the ground-truth TSDF and occupancy value, respectively. To avoid large deviations for a single feature in the latent space, we regularize the feature grid g by penalizing the mean of the channel-wise variance by $\overline{\sigma_{ch}^2(g)}$. We empirically set the loss weights to $\lambda_1 = 1.$, $\lambda_2 = 10.$, $\lambda_o = 0.01$, and $\lambda_g = 0.05$.

5.2 Experiments

We first discuss implementation details and evaluation metrics before evaluating our method on synthetic and real-world data in comparison to other methods. We further analyze our method for varying numbers of features N in an ablation study.

5.2.1 Implementation Details.

Our pipeline is implemented in PyTorch and trained on an NVIDIA RTX 2080. All networks were trained using the Adam optimizer [Kingma and Ba, 2015] with an initial learning rate of 0.01, which was adapted using an exponential learning rate scheduler at a rate of 0.998. For momentum and beta, we empirically found the default parameters to yield the best results. We trained all networks on synthetic data being augmented with artificial noise and outliers. The batch-size is set to one due to the nature of the sequential fusion process. However, we accumulate the gradients across 8 scene update steps and then update the network parameters. Our un-optimized implementation runs at ~ 7 frames per seconds with a depth map resolution of 240×320 on an NVIDIA RTX 2080. This demonstrates the real-time applicability of our approach.

5.2.2 Evaluation Metrics.

We use the following evaluation metrics to quantify the performance of our approach: Mean Squared Error (**MSE**), Mean Absolute Distance (**MAD**), Accuracy (**Acc.**), Intersection-over-Union (**IoU**), Mesh Completeness (**M.C.**) Mesh Accuracy (**M.A.**), and **F1** score.

Mean Squared Error (MSE) and Mean Absolute Distance (MAD). The mean squared error measures the reconstruction error on the TSDF field by penalizing large surface deviations and outliers. The mean absolute distance is also computed on the TSDF grid. However, it mainly quantifies the performance on reconstructing fine geometric details.

Accuracy (Acc.), F1 Score, Intersection-over-Union (IoU). The accuracy is computed over the occupancy obtained from the sign of the TSDF grid. We

also report the F1 score, which is the harmonic mean of precision and recall. By measuring both, completeness and accuracy, it is a more holistic metric for quantifying the performance of a reconstruction method. Moreover, we measure the IoU on the occupancy grid. The IoU especially quantifies artifacts typically encountered in reconstructions from noisy depth maps, such as surface and corner thickening and the vanishing of fine geometric details.

Mesh Completeness (M.C.) and Accuracy (M.A.). We compute the completeness using the evaluation pipeline from [Stutz and Geiger, 2018]. The completeness describes the distance from points sampled on the ground-truth mesh to the closest point on the reconstructed mesh. Vice-versa, the accuracy computes the distance from points sampled on the reconstructed mesh to the closest point on the ground-truth mesh.

5.2.3 Results on Synthetic Data

Datasets. We used the synthetic ShapeNet [Chang et al., 2015] and ModelNet [Wu et al., 2015] datasets for performance evaluation. From ShapeNet, we selected 13 classes for training and evaluate on the same test set as RoutedFusion consisting of 60 objects from six classes, for which pretrained models [Park et al., 2019] are available. For ModelNet [Wu et al., 2015], we trained and tested on 10 classes using the train-test split from RoutedFusion. We first generated watertight models using the mesh-fusion pipeline used in [Mescheder et al., 2019] and computed TSDFs using the mesh-to-sdf¹ library. Additionally, we render depth frames for 100 randomly sampled camera views for each mesh. These depth maps are the input to our pipeline and existing methods. For both datasets, we found that training on one single object per class is sufficient for generalization to any other object and class.

Comparison to Existing Methods. For performance comparisons, we fuse depth maps and augment them with artificial depth-dependent noise as in [Riegler et al., 2017a]. We compare to state-of-the-art learned scene representation methods DeepSDF [Park et al., 2019], OccupancyNetworks [Mescheder et al., 2019], and IF-Net [Chibane et al., 2020], as well as to the online fusion methods TSDF

¹https://github.com/marian42/mesh_to_sdf

Fusion [Curless and Levoy, 1996a] and RoutedFusion. We further implemented two additional baselines to demonstrate the benefits of a fully learned scene representation for depth map fusion: (1) one baseline performs a learned 2D noise filtering before fusing the frames using TSDF Fusion [Curless and Levoy, 1996a], and (2) a baseline that post-processes models fused by TSDF Fusion using a simplification of our translation network - the principle is similar to OctnetFusion [Riegler et al., 2017a], but on a dense grid.

We compare all baselines on the test set of RoutedFusion in Figure 5.5. For input data augmentation, we used the same scale 0.005 as in RoutedFusion. Figure 5.5 shows that our method significantly outperforms all existing depth map fusion as well as learned scene representations. We especially emphasize the increase in IoU by more than 10%. This significant increase is due to many fine-grained improvements, where RoutedFusion wrongly predicts the sign, as shown in Figure 5.6. In all experiments, we set the truncation distance of TSDF Fusion to 4cm, which is similar to the receptive field of our fusion network.

Higher Input Noise Levels. We also assess our method in fusing depth maps corrupted with higher noise levels on the ModelNet dataset [Wu et al., 2015] in Figure 5.7. For this experiment, we augment the input depth maps with three different noise levels. We fuse the corrupted depth maps using standard TSDF Fusion [Curless and Levoy, 1996a] and RoutedFusion. Since we showed in RoutedFusion that the proposed routing network significantly improved the robustness to higher input noise levels, we also tested our method with depth maps pre-processed by a routing network. For these experiments, we use the pre-trained routing network we trained for RoutedFusion.

Outlier Handling. As discussed, one of the main drawback of RoutedFusion is its limitation in handling outliers. To this end, we run an experiment, where we augment the input depth maps with random outlier blobs. We create this data by sampling an outlier map from a fixed distribution scaled by a fixed outlier scale. Additionally, we sample three masks with a given probability (outlier fraction) and dilate it once, twice, and three times, respectively. Then, these masks are used to select the outliers from the outlier map. We report the results of this experiment in Figure 5.8. Note that the results might be better with even higher

Table 5.1: Ablation Study. We assess our method for different numbers of feature dimensions N . The performance saturates around $N = 8$. Note that $N = 1$ did not converge.

N	MSE↓ [e-05]	MAD↓ [e-02]	Acc.↑ [%]	IoU↑ [0,1]
1	-	-	-	-
2	9.45	0.64	94.67	0.717
4	4.03	0.30	97.51	0.863
8	3.99	0.29	97.46	0.862
16	3.91	0.29	97.50	0.863

outlier fractions since we only evaluate on updated grid locations. The consistency in outlier filtering and increase in updated grid locations improves the metrics.

5.2.4 Ablation Study

In a series of ablation studies, we discuss several benefits of our pipeline and justify design choices.

Iterative Fusion. Ideally, fusion algorithms should be independent from the number of integrated frames and steadily improve the reconstruction as new information becomes available. In Figure 5.9, we show that our method is not only better than competing algorithms from the start, but also continuously improves the reconstruction as more data is fused. The metrics are averaged at every fusion step over all scenes in the test set used for all experiments on ShapeNet [Chang et al., 2015].

Frame Order Permutation. Our method does not leverage any temporal information from the camera trajectory apart from the previous fusion result. This design choice allows to apply the method also to a broader class of scenarios (*e.g.* Multi-View Stereo). Ideally, an online fusion method should be invariant to permutations of the fusion frame order. To verify this property, we evaluated the performance of our method in fusing the same set of frames in ten different random frame orders. Figure 5.10 shows that our method converges to the same result for any frame order and thus seems to be invariant to frame order permutations.

Feature Dimension. An important hyperparameter of our method is the feature dimension N . Therefore, we show quantitative results for the reconstruction from noisy and outlier contaminated depth maps using varying N in Table 5.1.

We observe that a larger N clearly improves the results, but the performance eventually saturates, which justifies our default choice of $N = 8$ features.

Latent Space Visualization. In order to verify our hypothesis that the translator network mostly filters outliers, since the fusion network can hardly distinguish between first entries and outliers, we visualize the fused latent space in Figure 5.11. While the translated end result is outlier free, the latent space clearly shows that the fusion network keeps track of most measurements.

Generalization from a Single Object. In order to demonstrate the compactness and generalization performance of our network, we train it only on a single chair object from the ModelNet [Wu et al., 2015] dataset. We augment the input depth maps with artificial noise of scale 0.01. We report the results in Table 5.2 and show that our method trained on a single object achieves almost the same performance as our method trained on the full training set. Moreover, it outperforms the currently best performing method - RoutedFusion - that is trained on the full training set. This result indicates the applicability of our method to many real-world scenarios,

Table 5.2: Our method trained on the standard training split and on a single chair object only. The model trained on a single object is almost on par with our model trained on the full training set and outperforms the next best existing method trained on the full training set.

Method	MSE↓ [e-05]	MAD↓ [e-02]	Acc.↑ [%]	IoU↑ [0,1]
RoutedFusion (full training set)	6.79	0.56	94.44	0.821
Ours (full training set)	4.84	0.42	96.30	0.874
Ours (single object)	3.94	0.44	94.51	0.848

where the sensor setup might change. In fact, only very little training data is required to retrain our method and achieve state-of-the-art reconstruction results.

5.2.5 Loss Ablation.

We have also run an ablation study to evaluate the importance of the different terms in our loss function. In Figure 5.12, we show that the combination of all three loss terms yields best results. The binary cross entropy is particularly useful to improve convergence in the beginning of the training as the network learns to

Method	Burghers		Stonewall		Lounge		Copyroom		Cactusgarden	
	M.A.	M.C.	M.A.	M.C.	M.A.	M.C.	M.A.	M.C.	M.A.	M.C.
TSDF Fusion [Curless and Levoy, 1996a]	21.01	22.58	17.67	21.16	21.88	26.31	39.56	42.57	18.91	18.63
RoutedFusion	20.50	41.32	19.44	80.54	22.63	53.45	38.07	57.35	19.20	41.41
Ours	18.19	18.88	17.01	20.27	16.45	17.96	19.06	20.25	15.87	16.96

Table 5.3: Quantitative evaluation on Scene3D [Zhou and Koltun, 2013]. Evaluated are mesh accuracy (M.A.) [mm] & mesh completeness (M.C.) [mm].

predict a coarse shape that is further refined by the losses on the SDF as training progresses.

5.2.6 Real-World Data

We also evaluate on real-world data and large-scale scenes to demonstrate scalability and generalization.

Scene3D Dataset. For real-world data evaluation, we use the lounge and stonewall scenes from the Scene3D dataset [Zhou and Koltun, 2013]. For comparability to RoutedFusion, we only fuse every 10th frame from the trajectory using a model solely trained on synthetic ModelNet [Wu et al., 2015] data augmented with artificial noise and outliers.

In Figure 5.13, we present qualitative results of reconstructions from real-world depth maps compared to RoutedFusion and TSDF Fusion [Curless and Levoy, 1996a]. We note that our method reconstructs the scene with significantly higher completeness than RoutedFusion. This is due to our learned translation from feature to TSDF space, which allows to better handle noise artifacts and outliers without the need for hand-tuned, heuristic post-filtering. Further, we show improved outlier and noise artifact removal compared to TSDF Fusion [Curless and Levoy, 1996a] while being on par with respect to completeness. These results are also quantitatively shown in Table 5.3.

Tanks and Temples Dataset. In order to demonstrate our methods outlier handling capability, we also run experiments on the Tanks and Temples dataset [Knapitsch et al., 2017]. We computed stereo depth maps using COLMAP [Schönberger and Frahm, 2016, Schönberger et al., 2016] and fused this data using our method, PSR [Kazhdan and Hoppe, 2013], TSDF Fusion [Curless and Levoy, 1996a], and RoutedFusion. To demonstrate the easy applicability to new datasets in scenarios with limited ground-truth, we train our method on one single scene

(Ignatius) from the Tanks and Temples training set. We reconstructed a dense mesh using Poisson Surface Reconstruction (PSR) [Kazhdan and Hoppe, 2013], rendered artificial depth maps, and used TSDF Fusion to generate a ground-truth SDF. Then, we used this ground-truth to train the fusion of stereo depth maps. Figure 5.14 shows the reconstructions of the unseen Caterpillar, Truck, and M60 scene from [Knapitsch et al., 2017]. Our proposed method significantly reduces the amount of outliers in the scene across all models. While RoutedFusion shows comparable results on some scenes, it is heavily dependent on its outlier post-filter, which fails as soon as there are too many outliers in the scene (see also Figure 5.1). Further, they also pre-process the depth maps using a 2D denoising network while our network uses the raw depth maps.

5.3 Discussion

5.3.1 Limitations

Compared to RoutedFusion, we have addressed the limitation of heuristic-based outlier filtering in this chapter. This led to an improvement in outlier handling and cleaner final reconstructions. Yet, the pipeline is still trained on synthetic data. The outlier distribution is artificially modelled in the synthetic data. While this approximation shows impressive results, better performance could be achieved if the pipeline is trained on real data with real outlier distributions, *e.g.* from time-of-flight sensors or multi-view stereo pipelines. Given the generalization capabilities, the dataset does not necessarily need to be large but the ground-truth needs to be accurate to avoid introducing unwanted biases into the reconstruction pipeline.

Large and diverse datasets with accurate groundtruth would be also beneficial, as the network could learn strong priors that are especially valuable for completion during the translation. However, acquiring large-scale datasets with accurate 3D ground-truth is expensive and representing this ground-truth in a dense voxel-grid is memory intensive. Thus, this limitation could be resolved by moving the supervision from the 3D space to the 2D domain by rendering depth maps from the geometry and supervising it on the sensor streams and coupling it with RGB

supervision.

Moreover, while our pipeline shows excellent generalization capabilities, *e.g.* generalizing from a single MVS training scene, it is biased to the number of observations integrated during training leading to less complete results on some test scene parts with very few observations. However, this issue can be overcome by a more diverse set of training sequences with different number of observations.

5.3.2 Summary

In this chapter, we presented a novel approach to online depth map fusion with real-time capability that leverages a learned scene representation. The key idea is to perform the fusion operation in a learned latent space that allows to encode additional information about undesired outliers and super-resolution complex shape information. The separation of scene representations for fusion and final output allows for an end-to-end trainable post-filtering as a translator network, which takes the latent scene encoding and decodes it into a standard TSDF representation. Our experiments on various synthetic and real-world datasets demonstrate superior reconstruction results, especially in the presence of large amounts of noise and outliers.

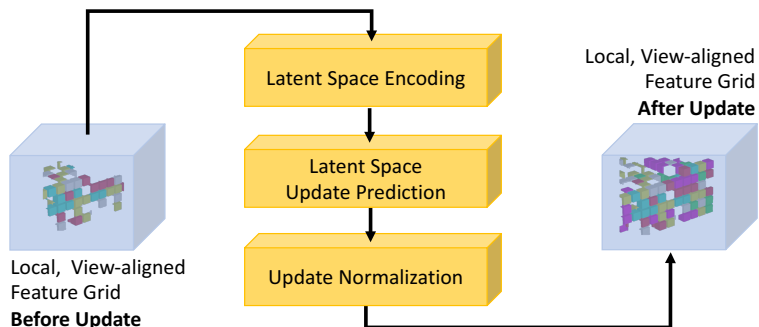


Figure 5.3: Feature Fusion Network. The feature fusion network consists of a latent space encoder that fuses information from neighboring rays. This is followed by a latent updated predictor that predicts the updates for the latent space. Finally, the predicted features are normalized along the feature vector dimension.

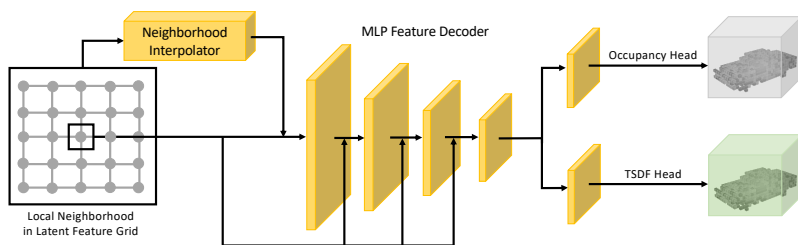


Figure 5.4: Translator Network. The translator network consists of a series of neural blocks with linear layers, channel-wise dropout, and tanh activations. The first block extracts neighborhood information that is concatenated with the central feature vector. From the concatenated features, the TSDF value is then predicted. All joining arrows correspond to a concatenation operation.

Method	MSE↓ [e-05]	MAD↓ [e-02]	Acc.↑ [%]	IoU↑ [0,1]	F1↑ [0,1]
DeepSDF [Park et al., 2019]	464.0	4.99	66.48	0.538	0.66
Occ.Net. [Mescheder et al., 2019]	56.8	1.66	85.66	0.484	0.62
IF-Net [Chibane et al., 2020]	6.2	0.47	93.16	0.759	0.86
TSDf Fusion [Curless and Levoy, 1996a]	11.0	0.78	88.06	0.659	0.79
TSDf + 2D denoising	27.0	0.84	87.48	0.650	0.78
TSDf + 3D denoising	8.2	0.61	94.76	0.816	0.89
RoutedFusion	5.9	0.50	94.77	0.785	0.87
Ours	2.9	0.27	97.00	0.890	0.94

Figure 5.5: Quantitative and qualitative results on ShapeNet [Chang et al., 2015]. Our fusion approach consistently outperforms all baselines and state of the art in both, scene representation and depth map fusion. The performance differences to [Weder et al., 2020] are also visualized in Figure 5.6.

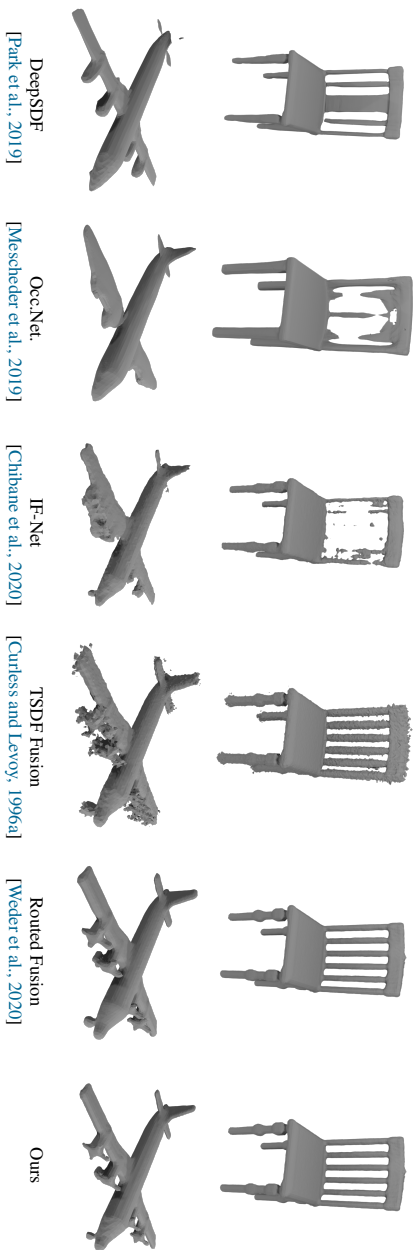




Figure 5.6: Mesh Accuracy (M.A.) visualization on ShapeNet meshes. Our method consistently reconstructs more accurate meshes than the baseline depth fusion methods. Especially thin geometries (table/chair legs, lamp cable) are reconstructed with better accuracy.

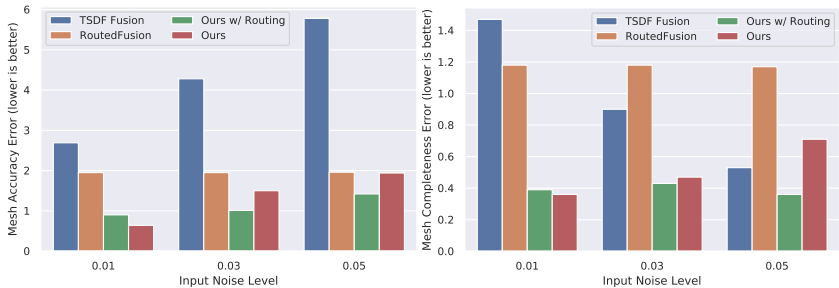


Figure 5.7: Reconstruction from Noisy Depth Maps. Our method outperforms existing depth fusion methods for various input noise levels. The performance can be further boosted by preprocessing the depth maps with the routing network from RoutedFusion leading to better robustness to high input noise levels, while it over-smooths in the absence of noise.

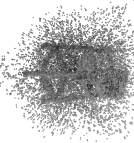
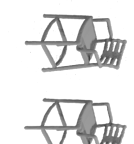
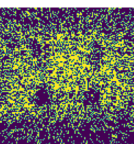
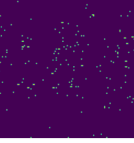


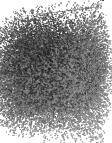
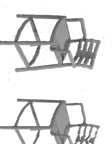
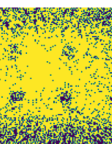
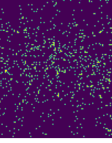


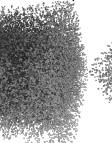
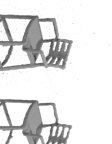
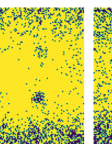
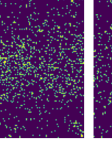

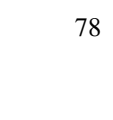
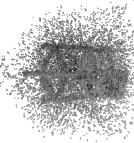
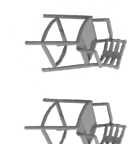
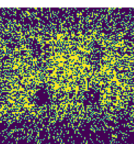
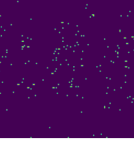


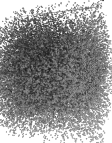
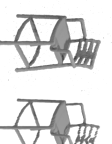
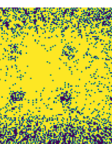
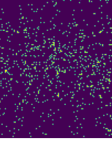


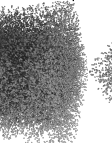
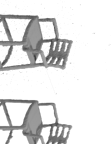
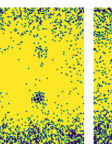
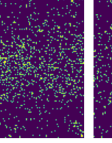

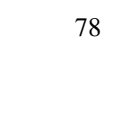
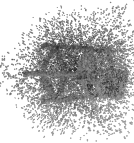
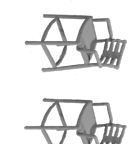
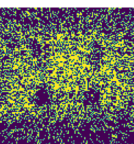
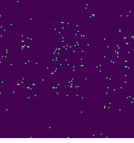


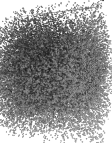
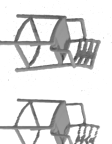
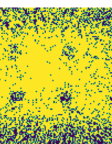
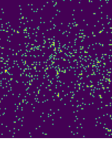


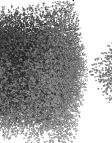
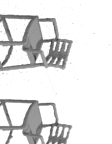
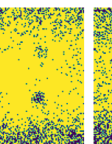
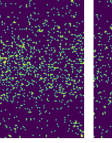

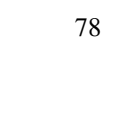
Outlier Fraction	Method	MSE \downarrow	MAD \downarrow	Acc \uparrow	IoU \uparrow	Reconstructed Geometry			Outlier Projection onto XY-Plane		
		[e-05]	[e-02]	[%]	[0,1]	TSDf Fusion	RoutedFusion	Ours	TSDf Fusion	RoutedFusion	Ours
0.01	TSDf Fusion	34.51	1.17	85.17	0.645						
	Routed-Fusion	5.43	0.57	95.21	0.837						
	Ours	2.27	0.29	97.57	0.884						
0.05	TSDf Fusion	80.72	2.02	73.86	0.432						
	Routed-Fusion	9.84	0.68	94.46	0.803						
	Ours	4.91	0.22	98.05	0.851						
0.1	TSDf Fusion	102.50	2.43	67.47	0.341						
	Routed-Fusion	14.25	0.77	92.95	0.764						
	Ours	3.35	0.22	98.48	0.865						

Figure 5.8: Reconstruction from Outlier-Contaminated Data. The left table states performance measures for various outlier fractions. The parts/sources/02-neural-fusion/figures on the right show corresponding reconstruction results and errors projected on the xy-plane for an exemplary ModelNet [Wu et al., 2015] model. Our method outperforms state-of-the-art depth fusion methods regardless of the outlier fraction, but in particular with larger outlier amounts. Note that high outlier rates are common in multi-view stereo as shown in the supp. material of [Knäpitsch et al., 2017].

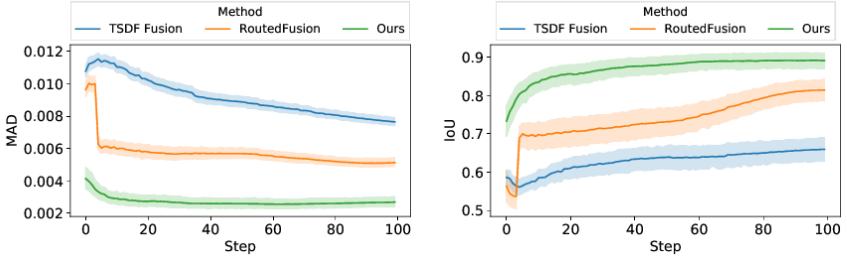


Figure 5.9: Performance of iterative fusion over time. Our method consistently outperforms both baselines - RoutedFusion and TSDF Fusion - at every step of the fusion procedure.

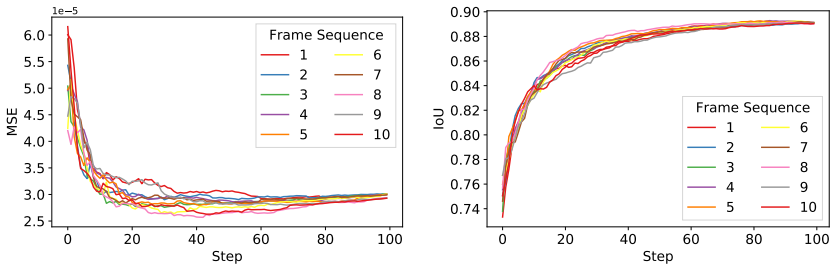


Figure 5.10: Random frame order permutations. The proposed method seems to be largely invariant to the frame integration order, since it always converges to the same result.

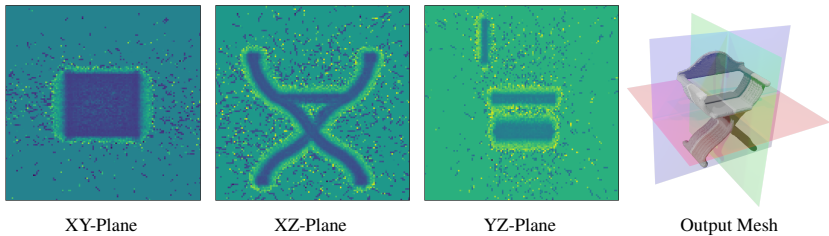


Figure 5.11: Visualization of our learned latent space encoding. Our asynchronous fusion network integrates all measurements including outliers, but the translator effectively filters the outliers to generate a clean output mesh.

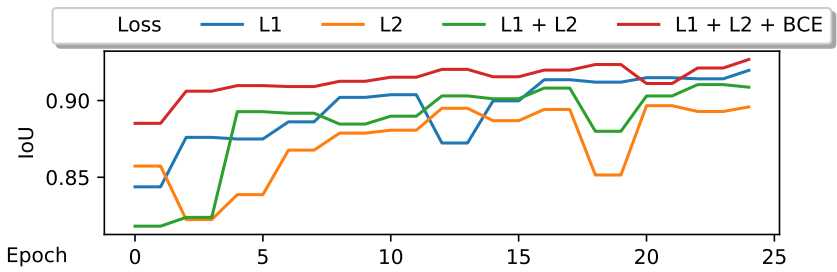


Figure 5.12: Loss Ablation. All losses combined yield best results.

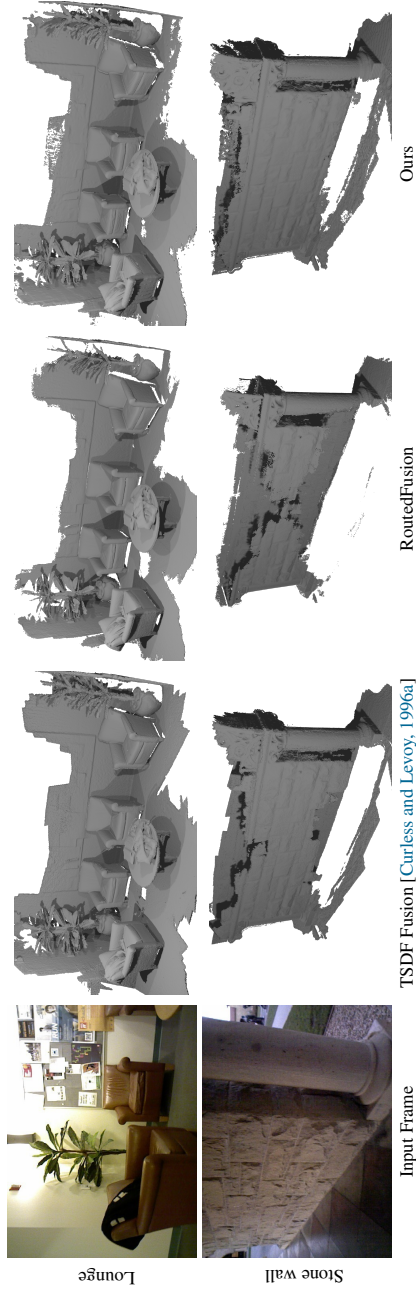


Figure 5.13: Depth map fusion results on Scene3D [Zhou and Koltun, 2013]. Our method yields significantly better completeness than RoutedFusion (see stonewall) and is on par with TSDF Fusion [Curless and Levoy, 1996a]. However, our method better removes noise and outlier artifacts (see lounge reconstruction).

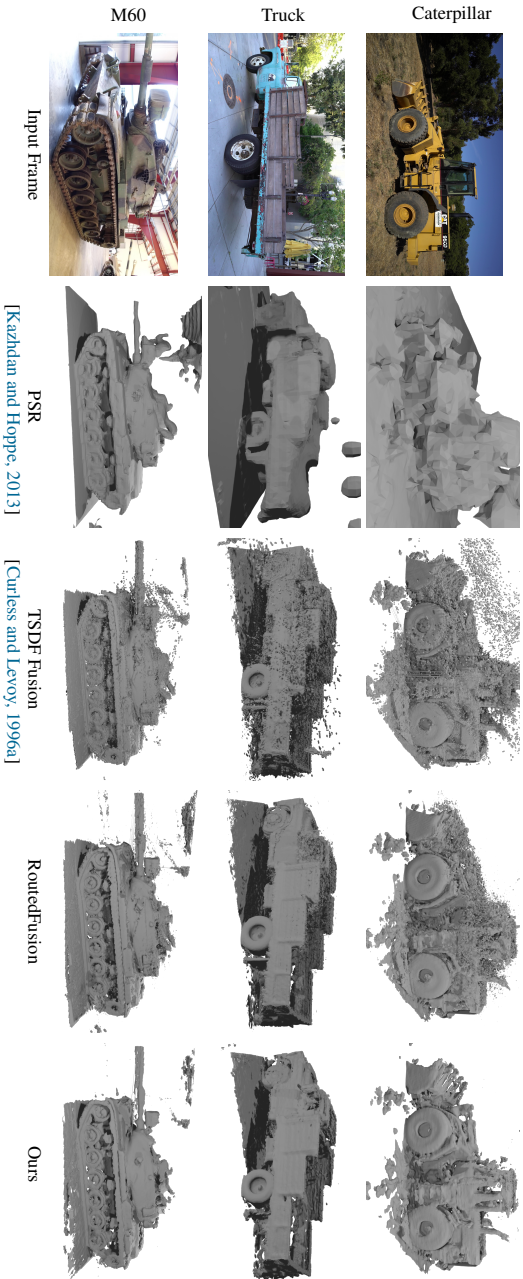


Figure 5.14: Results on Tanks and Temples [Knapitsch et al., 2017]. Our method significantly reduces the number of outliers compared to the other methods without using any outlier filtering heuristic and solely learning it from data. We especially highlight the results on the caterpillar scene, where our proposed method filters most outliers while the reconstructions of competing methods are heavily cluttered with outliers.

Learning-based Appearance Fusion

So far, we have only considered the geometric reconstruction of the 3D world around us. However, oftentimes it is not sufficient to only reconstruct the geometry. Many applications also require the reconstruction of the appearance of the world. *I.e.*, we reconstruct how the world “really” looks like. This allows to not only use the underlying geometry of the world, but also leverage the reconstruction for immersive experiences through rendering onto the users devices. *E.g.*, this technology would enable a real-time stream of the environment to another location such that it can be explored remotely from novel views.

To this end, realistic 3D model reconstruction from images and depth sensors has been a central and long-studied problem in computer vision. Appearance mapping is often treated as a separate post-processing step that follows 3D surface reconstruction and is usually approached using batch-based optimization methods [Debevec et al., 1996, Eisemann et al., 2008, Waechter et al., 2014, Fu et al., 2018b] that are unsuitable for many applications that do not have access to the entire dataset at processing time, for instance, robot navigation [Breitenmoser and Siegwart, 2012, Garrido et al., 2013, Bircher et al., 2015], augmented reality [Newcombe et al., 2011b, Schöps et al., 2017a], and virtual reality [Lombardi et al., 2018, Lombardi et al., 2019, Chu et al., 2020] applications, Simultaneous Localization and Mapping (SLAM) systems [Whelan et al., 2015], online scene

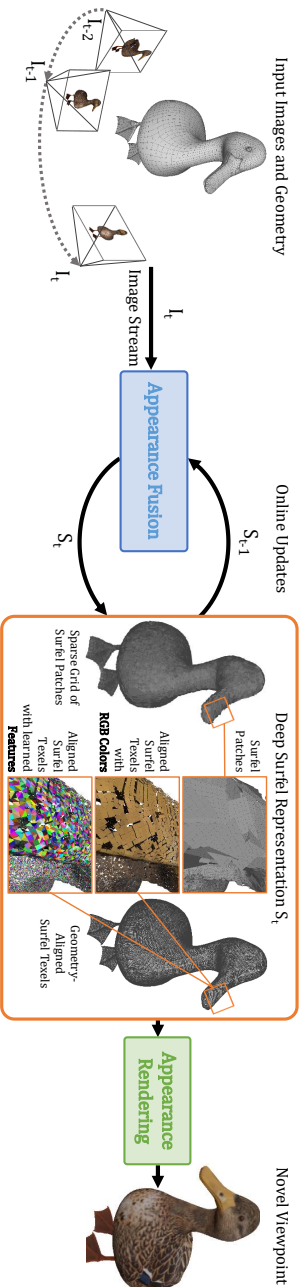


Figure 6.1: Overview of our online appearance fusion pipeline and the DeepSurfel scene representation. The **Appearance Fusion** network efficiently aggregates appearance information from a stream of camera views into the proposed DeepSurfel representation S_{t-1} that maintains high-frequency geometric and appearance information. DeepSurfels is a sparse grid of 2D patches that consist of surface-aligned textels, which encode appearance information either as RGB color values or learned feature vectors. The proposed **Appearance Rendering** network interprets aggregated and interpolated geometric and appearance information stored in DeepSurfels for rendering novel viewpoints. In this example we used DeepSurfels with a sparse 64^2 patch grid with 8×8 resolution surfel patches.

perception methods [Häne et al., 2017, Schöps et al., 2017b], and many others.

Common online fusion methods like KinectFusion [Newcombe et al., 2011a] and previously introduced RoutedFusion and NeuralFusion are well suited for online geometry fusion and can efficiently handle noise and topological changes. However, due to their high memory requirements at high voxel resolutions, they have strong limitations when it comes to encoding high-frequency appearance details on the surface. On the other hand, meshes with high-resolution texture maps [Waechter et al., 2014, Fu et al., 2018b, Eisemann et al., 2008] are well-suited for encoding high-frequency appearance information in an efficient manner, but they have difficulties in handling topology changes in an online reconstruction setting. Moreover, recent learning-based approaches [Sitzmann et al., 2019a, Sitzmann et al., 2019b, Mildenhall et al., 2020a, Oechsle et al., 2019] have achieved high-quality results by learning geometry and texture mapping directly from RGB images. However, they are not well suited for local online updates, do not scale to large-scale scenes, and easily overfit to the training data.

In this chapter, we approach the problem of online appearance reconstruction from RGB-D images by combining the advantages of **1)** implicit grids, which easily handle topological changes and where low resolution is often sufficient to encode the scene topology, **2)** scalable high-frequency appearance along the surface via texture maps or learned feature maps, and **3)** a learned scene representation to build a framework for learning-based appearance fusion that allows for online processing and scalability to large scenes. To this end, we propose a novel scene representation *DeepSurfels* and an efficient learning-based online appearance fusion pipeline, which is illustrated in Figure 6.3.

Our *DeepSurfels* representation is a hybrid between an implicit surface that encodes the topology and low-frequency geometric details and a surfel representation that encodes high-frequency geometry and appearance information in form of surface-aligned patches. These patches are arranged in a sparse grid and consist of surface-aligned texels that encode appearance information *either* in the classical form of RGB color values *or*, as proposed, via learned feature vectors. The sparse grid allows for efficient volumetric rendering and enables explicit scene updates that are crucial for online fusion, while the 2D patches enable quadratic memory

storage complexity like meshes or sparse grid structures. Depending on the DeepSurfel parameters it can approximate between simple colored voxels (high grid resolution, 1×1 patches) and textured meshes with high texture atlas resolutions (lower grid resolution, higher patch resolution). Our online appearance fusion pipeline iteratively fuses RGB-D frames into estimated DeepSurfels geometry and is optimized by using a differentiable renderer for self-supervision and the reprojection error as training signal. In this way, the pipeline does not require any ground-truth texture maps and the training procedure allows for efficient transfer to new sensors and scenes without the need for acquiring costly ground-truth data.

While we eventually target full online reconstruction of both geometry and texture from monocular video, we only focus on online appearance estimation in this chapter. Even in a setting with known geometry, our online approach has scalability advantages: We can fuse arbitrary numbers of input frames and the grid-aligned surfels have performance advantages during feature aggregation across local neighbors and for controlling the sampling density. Our grid-aligned surfel patches can also be seen as a spatial alignment of per-voxel sub-features being anchored along the surface. In contrast to works that only save a single feature vector per voxel, *e.g.* DeepVoxels [Sitzmann et al., 2019a], we can directly relate sub-features with particular image pixels via projective mapping and as such simplify the network learning task and improve output accuracy. As opposed to many novel view synthesis works [Sitzmann et al., 2019b, Sitzmann et al., 2019a, Mildenhall et al., 2020a], we do not overfit onto a single scene, but train a network that generalizes over multiple scenes without re-training. While those methods iterate many times over each input image in a slow optimization process, our method processes every image only once with a single network forward pass and is thus much faster. From the application point of view, our approach is thus closer to classical texture mapping methods like [Debevec et al., 1996, Eisemann et al., 2008, Fu et al., 2018b, Waechter et al., 2014].

We compare our novel scene representation and appearance fusion pipeline to existing methods on single and multi-object datasets and show that our scene representation better captures high-frequency textures. Moreover, our method generalizes well and compares favorably even to existing texture optimization

methods that jointly optimize all images together. This is a crucial step towards a fully end-to-end appearance fusion method that can be deployed to real-world applications. In summary, our key contributions of this chapter are:

- A novel scalable and memory-efficient 3D scene representation, termed *DeepSurfels*, closing the gap between traditional interpretable and modern learned representations.
- An end-to-end differentiable and efficient online appearance fusion pipeline compatible with classical and learned texture mapping. The method yields competitive texturing results without heavy optimization as every input frame is processed only *once* with a single network forward pass.
- Contrary to other learning-based novel view synthesis methods [Sitzmann et al., 2019a, Sitzmann et al., 2019b, Mildenhall et al., 2020a] that overfit onto a single scene, our method generalizes to new scenes without retraining.

6.1 DeepSurfels 3D Scene Representation

We propose *DeepSurfels* as a powerful, scalable, and easy-to-use alternative to mitigate previously mentioned problems of many scene representations.

6.1.1 Data Structure

DeepSurfels is a set of patches with $L \times L$ texels that can either store color information or learned feature vectors. The elementary building block is an oriented texel $\tau \in \mathbb{R}^c$ that is associated with its weight parameter ω and is stored on the objects' surface, where c denotes the number of feature channels. This number can be chosen arbitrarily for learned appearance fusion as suited for the problem setting, while we set $c = 3$ for deterministic RGB texturing. The texels τ are arranged in an $L \times L$ resolution patch $P_{xyz} : \{i, j \rightarrow \tau_{ij}; i, j \in [1, L]\}$ that is located in a sparse patch grid $\mathcal{P} = \{P_{xyz}\}_{x \leq X, y \leq Y, z \leq Z}$, where X, Y, Z represent DeepSurfels' grid resolution. Although the spatial patch size can be chosen arbitrarily, we empirically observed that texturing works best when the

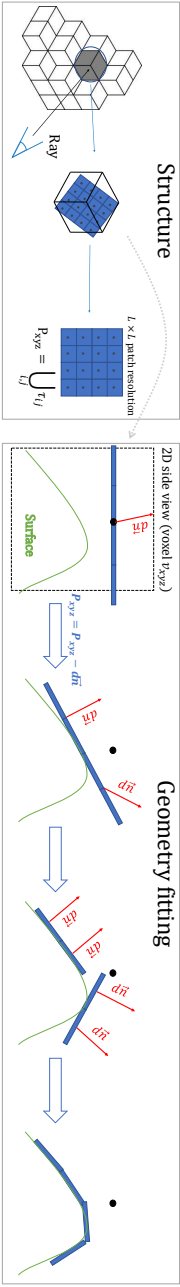


Figure 6-2: DeepSurfel surface fitting. In a recursive fitting procedure, we align the texels of each patch with the underlying SDF surface by shifting and alignment. In every step of the recursion, we shift the (sub-)patch onto the surface and align its normal with the normal of the surface. d denotes queried signed distance, \mathbf{n} denotes SDF gradient ∇SDF in x, y, z directions.

patch size is equal to the grid cell size such that there is no overlap between neighboring patches. For efficiency reasons, it is sufficient to store patches only for grid cells that intersect the objects’ surface. However, it is also possible to allocate more layers around the iso-surface to account for noisy geometry as it is common for geometric fusion approaches [Newcombe et al., 2011a].

6.1.2 Surface Fitting

We propose a recursive algorithm to align each texel τ_{ij} of the patch with the implicit surface of the geometry. We compute the patch position and orientation from a signed distance function (SDF) representing the Euclidean distance to the closest surface.

Initially, every patch P_{xyz} in the grid \mathcal{P} is positioned at the center of its grid cell. Then, the patches are shifted to the closest surface by using the pre-computed SDF, oriented according to the SDF gradient ∇SDF in all x, y, z directions, and rotated to maximize the surface coverage. These patches are subdivided into κ^2 non-overlapping patches of $\frac{L}{\kappa} \times \frac{L}{\kappa}$ resolution, where $\kappa \geq 2$ is the smallest integer to non-trivially divide L . Each sub-patch is aligned again using the SDF field, where we trilinearly interpolate the SDF value at non-integer grid positions. This patch subdivision and alignment is repeated recursively until the resolution reaches 1×1 when patches represent texels that lie on the isosurface. This process is visually illustrated in Figure 6.2.

6.2 Online Appearance Fusion Pipeline

We also propose a pipeline for learning appearance fusion (depicted in Figure ??) that incrementally fuses RGB measurements into DeepSurfels at every time step t and yields DeepSurfel state S_t . The input to our pipeline are intrinsic camera parameters K_t and extrinsic camera parameters R_t , an RGB image $I_t \in \mathbb{R}^{H \times W \times C}$, and corresponding depth map $D_t \in \mathbb{R}^{H \times W}$, where H, W and C denote image height, width, and the number of channels respectively. The pipeline consists of four main components detailed in the following.

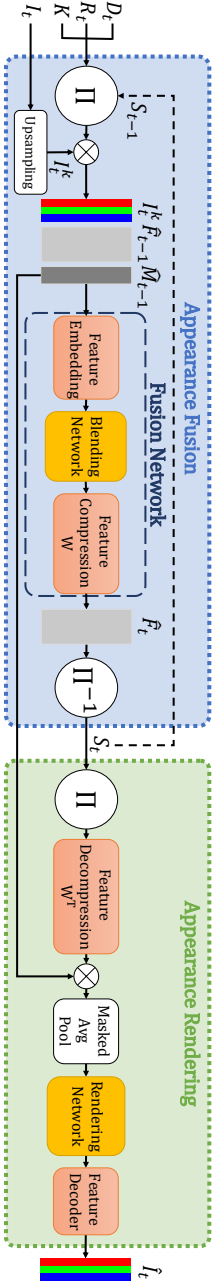


Figure 6.3: Overview of our learned appearance fusion pipeline. The pipeline consists of an **Appearance Fusion** module that integrates a new RGB measurement I_t into DeepSurfels S_{t-1} and a differentiable **Appearance Rendering** module that interprets and renders the content of representation F_t for a given viewpoint. White blocks denote differentiable deterministic operations, rectangular blocks denote data, rounded rectangular blocks are trainable modules, and \otimes is a feature stacking operation.

6.2.1 Differentiable Projection II

The projection module renders a super-resolved feature map $\hat{F}_{t-1} \in \mathbb{R}^{kH \times kW \times c}$, where k is an upsampling factor inspired by [Cook et al., 1984] to ensure dense coverage of the geometry. There are three steps to render this feature map from already stored scene content.

First, each pixel in the incoming frame D_t is subdivided into k^2 distinct sub-pixels p_{ij}^t ($i \in [1, kH]$, $j \in [1, kW]$), thus forming an upsampled image grid. Second, by leveraging camera and depth information, the center of the sub-pixel p_{ij}^t is un-projected into the scene. From the un-projected scene point, the closest DeepSurfel texel and all texels within the surrounding l_∞ ball are selected. The size of this ball is chosen proportional to the size of the un-projected sub-pixel in the world space. Third, an efficient uniform average of the selected texels determines the value of the feature entry $\hat{f}_{ij}^{t-1} \in \hat{F}_{t-1}$ (6.1):

$$\hat{f}_{ij}^{t-1} = \frac{1}{|T_{ij}^t|} \sum_{\tau \in T_{ij}^t} \tau, \quad (6.1)$$

where T_{ij}^t is the set of selected texels. This algorithm is simple, leverages the grid representation for fast rendering, and can flexibly render optionally stored features or a surface normal map \hat{N}_{t-1} that we jointly denote as meta features \hat{M}_{t-1} . Note that all operations are differentiable and the selection can be implemented as a differentiable multiplication by an indicator function.

6.2.2 Fusion Network

The input image I_t is deterministically upsampled $I_t^k \in \mathbb{R}^{kH \times kW \times C}$ by factor k (nearest-neighbor interpolation) and stacked \otimes with the super-resolved features $\hat{F}_{t-1} \otimes \hat{M}_{t-1} \otimes I_t^k$. This stacked representation is embedded into a higher-dimensional feature space by a trainable linear transformation (*Feature Embedding* module Figure ??). Then, the embeddings are refined by the *Blending Network* that consists of five convolutional layers (3×3 kernel size) interleaved with dropout and leaky ReLU activations. This network, based on a small receptive field, produces refined features aware of neighboring information that alleviates

the problem of discretization artifacts, which can occur for low DeepSurfels resolutions. Lastly, these features are compressed by the *Feature Compression* W layer to a lower dimensional feature space that is defined by DeepSurfels' number of channels. The final output is an updated feature map \hat{F}_t that blends old information from \hat{F}_{t-1} with the new appearance information from I_t .

6.2.3 Inverse Projection Π^{-1}

While the fusion module and the explicit geometry representation preserve spatial coherence, this module is responsible for integrating the new appearance information in a temporally coherent way. Without temporal coherence, a new observation could overwrite old states minimizing the reprojection error for the current frame while erasing valuable prior information. The inverse projection module Π^{-1} integrates the updated feature map \hat{F}_t into the representation S_{t-1} to produce the new state S_t . For efficiency reasons, only texel values $\forall \tau_{t-1} \in \bigcup_{i,j}^{kH,kW} T_{i,j}^{t-1}$ and their weights ω_{t-1} that were intersected by at least one of the sub-pixels are updated using the following moving average scheme:

$$\begin{aligned} \tau_t &= \frac{1}{\omega_{t-1} + 1} \left(\tau_{t-1} \omega_{t-1} + \frac{\sum_{i,j}^{kH,kW} \hat{f}_{ij}^t \mathbb{I}_{\tau_{t-1} \in T_{ij}^{t-1}}}{\sum_{i,j}^{kH,kW} \mathbb{I}_{\tau_{t-1} \in T_{ij}^{t-1}}} \right), \\ \omega_t &= \omega_{t-1} + 1, \end{aligned} \tag{6.2}$$

where \mathbb{I}_E is an indicator function being one, if E is true, and zero otherwise. The texel weights are initialized to $\omega_0 = 0$.

The new state S_t is optimally computed in 2D space without interrupting the gradient flow. This way, the scene is seamlessly stored in RAM or disk and can only be partially loaded and updated, which is crucial for scalability.

6.2.4 Appearance Rendering Module

In a first step, this module extracts compressed scene content S_t using Π and embeds these features into a higher dimensional space via a transposed linear compressor (*Feature Decompression* W^T) which acts as a regularizer. Pre-

computed meta features \hat{M}_{t-1} are optionally concatenated and all features are downsampled by a custom masked average pooling with a stride of k and $k \times k$ kernel size, where the mask indicates which features to ignore (features that are empty or located outside the scene space). The current $H \times W$ resolution feature map is passed through the seven-layer convolutional *Rendering Network* refining features and filling potential holes that occur when the scene representation is sparsely populated. Lastly, the high-level features are decoded to RGB values by (*Feature Decoder*) three linear layers interleaved with Leaky ReLU activation functions. The final output is activated using HardTanh activation for generating valid normalized RGB values.

6.2.5 Loss and Optimization

The entire pipeline is trained end-to-end from scratch until convergence using the reprojection error between the rendered image \hat{I}_t and input image I_t as self-supervision. Thus, the network can learn to optimally fuse and encode appearance information from 2D training data without any ground-truth textures. Our pipeline is trained using a weighted combination of \mathcal{L}_1 and \mathcal{L}_2 loss between input image I_t and rendered image \hat{I}_t given by

$$\mathcal{L}(I_t, \hat{I}_t) = \frac{1}{C \cdot H \cdot W} \sum_{p \in I_t, \hat{p} \in \hat{I}_t} \|p - \hat{p}\|_1 + \frac{1}{2} \|p - \hat{p}\|_2 \quad (6.3)$$

We empirically found that a $1 : \frac{1}{2}$ weight ratio worked best in our experiments. The entire pipeline has less than 0.6M parameters and was optimized using the Adam optimizer [Kingma and Ba, 2015] with a learning rate of 10^{-4} and batch size 1, except for the generalization experiment, where we used 2.

6.3 Evaluation

We evaluate our method by comparing the representation power of both, our learned and our deterministic approach (direct rendering from RGB surfels) with state-of-the-art methods on novel view synthesis tasks. We further demonstrate









GT		Ours		Deterministic		[Fu et al., 2018]		Texture Fields		TSDF Coloring		[Weachter et al., 2014]		Surfel-Meshing	
	PSNR↑ / SIM↓:		32.94 / 0.950		29.44 / 0.889		32.14 / 0.912		27.99 / 0.856		24.89 / 0.621		18.52 / 0.631		9.928 / 0.510
	PSNR↑ / SIM↓:		35.42 / 0.966		32.03 / 0.958		27.23 / 0.835		27.80 / 0.841		27.37 / 0.810		16.24 / 0.370		12.31 / 0.458

Figure 6.4: Qualitative and quantitative comparison on novel view synthesis with DeepSurfels on a 128^3 sparse grid with learned 3-channel 4×4 feature patches. The experiment demonstrates that our scene representation is able to better represent high-frequency textures compared to other state-of-the-art methods. "Ours deterministic" shows direct rendering from RGB surfel patches. Please note that SurfelMeshing [Schöps et al., 2019] is the only method in this comparison which also estimates geometry while the other methods use known geometry.

how our method generalizes to different scenes for a small number of distinct training samples and provide an ablation study to validate the design choices for our model.

6.3.1 Datasets

We conduct experiments on datasets generated from Shapenet [Chang et al., 2015], publicly available human and cat¹ models, the indoor Replica dataset [Straub et al., 2019], and the cube scene from [Sitzmann et al., 2019a]. Replica dataset images were rendered with Habitat-Sim [Savva et al., 2019] and all other models with Blender [Community, 2020].

6.3.2 Metrics

We quantify model performances with the following two metrics [Wang et al., 2004].

PSNR. The Peak Signal-to-Noise Ratio (PSNR) is the ratio between the maximum pixel value in the ground-truth image and the pixel-wise mean-squared error between ground-truth and rendered image.

SSIM. The Structural Similarity Index (SSIM)s measures similarity between patches of rendered and ground-truth images. We omit other perceptron based metrics because we are interested in recovering the true pixel value as our fusion approach can be used for more general types of data.

6.3.3 Novel View Synthesis

The model is optimized on 500 randomly rendered 512×512 training images for the cat and human model and the results for a single unseen frontal viewpoint² are compared with state-of-the-art batch-based methods (Fu et al. [Fu et al., 2018b], Texture Fields [Oechsle et al., 2019], Waechter et al. [Waechter et al., 2014]), and online methods (SurfelMeshing [Schöps et al., 2019], TSDF Coloring [Curlless and Levoy, 1996a]) on a 128^3 grid. The results in Figure 6.4

¹3D models from free3d.com and turbosquid.com.

²For a fair comparison with the results of Texture Fields [Oechsle et al., 2019].

demonstrate that our approach compares favorably even to slower batch-based methods in representing high-frequency textures. Figure 6.7 further shows that our approach does not suffer from blurry artifacts as the recently proposed SRNs [Sitzmann et al., 2019b], or from multi-view consistency issues like DeepVoxels [Sitzmann et al., 2019a]. Note that these approaches jointly estimate geometry and appearance while we only estimate appearance. Table 6.2 shows the effect of varying the number of channels on the cube dataset for DeepSurfels of 4×4 patches on a 64^3 sparse grid.

6.3.4 Generalization

Our pipeline scales and generalizes well on realistic room-size scenes. We trained our pipeline on 288 480×640 images of one Replica [Straub et al., 2019] room represented with DeepSurfels of 11cm voxel size with (3+3)-channel 6×6 resolution patches. We disentangled 3 color channels to improve generalization. The pipeline evaluation is performed on every 25th unseen frame in a sequence of frames generated by a moving agent in the Habitat Sim [Savva et al., 2019]. Figure 6.5 shows results of our trained pipeline on an optimized (left) and a non-optimized (right) scene. Our learned approach outperforms baselines in representing fine details.

We further demonstrate that our pipeline generalizes well when trained on a larger set of distinct scenes. We render 100 312×312 training images from 150 Shapenet [Chang et al., 2015] car scenes and test the pipeline on 50 unseen scenes by fusing 100 views and evaluating results on additional 60 unseen viewpoints. The whole pipeline is trained to be frame order independent by randomly shuffling scenes and frames after each optimization step. Results on test scenes (Figure 6.6, Table 6.1) indicate that our learned approach improves for discretization artifacts and overall yields sharper results which are supported by higher PSNR and SSIM scores.

6.3.5 Ablation Studies

For the unobserved test car scenes, we quantify in Table 6.1 the impact of: (i) depth as a meta feature that helps our method to reason about the confidence of

updates since pixels with larger depth values are less important; **(ii)** multi-view consistency regularization that corrects for geometric misalignments and improves interpolation among neighboring viewpoints by adding an additional error signal (6.3) for a viewpoint closest to the fused frame; **(iii)** pixel ray directions with surface orientation map to improve reasoning about light information and non-Lambertian surfaces; **(iv)** DeepSurfel parameters (grid and patch resolution). For almost all experiments use 3 feature and 3 disentangled color channels (denoted as 3+3 in Table 6.1) which outperforms the baselines. We observe that every attribute (i-iii) improves generalization and that higher DeepSurfel resolution (iv) consistently benefits reconstruction quality. **(v)** Lastly, we demonstrate the benefit of explicitly modeling the reprojection of pixel colors via texels which can be seen as subfeatures of a large feature vector stored in every voxel. We compare the proposed 6×6 patches, 3+3 channels to 1×1 patches, 213+3 channels which amounts to the same number of features per voxel. We argue that the benefit of using more features per voxel quickly saturates unless subfeatures are anchored along the surface and trained with separate pixel data as supported by our results.

6.3.6 Real-world data

In Fig. 6.8 we present results on unseen real-world data from [Maier et al., 2017b] for which our method yields the most detailed appearance reconstructions.

6.3.7 Runtime

Our method takes 57ms and 21ms for fusing and rendering a single 312×312 frame on 32^3 DeepSurfels with 6-channel patches with resolution 6×6 (Table 6.1). This is significantly faster compared to other deep learning methods that overfit on a single scene. For example, the state-of-the-art method NeRF [Mildenhall et al., 2020a] requires ~ 2 days for training on a single scene being unable to generalize to other scenes, while our method can easily be used on unseen scenes without any optimization as demonstrated in Figure 6.5, which is a speed up of over a thousand times on unobserved scenes for comparable or even favorable results.

6.4 Discussion

6.4.1 Limitations

One of the main limitations of the proposed methods is the assumption of known geometry. We assume that the geometry is given as a signed distance field that can be created from previously proposed fusion methods RoutedFusion and NeuralFusion or standard TSDF fusion [Curless and Levoy, 1996a]. This signed distance field is used to align the proposed DeepSurfel architecture. Because of this assumption, the underlying geometry is fixed and cannot be changed. This fixation makes our proposed method suffer from camera-geometry misalignment that can lead to blurry results as it is the case for standard TSDF Coloring. A joint reconstruction of geometry and appearance would allow us to alleviate the sensitivity to these misalignments by adjusting the geometry on the fly.

A second limitation in order is the lack of modeling view-dependent effects. This leads to washed-out colors due to the local feature averaging. Integrating the view-dependency into the model would allow us to generate higher quality renderings similar to neural radiance fields [Mildenhall et al., 2020a].

The final limitation is given by the limited training data. As we require known geometry, we suffer from the same limitation as the previously presented reconstruction algorithms. However, this limitation is exacerbated by the fact that we also map appearance into learned features. In order to learn this mapping much more training data is required compared to the geometric case to generalize to any test scenario. As we do not have enough training data available, our method sometimes suffers from distorted colors as the model has not seen the correct color during training and cannot represent it.

6.4.2 Summary

We introduced DeepSurfels in this chapter, a novel scene representation for geometry and appearance encoding, that combines explicit and implicit scene representation to improve for scalability and interpretability. It is defined on a sparse voxel grid to maintain topology relations and implements 2D geometry

oriented patches to store high-frequency appearance information. We further presented a learned approach for online appearance fusion that compares favorably to existing offline and online texture mapping methods since it learns to correct for typical noise and discretization artifacts.

As future work we consider the joint online fusion of shape and appearance and address some weaknesses of our appearance fusion pipeline such as the limitation in filling large missing parts and rendering translucent surfaces.

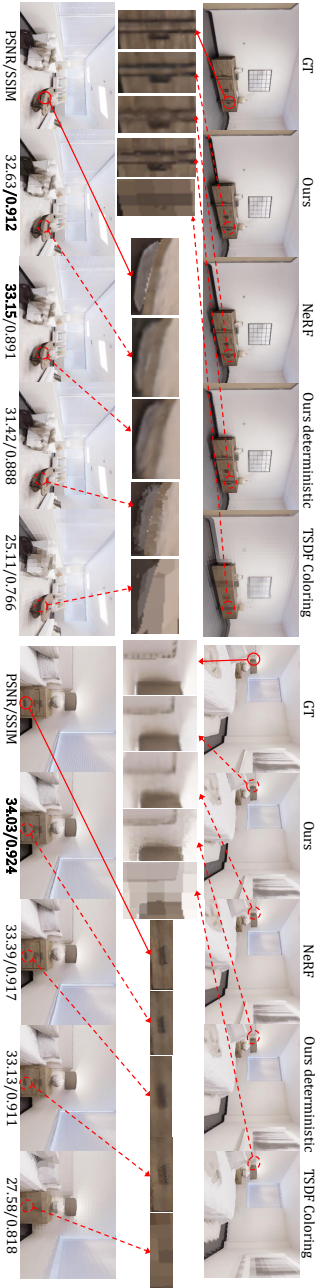


Figure 6.5: Novel view synthesis for Replica [Straub et al., 2019] indoor scenes. The figure shows different views on two scenes (left and right). Our learned approach has been trained only on the room on the left. NeRF [Mildenhall et al., 2020a] is optimized separately on both scenes.

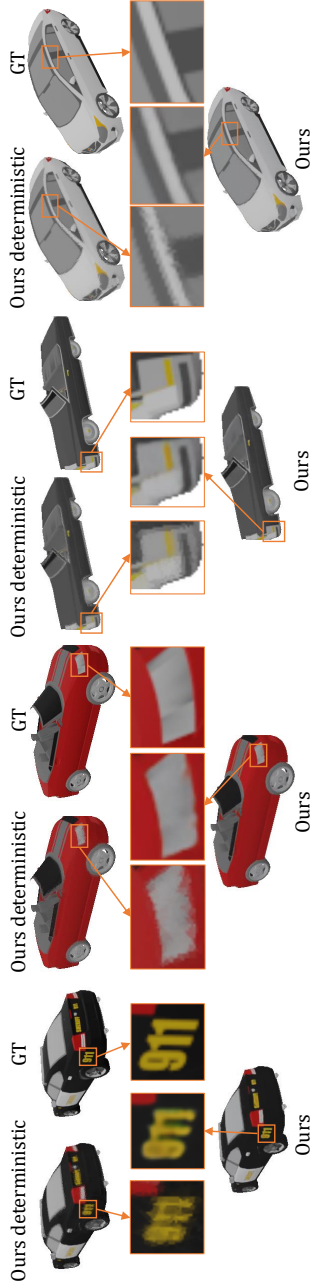


Figure 6.6: Qualitative results of our model on unseen scenes from ShapeNet [Chang et al., 2015]. Compared to deterministic rendering from RGB values, DeepSurfels with learned $(3+3)$ -channel 6×6 patches on a sparse 32^3 grid yields significantly more high-frequency details. As it is shown in the close ups, storing learned features in the texels is particularly useful to correct discretization artifacts.

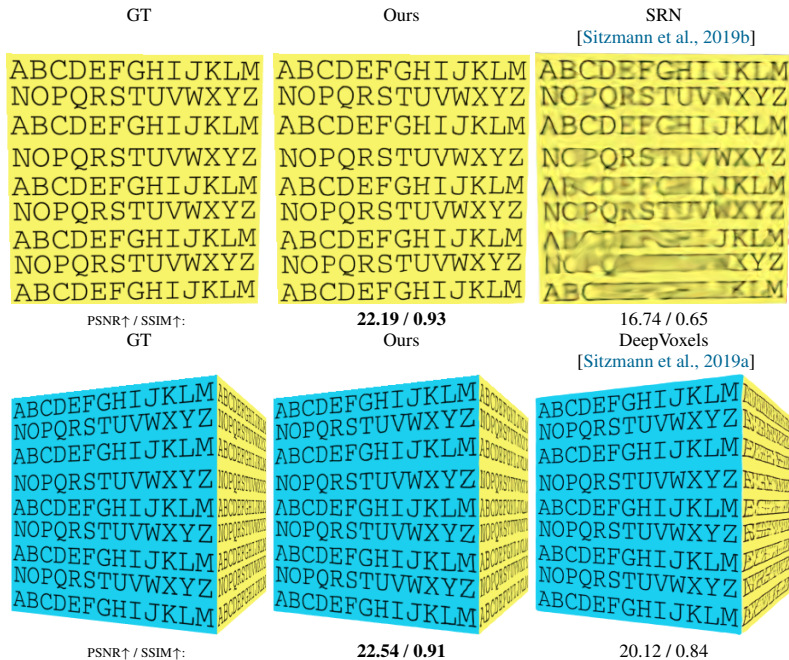


Figure 6.7: Comparison of SRNs [Sitzmann et al., 2019b] and DeepVoxels [Sitzmann et al., 2019a] to our learned DeepSurfel fusion with a 64^3 grid of 8-channel 1×1 resolution feature patches on the synthetic cube dataset from [Sitzmann et al., 2019a]. Our method produces fewer blur artifacts and multi-view inconsistencies and overall yields significantly better images reconstruction results than both baselines. Note that both baselines perform global appearance fusion with unknown geometry.

Table 6.1: Ablation study on ShapeNet [Chang et al., 2015] cars. The **top part** of the table compares various baselines. Our deterministic coloring at 32^3 is still better than TSDF Coloring at 128^3 resolution. The **mid part** shows the impact of the proposed losses. The **bottom part** shows the influence of the voxel grid, surfel patch and channel resolution, demonstrating that quality improvements saturate for higher resolutions. $3+3$ denotes 3 feature and 3 color channels (disentangled) per texel. We also compare to 1×1 patches with $213+3$ channels corresponding to the same number of features as for 6×6 , $3+3$, demonstrating the benefit of a spatial sub-feature alignment in our network.

	Method	PSNR \uparrow	SSIM \uparrow
Baselines	SurfelMeshing [Schöps et al., 2019]	13.92	0.2748
	Wachter <i>et al.</i> [Wachter et al., 2014]	18.27	0.4753
	Fu <i>et al.</i> [Fu et al., 2018b]	18.84	0.5196
	TSDF Coloring [Curless and Levoy, 1996a] (32^3)	21.57	0.6375
	TSDF Coloring [Curless and Levoy, 1996a] (64^3)	24.05	0.7552
	TSDF Coloring [Curless and Levoy, 1996a] (128^3)	26.68	0.8526
	Ours Det. (32^3 , 6×6 , 3)	27.20	0.8723
Ours Det. (64^3 , 4×4 , 3)	28.73	0.9036	

	Learned (32^3 , 6×6 , $3+3$)	28.27	0.8777
Ablation	+ depth	28.31	0.8782
	+ multi-view consist.	28.36	0.8889
	+ viewing direction & surface orientation	28.89	0.8907

DeepSurfel Parameters	32^3 , 1×1 , $213+3$	22.95	0.7083
	64^3 , 1×1 , $213+3$	25.41	0.7940
	64^3 , 4×4 , $3+3$	29.92	0.9086
	64^3 , 5×5 , $3+3$	30.15	0.9126
	64^3 , 6×6 , $3+3$	30.27	0.9147
	128^3 , 1×1 , $213+3$	26.75	0.8324
	128^3 , 2×2 , $3+3$	30.23	0.9133
	128^3 , 3×3 , $3+3$	30.51	0.9181
	128^3 , 4×4 , $3+3$	30.60	0.9196
	128^3 , 5×5 , $3+3$	30.63	0.9200
128^3 , 6×6 , $3+3$	30.64	0.9202	

Table 6.2: Varying number of feature channels for the cube [Sitzmann et al., 2019a] dataset on 64^3 sparse grid with 4×4 patches. Additional feature channels improve the reconstruction quality.

#Channels	PSNR \uparrow	SSIM \uparrow
2	25.95	0.9432
4	26.72	0.9506
6	27.33	0.9568
10	28.27	0.9638

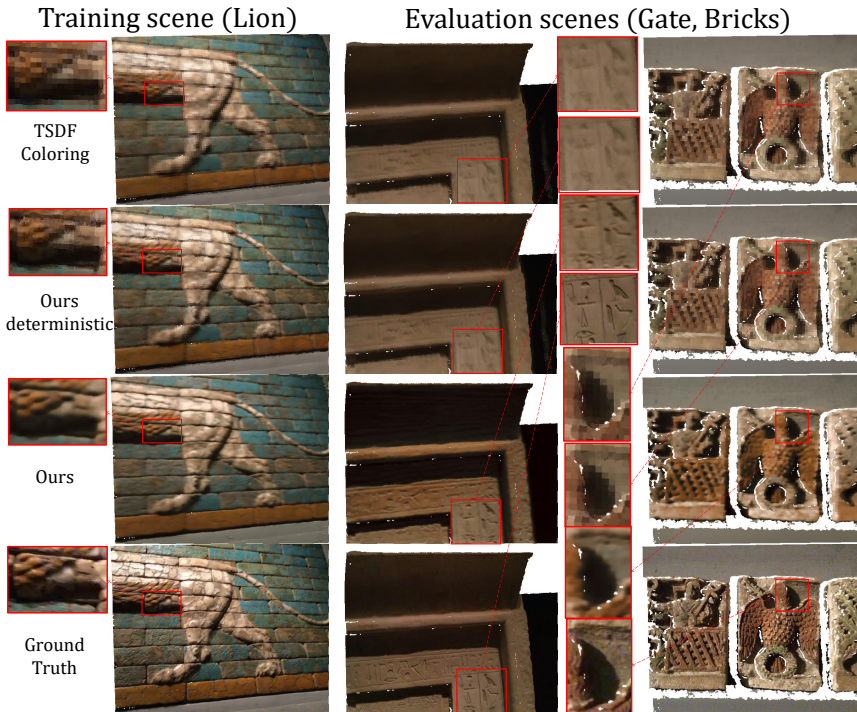


Figure 6.8: Novel-view synthesis on unseen real-world data [Maier et al., 2017b]. Our DeepSurfel method with 4×4 patches is trained on the *Lion* scene for 80 training iterations and then evaluated on the *Gate* and the *Bricks* scenes. The images show novel viewpoints.

Part III

3D Scene Understanding

Introduction

Humans possess both a spatial model and understanding of our environment. A spatial model is not worth much without understanding. The model, *i.e.* a mental map of the world, is only a pre-condition for interacting with it. In order to solve complex problems and accomplish our goals, we also need to have some form of understanding. Where are the chairs I want to rearrange before my friends come over for dinner? And where do I place the cutlery? On the chair or on the table? It is not sufficient to only know the geometry of the objects around us to answer these questions.

Imagine stepping into a room and instantly knowing its purpose, the objects within it, and how they relate to each other. *E.g.*, we recognize not just a chair, but understand its function, its potential interactions, and its place in the greater context of the space. This is the power of 3D scene understanding. What does it mean to truly "understand" a 3D scene? Is it merely recognizing our surroundings, like identifying we are in a kitchen versus a living room? Or is it deeper than that – discerning the function of each object, anticipating how they might interact, and predicting what we can do within that environment?

Understanding our surroundings occurs on various hierarchical levels. At the top level, we discern the type of building or environment we are in – whether it is a house, an office, or a public space. Within this context, we can further differentiate between specific rooms like a living room, bedroom, kitchen, or bathroom. Yet, as we dive deeper into lower levels, our understanding shifts to individual objects within these rooms. We begin to recognize chairs, tables, cups, and countless

other items that give context to the space. Further refinement can even allow us to distinguish exact boundaries between objects. This might be delineating objects of different categories, known as semantic segmentation, or even demarcating instances of the same category, termed instance segmentation. However, this hierarchical understanding is not a one way street; there is an intricate interplay among the levels. Recognizing the purpose of a room often is based on identifying its constituent objects. The presence of a table and a chair might suggest a living room. But context is paramount: in my kitchen, for example, both a chair and a table coexist.

So, how can enable machines with these capabilities? Since the early days of computer vision, understanding has been one of its central fields of research. We will give a review of the background and all relevant related work in the next chapter. In general understanding is driven by the advances in deep learning over the last decade.

For machines and general agents interacting with the world, understanding capabilities mirror the intrinsic cognitive functions humans possess. This understanding aids in planning, navigation, and interaction; think of a robotic vacuum cleaner discerning its way to a kitchen to cleanse the floor. The realms of Augmented Reality and Virtual Reality further amplify the importance of such understanding. Here, information and content tailored for the user's gaze need precise contextual alignment. Imagine the dissonance of seeing a virtual chair placed atop a real table; it belongs beside it. In mixed reality tools designed for field workers, the significance of correctly positioned information becomes even more paramount. When a worker's viewpoint gets cluttered by misplaced or irrelevant details, the application can range from being merely ineffective to downright dangerous. Hence, for these diverse applications, the crux lies in the machine's ability to understand its environment.

In this thesis, we delve deep into the intricacies of 3D scene understanding, focusing primarily on two pivotal questions. The initial query centers on the efficient acquisition of large-scale labels, vital for both training and evaluating 3D scene understanding techniques. The subsequent question dives into the realm of online semantic mapping of our surroundings. As agents traverse and engage with

their environment, their comprehension of it must continually evolve. Drawing parallels with our earlier discussions on 3D reconstruction, we posit that the majority of semantic mapping applications necessitate real-time, online processing. This is especially pertinent when considering their deployment on agents and devices actively navigating and mingling with their surroundings. Consequently, we enhance the methods formerly introduced, steering them towards online 3D scene comprehension. In this endeavor, we probe into the synergistic use of 2D and 3D data, harnessing the power of innovative neural network frameworks.

This part is structured as following. In the next chapter, we will provide all necessary background information that is required to understand this part's context. Then, we will discuss our work on improving datasets for 3D scene understanding without any human intervention. Finally, we will discuss our work on enabling online semantic 3D reconstruction using a spatio-temporal attention mechanism.

Background

8.1 Overview about Scene Understanding

As motivated above, understanding the world around us is crucial for many tasks. Thus, it has been one of the main goals to replicate this capability in machines since the advent of artificial intelligence and more specifically computer vision.

Scene understanding consists of different subtasks that address different levels on the hierarchy of understanding. While we mostly focus on 3D semantic segmentation in this thesis, we will give an initial review of the different levels and their evolution through the past three decades. We will start with the most high-level task of recognition and then move towards segmentation at the bottom of the hierarchy of understanding, which is the main focus of this thesis.

8.1.1 Recognition

Recognition is an overall term for scene understanding and its application in computer vision. Originally, it mainly involved classifying images and detecting known objects in images. The results were usually either per image labels or bounding boxes around the recognized objects. As mentioned above, recognition is a mid- to high-level understanding task. There were two main approaches in early works. Feature-based methods extract hand-crafted features from image regions and use subsequent matching [Lowe, 2004] or classification [Viola and Jones,

2001, Dalal and Triggs, 2005]. For these works, the main focus was on exploring different types of features and classifiers to improve the detection performance. With the advent of deep learning, this field also received a boost in performance. Methods leveraging neural networks, like the R-CNN line of work [Girshick et al., 2014, Girshick, 2015, Ren et al., 2015], improved the performance significantly. Alternatively, YOLO [Redmon et al., 2016] or SSD [Liu et al., 2016] networks are popular alternatives with a fast runtime and good detection performance that are used in many applications. In recent years, the performance increase, in both accuracy and runtime, can be mostly attributed to an increase in dataset size and various backbones such as more powerful transformer-based backbones [Li et al., 2022b] or decoders [Carion et al., 2020, Zhu et al., 2020] or more efficient backbones [Howard et al., 2017] While mid-level recognition is a powerful tool, in this thesis we are interested in low-level understanding namely segmentation. We will therefore go through the history of this application next

8.1.2 Image Segmentation

In parallel to the efforts in recognition, the problem of segmenting images into groups of pixels has been extensively explored. While understanding was limited as most methods did not attach a specific label to these groups, these methods were a pre-condition for later semantic segmentation. The first approaches to understanding images on a pixel level, were all based on the following principle. 1) Measure the affinity or similarity between pixels, 2) build a cost function from these similarities, 3) minimize the cost function to find a solution to the grouping the pixels into separate segments. The main focus of early works usually considered one or more of these three stages. In general, they can be separated into clustering-based and graph-based approaches. Clustering-based approaches [Tou and Gonzalez, 1974, Coleman and Andrews, 1979] extract features from images, or use the raw color data, and group these features into clusters using a similarity measure. One can differentiate between parametric algorithms such as k-means clustering [Tou and Gonzalez, 1974] as that require a-priori knowledge of the number of segments and non-parametric that do not require a fixed number of clusters with Mean Shift [Comaniciu and Meer, 2002] being the most prominent

member of this group. Alternatively, graph-based methods [Wu and Leahy, 1993, Shi and Malik, 2000, Felzenszwalb and Huttenlocher, 2004] approach the problem from a graph theoretic perspective. They build an undirected graph from the data, construct a cost function, and optimize the cost function to find an optimal solution to the segmentation problem. Most progress has been focused on proposing better cost functions that includes finding priors and regularization terms improving the results [Shi and Malik, 2000, Rother et al., 2004, Yu and Shi, 2004, Arbelaez et al., 2010]. Yet, all these methods only considered grouping low-level pixels into segments and these methods did not provide any understanding of the scene by classifying the objects into classes.

8.1.3 Semantic Segmentation

So far, image segmentation and image recognition have been treated as two distinct tasks. *I.e.*, grouping pixels of a segment into segments or recognize objects in an image. Semantic segmentation bridges this gap by assigning a category to each segment. Thus, we obtain a low-level understanding of the scene. The first approaches were based on techniques that were already used in recognition. Top-down approaches matched object templates to image regions and measure agreement [Borenstein and Ullman, 2002]. If an agreement is found, the segment is classified as the recognized object. Then, people started to integrate top-down approaches into bottom-up grouping methods. *I.e.*, they proposed ways to combine different priors in the cost function that connect the low-level grouping with higher-level understanding [Yu et al., 2002]. Then, the deep learning revolution happened and the entire field turned upside down. Almost overnight, the performance received a significant boost by neural networks. [Long et al., 2015] proposed to use fully convolutional networks for semantic segmentation. They showed that convolutional neural networks can be trained end-to-end with pixel supervision and their performance exceeds state-of-the-art. An impactful follow-up was U-Net [Ronneberger et al., 2015] that introduced skip connections and to date is the backbone of many applications in 2D and 3D. Afterwards, the performance has been mostly boosted by architectural changes that are oftentimes specifically adapted to a use-case such as autonomous driving

or indoor segmentation, or increased size of datasets for large scale training [Zhou et al., 2017]. *E.g.*, DeepLab [Chen et al., 2017a] proposed to leverage atrous convolutions in order to increase the receptive field and avoid too much signal downsampling for semantic segmentation. More recently, one major remaining problem has been addressed by coupling semantic segmentation with language. The main problem with semantic segmentation still is that most methods and datasets group the world in a fixed number of classes. While for very specific use cases this might be sufficient, the world has not a fixed number of object categories. Categories do change given the context and the environment your in. Moreover, semantic categories are oftentimes ambiguous and different people use different names for the same class (even when using the same language). Hence, works like [Ghiasi et al., 2022, Li et al., 2022a, Liang et al., 2023] coupled large language models with semantic segmentation models to couple the tasks and address this challenge. Alternatively, InternImage [Wang et al., 2022b] draws inspiration from recent large-scale vision models [Dosovitskiy et al., 2020] and trains large-scale CNNs for the task of semantic segmentation showing the benefit of large-scale pre-training.

8.1.4 3D Semantic Segmentation

However, in this work we are mostly interested in 3D scene understanding. In this task instead of grouping pixels, we group voxels, points, vertices or other geometric primitives into segments and classify them. With the rise of 3D datasets, this task became more and more popular in the community. The first approaches mostly focused building a 3D cost function and find its optimum [Vineet et al., 2015, Valentin et al., 2013, Häne et al., 2013, Cherabier et al., 2016] or map 2D predictions into 3D [Sengupta et al., 2013]. However, techniques that learn the task from data have been proven to be more effective. This required the creation of semantically annotated 3D datasets, which we will discuss in the next section. Once these datasets have been established, algorithmic advances could be made. Initial progress was made on point clouds and voxels as they offered ease of processing with neural networks by either using linear layers [Qi et al., 2017a, Qi et al., 2017b] or 3D convolutions [Dai and Nießner, 2018]. One major

breakthrough was the proposal of sparse convolutions for 3D processing [Graham et al., 2018, Choy et al., 2019a]. This significantly improved efficiency in both training and inference. Moreover, a significant improvement was made by boosting data augmentation during training [Nekrasov et al., 2021]. Most recently, 3D semantic segmentation got coupled with large language models [Peng et al., 2023, Rozenberszki et al., 2022].

8.2 Datasets for Scene Understanding

So far, we have ignored one key ingredient for the rapid advance of the field in the last decade. Datasets for scene understanding have been crucial since the inception of the field. While in the very early days it was sufficient to show that a proposed method works on a few images, systematic evaluation on benchmarks was a main driver for progress in the last decades. One of the first successful attempts to establish a benchmark was [Everingham et al., 2010]. This allowed to quantitatively compare different methods on a variety of images advancing the progress in the field. With the advent of machine learning in the field, datasets became not only essential for evaluation but also for training models that understand the world. This led to a huge increase of the number of datasets available for training and evaluation.

8.2.1 2D Datasets

Early datasets were all 2D due to a lack of 3D sensing technology. The first datasets mainly consisted of images containing a single object [Fei-Fei et al., 2006, Griffin et al., 2007]. This was ultimately scaled by ImageNet [Deng et al., 2009] the data backbone of many neural networks to date. For a long time, most datasets were designed for a specific use-case or application. LabelMe [Russell et al., 2008] provided in addition to images annotated with bounding polygons a convenient web-based labeling tool. As already mentioned, one of the first standard benchmark datasets for object detection was [Everingham et al., 2010]. It provides a dataset and a corresponding benchmark for originally 500k images with 20 annotated object classes. The next step in scale and annotation quality

was Microsoft COCO [Lin et al., 2014]. It provides one of the first datasets that tackle the challenge of semantic instance segmentation. Further, there are several more specific datasets for different domains in the last years. Cityscapes [Cordts et al., 2016] is one of the most established 2D semantic segmentation datasets focusing on autonomous driving. Similar frame-by-frame manual annotations were provided in NYU Depth [Silberman et al., 2012], ADE20k [Zhou et al., 2017], or COCO-stuff [Caesar et al., 2018]. While frame-by-frame annotations yield very high quality segmentation masks, they are expensive to obtain. Although the effort can be reduced through comfortable annotation tools [labelme github contributors, , Bréhéret, 2017], it cannot be avoided that a human inspects every image and performs at least a couple of clicks.

8.2.2 3D Datasets

With the commodization of 3D sensing technology 3D scene understanding datasets came into reach. Early 3D datasets for scene understanding [Nathan Silberman and Fergus, 2012, Song et al., 2015, Xiao et al., 2013] provided RGB-D sequences with semantic annotations. Yet, their scale was limited, the camera poses not available for all frames, and the full 3D reconstruction and dense annotations were not always provided. The first large 3D scene understanding datasets got proposed by [Hua et al., 2016], [Dai et al., 2017a], and [Armeni et al., 2016]. [Armeni et al., 2016] provides 3D pointclouds together with semantic annotations. [Dai et al., 2017a] and [Hua et al., 2016] also provide annotated 3D meshes that are associated with RGB-D scans. They additionally provide instance annotations and established a popular benchmark for 2D and 3D semantic instance segmentation. More recently, with the integration of 3D sensing technology in consumer products, more datasets have been collected that are closer to real-world applications. [Johanna Wald, 2019] offers semantic instance labels in changing environments, *i.e.*, they re-scan the same room after some time to capture dynamic changes. ARKitScenes [Baruch et al., 2021b] is a large-scale indoor datasets for different tasks. It leverages the ARKit [Apple, 2017] framework provided in Apple products to get odometry and RGB-D data. This data is accompanied with high-resolution laser scans. While there are some annotations, they are sparse

and not complete. Thus, it does not fulfill its complete potential. An even newer indoor dataset is MultiScan [Mao et al., 2022] that not only annotated 3D scans with posed RGB-D trajectories but also provides scene dynamics.

An entire line of datasets [Chang et al., 2017, Xia et al., 2018, Straub et al., 2019] has been integrated into the Habitat AI environment [Savva et al., 2019]. They all offer easy access semantic object annotations. Habitat AI can be used to render trajectories from these datasets. Nevertheless, the quality of the generated data often falls short in comparison to scanned datasets, primarily due to underlying issues in the scanning methodology and mesh reconstruction.

8.2.3 Annotating Datasets for Scene Understanding

If scenes are annotated in 3D, their annotations can easily be rendered into any localized camera image in the same scene, therefore potentially reducing 2D labeling effort. This approach was followed in Replica [Straub et al., 2019] and ScanNet [Dai et al., 2017a]. iLabel [Zhi et al., 2021b] pioneered to use neural radiance fields (NeRFs) for this type of rendering, additionally showing that NeRFs have an intrinsic capability to segment whole objects along texture boundaries from a few clicks. Similarly, [Kontogianni et al., 2023] also reduces the manual labeling effort to a few positive and negative clicks per object. Matterport [Chang et al., 2017] consists of large labeled 3D scans, but does not have corresponding 2D images and therefore can only be used for 3D methods.

Yet, to leverage the full power of large neural networks for 3D scene understanding, large-scale 3D datasets with dense annotations are required. These annotations are costly to obtain if human labor is needed and oftentimes suffer from issues in label quality. Hence, we present a fully automatic labeling pipeline for RGB-D trajectories to generate both 2D and 3D labels of scenes for training and evaluation in chapter 9 of this thesis.

8.3 3D Semantic Segmentation

Given the focus of this thesis, we review 3D scene understanding in more detail and pay special attention to online processing for 3D scene understanding

as - similarly do 3D reconstruction - agents usually consume a stream of data for this task.

8.3.1 Offline vs. Online Processing.

3D semantic segmentation can be split into two separate design principles. The first principle follows an *offline* design while the second follows an *online* design.

Offline Methods. Offline methods require an a-priori reconstruction of the scene. This model can be reconstructed from images or 3D measurements and be stored in one of the common 3D representations such as point clouds, triangle meshes, or voxel grids. Moreover, the color and sometimes surface normals are usually required as input to the offline model. Hence, these features are stored in the points, voxels, or vertices. The offline method takes the model as its input and predicts a semantic label for each point, voxel, or vertex. A benefit of offline methods is their large receptive field that is advantageous for semantic segmentation as it helps to incorporate high-level context into the labeling decision. Yet, this comes at the cost of increased runtime and memory footprint, which make offline methods difficult to directly apply to real-time applications on mobile devices.

Online methods. In contrast to offline methods, online methods do not require an a-priori reconstruction of the scene. They jointly reconstruct the geometric and semantic map using an online mapping mechanism such as TSDF fusion [Curless and Levoy, 1996a, Newcombe et al., 2011a], RoutedFusion, or NeuralFusion. They consume RGB(-D) measurements and take these as an input to predict semantic labels for every point in the scene. Many methods predict semantic labels in 2D and map these into 3D, where they are potentially refined. We will review these methods in the next section. While the online design is adapted to real-time applications, it is challenging to incorporate required context for semantic segmentation available in offline methods.

We first review prior work on *offline* semantic segmentation and then look at existing *online* methods in the context of incrementally building semantic 3D maps.

8.3.2 Offline 3D Semantic Segmentation

Offline 3D semantic segmentation is the problem of assigning a class label to each point, voxel, or vertex given a reconstruction of the 3D scene. It is central to many applications and pipelines that require some form of understanding. In recent years, many different methods tackled this problem. Semantic Stixels [Schneider et al., 2016] predict 2D semantic labeling and stereo depth maps that are aggregated in a 3D stixel representation. While this representation can be sufficient for outdoor applications, it lacks representation power for indoor applications. [Graham et al., 2018] proposed sparse sub-manifold convolutions to improve efficiency for 3D semantic segmentation. This has been extended by [Choy et al., 2019b] into a complete framework for sparse 3D neural networks and it has been established as the backbone of many 3D semantic segmentation methods to date. Kundu *et al.* [Kundu et al., 2020] address the problem of lack of context in the 2D views by rendering views from an already reconstructed mesh to have a larger field-of-view that improves the performance of 2D semantic segmentation. The predictions are afterwards aggregated again on the 3D mesh. As this approach is dependent on an already reconstructed mesh, it is not suitable for an online fusion approach. Atlas [Murez et al., 2020] jointly reconstructs a semantic and geometric map from visual inputs by learning multi-view fusion. As this approach needs to aggregate dense viewing frustums to solve the multi-view stereo problem, it is not suitable for fast online updates. SemanticNeRF [Zhi et al., 2021b] proposes the application of recently proposed neural radiance fields [Mildenhall et al., 2020b] to the problem of 3D semantic segmentation. While this approach shows impressive results, it is also not applicable to fast and accurate online updates of a semantic map. Mix3D [Nekrasov et al., 2021] boosts the performance of 3D segmentation methods by proposing a novel data augmentation technique that combines different scenes. While this augmentation works for global methods, it cannot be applied to online fusion systems since we jointly learn the fusion across time and segmentation of the scene. BpNet [Hu et al., 2021a] couples 2D and 3D predictions of scene labels by proposing a bi-directional projection module. This boosts the performance on 3D semantic segmentation but is dependent on global processing and a-priori scene reconstructions. VMNet [Hu et al., 2021b] combines

Euclidean and geodesic information to address the short-comings of voxel-only approaches. Yet, it also requires global processing to unfold its full potential. OccuSeg [Han et al., 2020] enhances supervoxel-based geometric segmentation with learned features and refines them using graph-based clustering but requires a global receptive field, which makes them unnecessarily expensive for online processing.

8.3.3 Online 3D Semantic Segmentation.

In contrast to the previously mentioned offline approaches, online methods iteratively process the scene making them better suitable to real-time applications, where agents are interacting with their environment such as robotics or mixed reality in unknown environments. As discussed before, there is a long line of work aiming at the real-time reconstruction of geometry and appearance such as [Curless and Levoy, 1996a, Newcombe et al., 2011a], RoutedFusion, and NeuralFusion. These works have been extended to scene understanding to enable agents with understanding capabilities. The approaches [Vineet et al., 2015] and [McCormac et al., 2017] proposed to fuse 2D semantic predictions into a global semantic map that is refined using a conditional random field (CRF). This idea has been extended by several works. SceneCode [Zhi et al., 2019] stores a per-keyframe latent code encoding the semantic information of the scene that is optimized at test time. Meanwhile, MaskFusion [Rünz et al., 2018] and Fusion++ [McCormac et al., 2018] focus on 3D object segmentation while ignoring their semantic class. ProgressiveFusion [Pham et al., 2019] improves efficiency by clustering voxels into supervoxels and apply a CRF on that level. SemanticReconstruction [Jeon et al., 2018] follows a similar approach as [McCormac et al., 2017], but shows that their scene representation can be used for downstream tasks such as scene completion and manipulation. PanopticFusion [Narita et al., 2019] estimates 3D semantic instance maps by predicting 2D semantic and instance segmentation using off-the-shelf networks, aggregates them in 3D, and also regularizes them using a CRF. While these works leverage 2D processing in combination with optimization-based 3D regularization, they all resort to traditional voxel fusion and do not utilize trainable 3D neural networks. This shortcoming has been

addressed in SVCNN [Huang et al., 2021b], which clusters voxels that store explicit semantic information into supervoxels, and then processes them using a special convolutional operator designed for supervoxels. However, [Huang et al., 2021b] still resorts to an explicit fusion of 2D semantic information into voxels. An alternative is FusionAware [Zhang et al., 2020] that represents scenes using efficient point cloud representations and uses point-convolutions to aggregate new information. More recently, [Liu et al., 2022b] presented an online method predicting semantic instance maps using only 3D processing. Nevertheless, these two works disregard useful 2D information.

In chapter 10 of this thesis, we address these limitations by a) combining 2D and 3D information in a temporal expert network leveraging both sources of information, and b) applying a powerful yet lightweight 3D network on the current viewing frustum.

Automatic Annotation for 3D Semantic Segmentation

As we have learned above, semantic perception is a central element of many applications that interact with the world around us. Without semantic perception, meaningful interaction with our environment is hardly possible. Thus, semantic scene perception has been a long-standing problem in computer vision and robotics. As reviewed above, most solutions have converged towards using deep neural networks in recent years. However, training and evaluating these networks is hard. As recent works such as SAM [Kirillov et al., 2023], language-based models [Liang et al., 2023, Ghiasi et al., 2021], or InternImage [Wang et al., 2022b] have shown, huge quantities of training data, orders of magnitude larger than any single existing research dataset, are necessary to achieve good generalization. On the other hand, generalization is necessary because the distribution of the deployment environment - e.g. a particular user's home, in which a robotic application is to be deployed - is outside of the distribution of existing annotated training datasets. To evaluate generalization in or adapt to specific deployment environments, labeled data of these environments is necessary. From both training and deployment perspectives, the availability of labeled data is therefore a key problem. Unfortunately, the acquisition of this data is usually very expensive as semantic groundtruth annotation is a time-consuming manual process. Therefore, we propose a fully automatic method to obtain annotations for 3D scene

understanding in this chapter.

In particular, we focus on 3D semantic segmentation. The available scale of 3D semantic segmentation data such as ScanNet [Dai et al., 2017a] or Matterport3D [Chang et al., 2017] is far below the scale of 2D semantic segmentation datasets like ADE20k [Zhou et al., 2017], COCO-stuff [Caesar et al., 2018], or others [Silberman et al., 2012, Yu et al., 2020, Cordts et al., 2016]. Even tough tasks such as semantic segmentation or online semantic reconstruction gain maturity and are crucial for interactive applications, there is even less semantic data with paired camera trajectories and corresponding scene reconstructions. ScanNet [Dai et al., 2017a] is by far the largest in this domain with an abundance of scenes and a well-established benchmark. However, both camera images and labels are oftentimes noisy, making it hard to generalize from ScanNet to other datasets. ARKitScenes [Baruch et al., 2021b] shows the growing possibility to capture RGB-D trajectories at scale, and at the same time illustrates the cost of semantic annotations. It provides RGB-D scans of several thousand rooms paired with high-resolution laser scans providing accurate 3D geometry. Yet, the annotations are extremely sparse. Instead of dense semantic annotations, ARKitScenes only provides an incomplete list of bounding boxes.

To push the scale and accuracy of 3D semantic segmentation datasets, we present *LabelMaker* in this chapter. *LabelMaker* automatically creates labels that are on the same level of accuracy as the established ScanNet benchmark, but without any human annotation. Further, we show that it can produce better labels than the original ScanNet labels when using the human annotations as an additional input.

The design of our method is motivated by two observations. The first observation is on recent advances in 2D semantic segmentation, where a leap in training data scale through combination of different tasks and datasets [Wang et al., 2022b] or visual-language models [Liang et al., 2023] has boosted generalization. The second observation is in the field of neural radiance fields, where [Zhi et al., 2021b, Liu et al., 2023, Siddiqui et al., 2023] have shown that NeRFs can be used to denoise semantic input labels and learn a multi-view consistent semantic label field.

We leverage these two observations and motivate an automatic labeling pipeline with two main components at its heart. First, we leverage large 2D models, that combine the power of different tasks and input modalities, in order to predict different hypothesis for labels in 2D. These labels are aggregated using our consensus voting mechanism in order to obtain a single 2D prediction for every frame. Second, all 2D predictions are aggregated and made consistent using a neural radiance field. This neural radiance field can be used to render clean and consistent 2D label maps. Alternatively, the labels can be aggregated and mapped into 3D to obtain labeled point clouds or meshes.

In an extensive comparison to state-of-the-art methods and datasets and an detailed ablation studies, we showcase that our method automatically generates labels of similar quality than human annotators. We also demonstrate fully automatic labeling for ARKitScenes, for which no dense labels exist to date.

In summary, our main contributions in this chapter are the following:

- A curated mapping between the established indoor label sets NYU40, ADE20k, ScanNet, Replica, as well as a mapping into the wordnet graph.
- A pipeline to automatically label trajectories of RGB-D sensors, as well as corresponding 3D point clouds, that achieves higher quality than the original labels of ScanNet.
- Generated labels in 3D meshes and 2D images for ScanNet [Dai et al., 2017a] and ARKitScenes [Baruch et al., 2021b].

9.1 Method

We present an automatic labeling pipeline for RGB-D trajectories. This requires obtaining automated labels using strong pre-trained models, aggregated them in a unified label space, and fuse them into a globally consistent 3D model.

9.1.1 Base Models

Our aim is to build a fully automatic labeling pipeline to generate semantic annotations for RGB-D trajectories. To this end, we somehow need to obtain

automatic semantic labels. Therefore, we employ an ensemble of strong base models, each state-of-the-art in their respective task and data characteristic. We use these models to predict a 2D semantic segmentation for each frame in the trajectory.

InternImage [Wang et al., 2022b]. is a supervised 2D RGB-only semantic segmentation model that at the time of writing has state-of-the-art performance on the Cityscapes and ADE20k benchmarks. It proposes the use of dynamic sparse kernels, deformable convolutions, to model long-range dependencies and adaptively aggregate spatial information while still being compute and memory efficient. Using this underlying operator, the overall network size is scaled to one billion parameters. To fully leverage this capacity, the overall network is pre-trained on 427 million images obtained from combining Laion-400M [Schuhmann et al., 2021], YFCC-15M [Thomee et al., 2016], and CC12M [Changpinyo et al., 2021] and fine-tuned on ADE20K [Zhou et al., 2017]. In this work, we use the ADE20k fine-tuned variant.

OVSeg [Liang et al., 2023]. is an open-vocabulary semantic segmentation model based on CLIP [Radford et al., 2021], a visual-language representation model. OVSeg segments images by assigning region proposals to a set of given prompts and is therefore not limited to a fixed set of classes. In particular, we added such an open-vocabulary segmentation model not because they achieve the best performance on a given task but because of their generalization ability. We generate prompts from our set of wordnet synkeys discussed in the next section by averaging over language prompts such as “A _ in a room.”, but also using all possible synonyms according to wordnet.

CMX [Zhang et al., 2023b]. is at the time of writing the state-of-the-art 2D semantic segmentation model for NYU Depth v2, a RGB-D indoor dataset. CMX unify multi-model semantic segmentation in a single framework. To this end, the propose a cross-modal feature rectification module to better utilize the complementary information in different models and a feature fusion module combining the information extracted from the different modalities. In our work, we use RGB and depth map as the two input modalities.

Mask3D [Schult et al., 2023]. is at the time of writing the state-of-the-art 3D

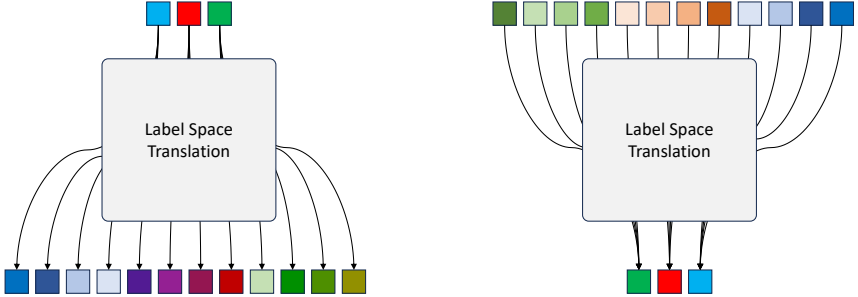


Figure 9.1: Label space translation. We created a curated mapping that allows to translate each label space (*e.g.* NYU40, ADE20K, ScanNet200) to every other translation space. While translating from a high-resolution label space to a low-resolution space is straightforward, the challenge lies in translating from the low-resolution to high-resolution.

instance segmentation model on ScanNet200 [Rozenberszki et al., 2022]. This method operates on an accumulated point cloud of a scene instead of frames, therefore taking even better the geometry into account. We use the publicly released weights trained on ScanNet200. We render the 3D semantic instance predictions into the 2D training frames to map them into the same space as all other base models.

The four semantic models produce classifications in four different sets of classes. InternImage predicts 150 ADE20k classes, CMX predicts 40 NYU classes, Mask3D predicts 200 ScanNet classes, and our OVSeg prompts cover 186 wordnet classes. In addition to the semantic models, we use OmniData [Eftekhari et al., 2021] to complement the depth sensor.

9.1.2 Translation between Label Spaces

As we learned above, we employ different models that were trained on different datasets with different numbers and definitions of classes. To use these predictions in an ensemble, we need to merge them in a unified prediction space. This requires translating between different prediction spaces. We therefore build a mapping between the class definitions of NYU40, ADE20k, ScanNet20, ScanNet200, Replica, and the WordNet semantic language graph.

In this effort, we build on top of previous work, as the original ScanNet [Dai et al., 2017a] already defined a mapping between ScanNet classes, NYU40

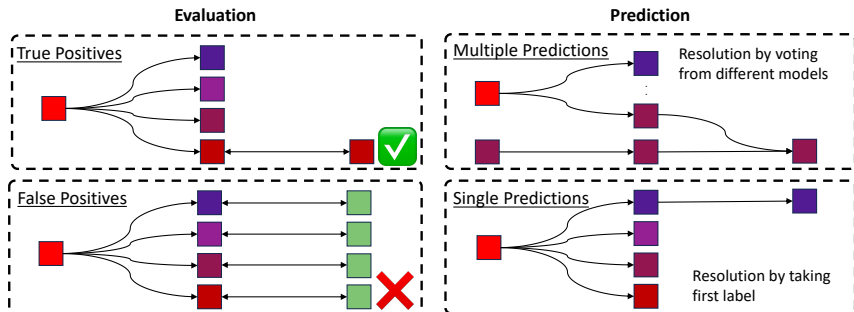


Figure 9.2: Resolving the translation from low-resolution to high-resolution label space. In evaluation, if the one of the translated classes is the groundtruth class it is counted as true positive. If none of the translated classes is the groundtruth, all of them are counted as false positive. During prediction, the conflict is either resolved through voting (multiple predictions) or taking the first label (single prediction).

[Nathan Silberman and Fergus, 2012] classes, Eigen13 [Eigen and Fergus, 2015] classes, and wordnet synkeys [Miller, 1995]. Further, [Lambert et al., 2020] curated mappings between the taxonomies of semantic segmentation datasets, out of which mappings between NYU40, SUN RGB-D [Song et al., 2015], and ADE20k are most relevant for indoor perception. We took the union of both works as initial mapping, but found that many corrections were needed especially with regard to wordnet synkeys and many ADE20k were missing because [Lambert et al., 2020] only considered 20 NYU categories. Thus, we also added mappings to the Replica categories for the purpose of evaluation, since Replica is one of the most accurately annotated indoor semantic datasets.

When mapping between two label spaces, for any class in the source space there are three cases in the target space: *a*) there is no corresponding class in the target space, *b*) there is exactly one corresponding class in the target space. This may be an exact match, or a class to which multiple class ids from the source space are matched (e.g. the source space may distinguish between office chair, chair, and stool but the target space just has one general chair class), *c*) there are multiple corresponding classes in the target space because the target space has a higher resolution than the source space (e.g. a general chair class in the source space can be split up in the target space to distinguish between office chair, chair, or stool).

For (a) and (b), mappings are straightforward. We resolve (c) dependent on the use cases:

- *During evaluation:* If one of the classes after translation corresponds to the groundtruth label it is counted as a true positive. If none of the translated classes is the true label, all of them are counted as false positives.
- *During model consensus computation:* Predictions in the source space vote for all possible correspondences in the target space. The ambiguity between the possible correspondences is usually resolved through an additional predictor with a prediction space of higher resolution. If no resolution is achieved, we pick the first of the possible classes.

9.1.3 Model Consensus

Given the individual predictions of the base models that have been translated into the unified label space using the mappings described above, we merge the predictions into a single consensus prediction. In addition to the straight-forward base model predictions, we also exploit test time augmentation to make the consensus more robust. Therefore, we use left-right flipping, which means we horizontally flip the image before passing it through the network and flip the result back again. Finally, each pixel receives votes for possible classes from:

- the predictions of the standard RGB images of all 2D segmentation models InternImage, CMX, and OVSeg.
- the predictions of the flipped version of the RGB images for the 2D segmentation models.
- two votes (to equalize the test-time augmentation of the RGB frame) from the Mask3D prediction rendered into the 2D frame
- if we have human annotations available and also use them, we take five votes from the original labels

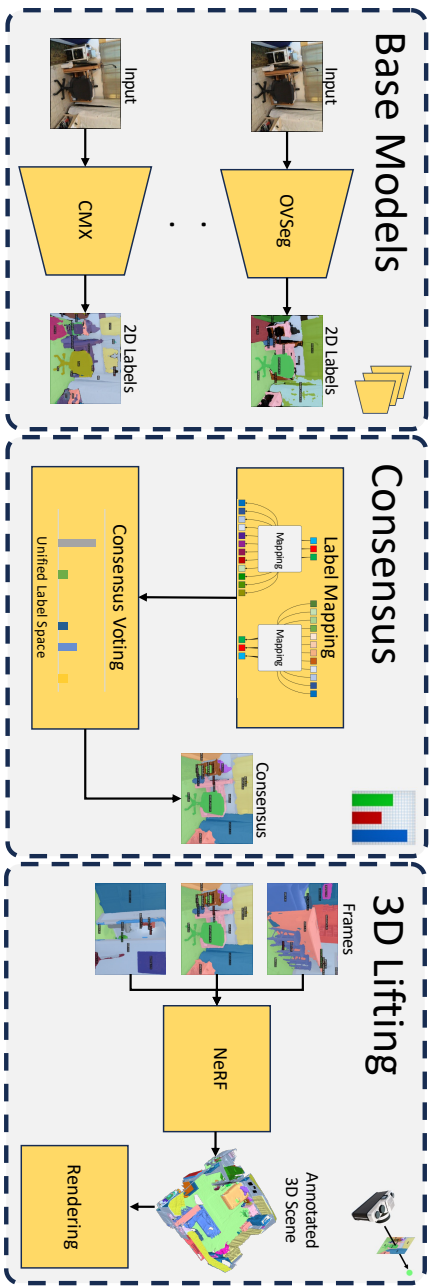


Figure 9.3: LabelMaker pipeline overview. The LabelMaker pipeline consists of three main blocks. In the prediction block, state-of-the-art 2D and 3D models predict per-frame semantic maps. These predictions are aggregated in the aggregation block. To aggregate the predictions of different models, we need to first translate the different label spaces into our unified label space, before we run the consensus voting to obtain the final per-frame consensus. In the 3D lifting block, the per-frame consensus is lifted into 3D using a neural radiance fields. This further denoises the predictions and makes the labeling multi-view consistent. From this representation, we can finally render the final annotations back into the individual frames.

For every pixel, we choose the class with the maximum number of votes. For additional robustness, we check if the maximum number of votes is above a certain threshold. If it is not, the prediction is set to 'unknown' and this particular pixel is not used in the optimization during 3D lifting.

9.1.4 3D Lifting

By computing a consensus over a diverse set of 2D predictors, we leverage the knowledge and scale of 2D semantic segmentation datasets. However, the per-frame predictions are noisy and often inconsistent, especially around image boundaries. These inconsistencies can be mitigated and the performance can even be improved, as previous work has shown [Siddiqui et al., 2023, Liu et al., 2023], by lifting the 2D predictions into 3D.

Therefore, we leverage the recent progress on neural radiance fields to generate multi-view consistent 2D semantic segmentation labels in all frames. Previous works [Siddiqui et al., 2023, Liu et al., 2023] observed that neural radiance fields hallucinate geometry to explain inconsistencies in different 2D predictions. Yet, if these hallucinations are avoided and accurate geometry is enforced, the inconsistencies between the frames can be resolved. There, we train an implicit surface model from sdfstudio [Yu et al., 2022] that has a more explicit surface definition compared to a NeRF yielding improved geometry compared to vanilla NeRF. Thus, we add a semantic head to the Neus-Acc model, train it on all available views and supervise it with RGB, depth, normals, and our semantic losses. The RGB and depth are directly used from the sensor while the normals are estimated using omnidata [Eftekhari et al., 2021] and the semantic segmentation is obtained from the previously described consensus voting. Finally, we render the optimized semantics back into all camera frames.

To generate consistent 3D semantic segmentation labels, we follow an established and more direct approach. Given a point cloud of the scene, we project the point cloud into each consensus frame to find corresponding pixels and then take a majority vote over all pixels corresponding to a point. This approach has proved to be as effective yet much cheaper than the NeRF-based approach to obtain the 3D labels.

9.1.5 Relabeling ScanNet Scenes

To evaluate the quality of LabelMaker, we want to compare it against existing human annotations. We choose the ScanNet dataset because its scale has a large potential for automatic processing. To be able to evaluate the quality of the existing labels and compare them with LabelMaker, we create high-quality annotations for a selection of scenes.

The original ScanNet [Dai et al., 2017a] labels were created using free text user prompts. They consequently have duplicates or are ill-defined. This reflects the open-world approach of [Dai et al., 2017a], but contradicts the use as benchmark labels, for which they map them to other class sets. As a set of annotation classes, we therefore did not directly annotate with ScanNet classes, but use wordnet [Miller, 1995] synkeys¹. In particular, we start from the mapping that ScanNet defined between their labels and wordnet and take the categories that occur at least three times in the dataset. This yields an initial list of 199 categories, already resolving many ambiguities. We then check the definitions of all of these categories in the wordnet database and correct the initial mapping, as well as merged categories that are still too ambiguous by their definitions in wordnet (e.g. `rug.n.01` “rug, carpet, carpeting; floor covering consisting of a piece of thick heavy fabric (usually with nap or pile)” and `mat.n.01` “a thick flat pad used as a floor covering”). The result are 186 categories that come with a text definition, a defined hierarchy, and all possible synonyms that describe the category.

We then annotate the selected ScanNet scenes with these 186 categories based on their wordnet definitions. We use [Kontogianni et al., 2023] to annotate the fine meshes of the scenes with a minimum number of necessary clicks. Only the authors of this paper provided annotations, and each annotation was cross-checked by at least one other author. In case of doubt, individual objects were discussed together. On average, labeling of a scene took 5 hours.

¹Wordnet is a dictionary and synkeys are the names of its entries. I.e., a set of synonymous words has one synkey, but a word with different meanings as one synkey per definition.

9.2 Experiments

9.2.1 Implementation Details

For the 2D models, we use the corresponding available open-source code and adjust it to our pipeline. As described in Section 9.1.2, we generate votes from each 2D model into a common label space. We choose our defined 186 class wordnet label space as output. We choose the label with highest votes, but require a minimum of 3 out of 13 (with ScanNet annotations) resp. 4 out of 8 (automatic pipeline) votes. For 3D optimization, we build on top of SDFStudio [Yu et al., 2022], specifically the Neus-Acc [Wang et al., 2021] model, and add a semantic head and semantic rendering similar to [Zhi et al., 2021b].

9.2.2 Datasets

We run our proposed method on three different datasets to show its performance and validate our design choices.

ScanNet [Dai et al., 2017a]. We randomly select 5 scenes from the ScanNet that cover all frequent room types. We carefully annotate high-resolution meshes of the scenes using [Kontogianni et al., 2023] as described in Section 9.1.5 in order to have a complete and accurate groundtruth to evaluate against.

Replica [Straub et al., 2019]. We also evaluate our method on the Replica dataset. This is a semi-synthetic dataset, captured as a high accuracy mesh from real environments and then rendered into trajectories in [Zhi et al., 2021b]. We select the 3 ‘room’ scenes and evaluate against the given annotation.

ARKitScenes [Baruch et al., 2021b]. To showcase the automatic labeling pipeline on an existing dataset, we run it on selected scenes of the ARKitScenes dataset, where only sparse bounding box labels are available up to date. ARKitScenes consists of trajectories captured with consumer smartphones which are registered to a professional 3D scanner.

evaluation classes	2D						3D					
	NYU (40 classes)			wordnet (186 classes)			NYU (40 classes)			wordnet (186 classes)		
	mIoU	mAcc	tAcc	mIoU	mAcc	tAcc	mIoU	mAcc	tAcc	mIoU	mAcc	tAcc
ScanNet labels [Dai et al., 2017a]	47.7	56.2	69.2	38.1	46.3	69.7	40.1	48.2	68.6	17.7	21.3	70.6
SemanticNerf* [Zhi et al., 2021b]	45.2	56.6	69.3	32.9	43.7	71.2	36.7	47.1	68.4	14.8	19.3	71.0
LabelMaker w/o ScanNet (automatic labels)	50.7	64.0	75.3	33.5	43.5	72.3	41.3	47.3	71.2	15.7	18.1	71.5
LabelMaker (Ours)	53.4	65.0	77.5	39.1	49.3	77.2	44.1	53.4	76.1	18.2	22.0	76.7

Table 9.1: Comparison of the label quality of the ScanNet labels. We compare with the original ScanNet labels and the original ScanNet labels refined by SemanticNerf [Zhi et al., 2021b]. Further, we compare LabelMaker without any human input, and LabelMaker taking the ScanNet annotations as additional input. The results are measured over 5 scenes from ScanNet against newly annotated high-quality ground truth. Based on our translation of prediction spaces, we measure metrics over the medium-tail NYU40 set of categories and our full long-tail ground truth categories. For NYU40 classes, LabelMaker is capable of producing labels of higher quality than the ScanNet human annotations, without any human input. For more long-tail categories, the automatic mode does not reach the quality of ScanNet, but LabelMaker is able to considerably improve human annotations.

9.2.3 Baselines

We mainly compare LabelMaker to the existing manually created annotations in ScanNet [Dai et al., 2017a]. As an additional baseline, we report the result of fitting and rendering the ScanNet annotations with our adapted SemanticNeRF [Zhi et al., 2021b]. We briefly discuss these two main baselines in the following.

ScanNet [Dai et al., 2017a]. For this baseline, we measure the quality of the annotations in ScanNet. This baseline is motivated by the fact that datasets that have been annotated using some form of crowdsourcing oftentimes suffer from noisy groundtruth annotations. Therefore, we aim to quantify this issue in case of ScanNet. To this end, we take the raw ScanNet labels and map them into our label space defined by wordnet. The mapping from ScanNet IDs to wordnet synkeys is to a large extent already provided in [Dai et al., 2017a].

SemanticNeRF [Zhi et al., 2021b]. This baseline is inspired by [Zhi et al., 2021b] and adapted to our pipeline by integrating the semantic head into SDFStudio. Then, we run this version of SemanticNeRF on the ScanNet 2D semantic labels. Thus, we can measure the effect of multi-view aggregation and optimization on the groundtruth ScanNet labels. The hypothesised effect is that through the extra RGB and geometry information provided to the NeRF, segmentation boundaries may be smoother than those of the ScanNet ‘supervoxels’.

9.2.4 Comparison to State-of-the-Art

In Table 9.1, we compare LabelMaker to the state-of-the-art baselines ScanNet and SemanticNeRF. We report mean intersection-over-union (mIoU), mean accuracy (mAcc), as well as total accuracy (tAcc). We evaluate the methods in 2D by comparing the renderings or labeled frames with renderings from the groundtruth 3D mesh and in 3D by mapping the 2D renderings onto the corresponding vertices in the 3D groundtruth mesh. Further, we measure the metrics over two different label sets. The NYU40 label set [Silberman et al., 2012] consists of 40 semantic classes representing the common indoor classes in the short tail of the label distribution. Our curated wordnet label set consists of 186 classes, therefore additionally measuring performance over the long tail of the label distribution.

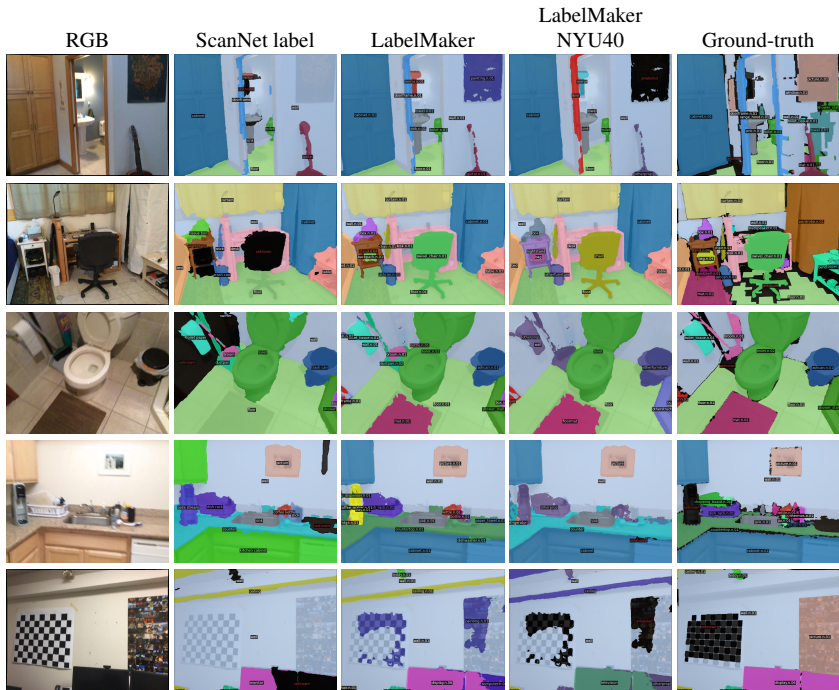


Figure 9.4: LabelMaker generates more accurate and more complete labels compared to the labels annotated by humans and provided by ScanNet. Particularly, unlabeled sections in ScanNet are correctly filled in and many wrong annotations such as missing rogs and pictures are corrected. The output labels can then be projected into different label spaces, such as our wordnet space or the NYU40 categories.



Figure 9.5: Dense 3D labels for ScanNetv2 [Dai et al., 2017a]. We generate more consistent labels compared to human annotators and preserve rare classes (e.g. swivel chair in front of the desk). Further, the labels are more complete (e.g. wall in bathroom) and we can capture all object in the scene (e.g. dustpan in bathroom).

	ScanNet (186 classes)			Replica (150 classes)		
	mIoU	mAcc	tAcc	mIoU	mAcc	tAcc
OVSeg	15.3	24.4	43.7	20.7	26.5	69.4
InternImage	30.8	43.5	59.4	38.3	47.7	84.6
CMX	28.2	41.0	54.2	17.0	38.0	84.6
Mask3D	33.7	40.2	38.5	22.6	27.9	30.4
Consensus	38.9	48.3	77.0	39.1	46.2	84.3
LabelMaker (ours)	39.1	49.3	77.2	42.1	51.0	86.7

Table 9.2: Ablation of all base models in LabelMaker on our 5 labelled ScanNet [Dai et al., 2017a] scenes and Replica [Straub et al., 2019]. InternImage is the strongest single base model, but the fusion with other predictions and 3D lifting increases the accuracy considerably beyond any of the state-of-the-art single models.

We demonstrate that our proposed pipeline generates better labels than both human-annotated ScanNet labels as well as their lifted version through SemanticNeRF [Zhi et al., 2021b]. Particularly, on the short tail of the distribution (NYU label set), our pipeline significantly improves over the human annotated labels. This is due to more accurate object boundaries as well as more consistent and complete labels. For the long tail of the label distribution, our method also outperforms all existing baselines indicating that different 2D expert votes and 3D aggregation boosts the quality of the annotated labels. Finally, we show that even our fully automatic pipeline outperforms human annotations on NYU40 classes, showing the potential of LabelMaker to generate labels at scale.

Qualitative comparison with ScanNet [Dai et al., 2017a]. In Figure 9.5, we compare qualitative results for ScanNet [Dai et al., 2017a] with LabelMaker, and our groundtruth. To this end, we mapped the 2D renderings onto the high-resolution groundtruth mesh by projecting the mesh vertices into all labels using a visibility check. One can see that our pipeline produces consistently more complete and correct labels than the human annotations provided by ScanNet [Dai et al., 2017a]. *E.g.*, our method consistently labels the kitchen counter top, the mats in the bathroom, and even the folded chair leaned against the desk.

ScanNet Label Quality. Because our experiments require new high-accuracy annotations of ScanNet scenes, we are able to estimate the quality of the default ScanNet labels. As Table 9.1 shows, but also any human who inspects the ScanNet

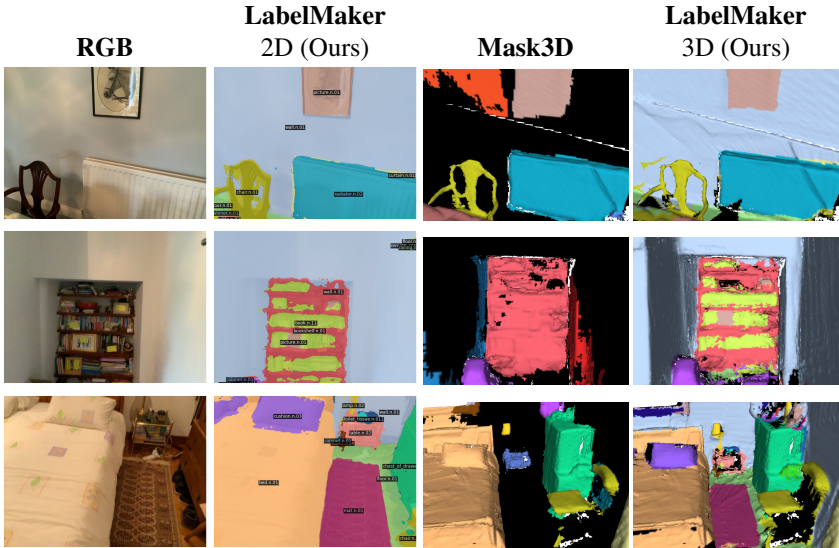


Figure 9.6: Automatic dense labeling of ARKitScenes. We demonstrate the applicability to label RGB-D datasets that do not have dense labels available. Compared to state-of-the-art Mask3D [Schult et al., 2023], we generate dense annotations for all classes in the scene. Further, we segment on a higher level of detail (see picture and books in bookshelf, or objects on the cabinet/nightstand). Thus, our labeling pipeline can readily be used on non-label dataset to provide training data for segmentation methods.

labels knows, these are not perfect. We argue in Section 9.1.5 that this reflects the open-world approach of the dataset and annotation workflow, where – exactly as in any real application – semantics are ambiguous and not always clearly defined. We should also point out that even the detailed annotations we provide are not fully perfect. However, given the background that the ScanNet labels are also used as a benchmark to compare accuracy of semantic classifiers, our results indicate that a perfect prediction would reach accuracy values much lower than 100%. If two methods achieve higher mIoU on ScanNet than the ScanNet labels themselves, it is not possible to draw a clear conclusion about which method is better. This highlights the usefulness of improving the quality of the labels in data sets where some labels already exist.

9.2.5 Ablation Studies

Does consensus voting make the model better? Table 9.2 shows the evaluation on the standard metrics (mIoU, mAcc, tAcc) in 2D for the ScanNet and the Replica datasets. We demonstrate that aggregating individual 2D predictions with our consensus voting mechanism improves upon the individual 2D models. Further, we also show that lifting the 2D consensus into 3D using our optimization pipeline further improves the results compared to the individual 2D models.

Which model is the most important? Table 9.2 shows that the performance of models differs noticeably. Compared to the others, InternImage and Mask3D have the strongest positive impact on the segmentation quality. Additionally and unsurprisingly, Table 9.1 shows that using ScanNet [Dai et al., 2017a] labels as additional votes further improves performance.

Importance of 3D Lifting? We show in Table 9.2 the effect of 3D lifting to aggregate semantic labels and make them multi-view consistent. We compare LabelMaker with the aggregated consensus, as well as with individual models, and compute the 2D metrics on ScanNet and Replica. One can see that the 3D lifting significantly improves the performance by at least +1 mIoU.

9.2.6 Experiments on ARKitScenes

To demonstrate the applicability of our labeling pipeline to new datasets, for which no dense labels exist, we run our pipeline on a set of scenes from the ARKitScenes [Baruch et al., 2021b] dataset. To this end, we process the smartphone trajectories using the low resolution depth maps as sensor depth and the corresponding VGA-resolution images as RGB input. We established these correspondences by synchronizing the depth and RGB timestamps. In Figure 9.6, we show qualitative results for 2 scenes of the data set. One can see that the produced labels are more complete and accurate than for Mask3D, a state-of-the-art 3D instance segmentation method. Thus, we demonstrate the feasibility of automatically labeling huge datasets with zero human intervention.

9.3 Discussion

9.3.1 Limitations

One main limitation is that LabelMaker is still limited to a fixed set of classes. Thus, it does not address one of the major shortcomings of existing semantic segmentation datasets. An interesting direction to mitigate this limitation is to optimize the scene with language embeddings. Extending it to output language embeddings instead of classes would make it more flexible and potentially help to resolve ambiguities.

A second limitation are the many hyper-parameters involved in the 3D lifting. Existing work [Zhi et al., 2021b] has shown that semantic segmentation benefits from 3D lifting and this has been confirmed by our work. However, this comes at the cost of an increased number of hyper-parameters. Our 3D lifting pipeline, SDFStudio, has numerous hyper-parameters. Oftentimes the optimization is rather sensitive to the hyperparameter setting and we have not yet found the optimal setting. At the moment, the optimization has to be carefully tuned on a per scene basis in order to get the best results. Thus, the quality of the automatic labels could be a) improved and b) the scaling could be even better if we use a more robust lifting mechanism.

Finally, the pipeline can be further profit from newly developed models as research progresses, which will improve the output quality. Thus, we continuously add or replace models to our pipeline. An interesting next step would be to implement a feedback loop where LabelMaker is used to produce a vast amount of automatically labeled training data, on which an additional model can be bootstrapped in a distillation mechanism.

9.3.2 Summary

In this chapter, we presented LabelMaker, a fully automatic labeling pipeline that generates semantic annotations of similar quality to human annotations, but with zero manual human labeling effort. The method can also improve the accuracy and consistency of existing annotations. We quantitatively validate the performance of our pipeline on the ScanNet and Replica datasets. On ScanNet, it outperforms the existing human annotations, and on Replica it is better than all baseline methods. Finally, we demonstrate the applicability to large-scale 3D datasets and label images and point clouds of ARKitScenes.

Online Semantic 3D Reconstruction

As we have learned, humans not only require low-level spatial awareness of their surroundings to interact with the world, but also rely on a higher-level semantic understanding of its contents in real-time. In order to enable autonomous agents with similar capabilities, we aspire to model these processes using computational methods - this is the goal of this chapter. Building spatial awareness through 3D reconstruction has been a long-standing topic in computer vision. More recently, and given the success in spatial reconstruction, the challenge of understanding the world on a higher-level has been approached. While we have presented a method to acquire automatically annotated data in the previous chapter 9, we now present a method that enables agents to understand the world in an online scenario.

We have discussed in Chapter 8 that one can understand the world on different levels – from low-level reconstruction of 3D geometric primitives to high-level scene classification. While high-level understanding (*e.g.*, scene classification) is usually sufficient for planning and navigation, interaction also requires a fine-grained understanding with accurate semantic boundaries. As such, semantic segmentation is oftentimes at the heart of algorithms and pipelines that interact with the world, in which most approaches assign a discrete semantic class label to each reconstructed point in the scene.

In the past, there has been a large number of works in 3D scene understanding

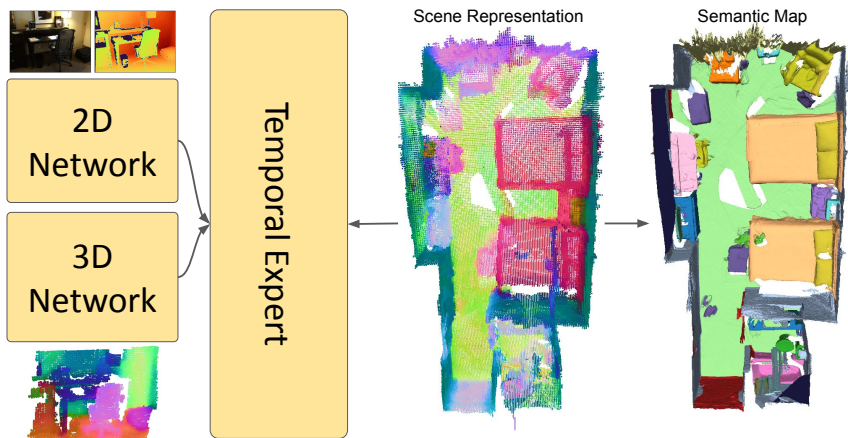


Figure 10.1: We propose an online semantic 3D reconstruction pipeline, which fuses RGB-D observations into a globally consistent semantic map. The key component is a local spatio-temporal expert network that fuses new observations into a learned scene representation. This temporal expert learns to select information from 2D, 3D, and previous steps using an attention mechanism.

which take a point-cloud, a mesh, or a voxel-grid as input, and estimate a per point semantic label [Nekrasov et al., 2021, Han et al., 2020, Kundu et al., 2020, Choy et al., 2019a]. While there has been impressive progress with these works in recent years, a large majority of these have one major shortcoming that we aim to address. All these works require an a priori reconstruction of the scene and use this global information for the understanding task. Therefore, they are considered offline methods. However, autonomous agents (as well as humans) typically build a “mental” map of the environment in an incremental manner, that is, continuously update it over time as new information is collected. Thus, scene understanding must inherently be an iterative process, as an autonomous agent cannot assume all information known a priori. In this chapter, we investigate this particular problem, where we incrementally build a semantic map given a stream of posed RGB-D data that allows for online processing of the incoming data streams and can be integrated in real-time systems. This online processing is essential for enabling real-world applications, such as robotics and mixed reality, where an updated semantic map is required to solve complex tasks in the world.

Only few approaches tackle the problem specified above. The seminal works

of Vineet *et al.* [Vineet et al., 2015] and SemanticFusion [McCormac et al., 2017] map 2D semantic predictions into 3D. One step further, PanopticFusion [Narita et al., 2019] predicts a semantic instance map in 2D that is mapped and aggregated in 3D. While these methods reason in 2D as well as 3D, the 3D reasoning is a CRF-based regularizer that is limited compared to modern neural networks. Further, the CRF requires global information of the entire scene that limits the scalability of the methods to small scenes. Similarly, INS-Conv [Liu et al., 2022b] estimates semantic instance maps using 3D processing with a large UNet that requires global processing to avoid drifting errors. In contrast, other works [Zhang et al., 2020, Huang et al., 2021b] perform 3D reasoning using point- or supervoxel convolutions in a local frame. However, they only store explicit labels that only encode per-point information, while our work uses learned features encoding low-, mid- and high-level information.

Our work is based on the observation that 2D and 3D information is complementary for the task of scene understanding. Some elements are better to be understood in 2D depending on context and geometry whereas others are easier to be segmented in 3D given their spatial structure. To this end, we present a novel attention-based aggregation mechanism that fuses 2D, 3D, as well as existing features into the scene. Our method only operates in a local region defined by the new measurement and integrates the updates into the learned global scene representation. Through this design, our method is independent of the scene size and can scale to large-scale scenes.

In an extensive experimental evaluation, we show that our method is competitive with existing approaches to online semantic 3D reconstruction while not requiring passes over the entire reconstruction as opposed to some other methods [Liu et al., 2022b]. This is particularly important for online processing on mobile devices and agents that are constrained in the amount of compute and memory available. We evaluate our method on ScanNet as well as SceneNN and present in-depth ablation studies to motivate our design choices. We will release the source code on acceptance of this paper to foster further research in this direction. In summary, our key contributions in this chapter are:

- We show that 2D and 3D information are complementary for the task of online

3D semantic reconstruction and improve the overall result.

- We propose a novel local fusion approach that leverages an attention mechanism to combine existing features with new 2D and 3D information in an online fashion. We evaluate our pipeline design on the well-known ScanNet [Dai et al., 2017a] benchmark and show competitive results compared to existing online local reconstruction methods.

10.1 Method

How do we reconstruct and understand the world around us from a stream of RGB-D data? In this chapter, we present an online reconstruction pipeline shown in Fig. 10.2 that contains a spatio-temporal expert network to enable efficient local updates of the scene representation.

10.1.1 Overview

How is our overall model designed? Our proposed model consists of three major components (Fig. 10.2) and a learned 3D scene representation. The first stage is a 2D encoder F^{2D} that extracts 2D feature maps from an incoming stream of RGB-D images. The second stage is a 3D encoder F^{3D} that incorporates 3D geometry into each feature map after lifting it to 3D using the given camera parameters and depth maps. The third stage is a new *temporal expert* network $F^{\Delta T}$ that consolidates 3D scene representations using complementary information from 2D and 3D as well as the so-far reconstructed 3D scene.

10.1.2 Scene Representation

As discussed in chapter 3, the backbone of every online reconstruction method is a suitable scene representation. Typically, the primary choice are voxels, points, meshes or implicit (neural) representation. Meshes and implicit representations are difficult to update with new observations, while points lack information about geometric connectivity. This is crucial for scene understanding where decisions about segmentation boundaries are oftentimes guided by geometric boundaries.

Therefore, we represent scenes using a hybrid representation S combining learned and explicit features that are stored in a sparse voxel grid. Sparse voxel grids allow for efficient processing using neural networks. In particular and similar to NeuralFusion presented in 5, each voxel stores a learned feature F of dimension $D_F = 40$ encoding the aggregated information about the scene content. This learned scene representation allows to store high-, mid-, and low-level information useful for the semantic segmentation task. This mitigates the need for expensive re-processing in deep neural networks at every time step to fuse existing and new information. The voxels also store the number of per-voxel observations, which is relevant for the subsequent fusion step.

10.1.3 2D Encoder

The aim of the first stage is to extract semantic features from incoming 2D RGB-D images using a 2D convolutional network F^{2D} . The 2D network F^{2D} with trainable parameters θ^{2D} takes RGB-D frames (I_t, D_t) as input and predicts semantic features \tilde{f}_t^{2D} per frame:

$$\tilde{f}_t^{2D} = F^{2D} \left([I_t, D_t, N_t]; \theta^{2D} \right) \quad (10.1)$$

The normal map N_t is estimated from the depth map D_t and serves as additional input. The network consists of DeepLabV3+ [Chen et al., 2017b] and uses, similar to [Kundu et al., 2020], an Xception65 [Chollet, 2017] encoder that is adjusted to handle RGB-D-N input data (color, depth, normals) since geometric information improves semantic segmentation [Gupta et al., 2014, Wang et al., 2016, Hazirbas et al., 2016]. The semantic features \tilde{f}_t^{2D} are directly obtained from the DeepLab decoder. However, the original dimension of the features \tilde{f}_t^{2D} is $D_{\tilde{f}_t^{2D}} = 256$ which is too memory intensive for online processing of large indoor scenes. Instead, we project the feature maps to $D^{2D} = 40$. This compression allows online processing for the remaining of the pipeline while still retaining all relevant information needed for 3D semantic understanding. The 2D network F^{2D} is pre-trained on ImageNet [Deng et al., 2009] and fine-tuned on 2D training data from ScanNet [Dai et al., 2017a]. During the training of the full pipeline, the

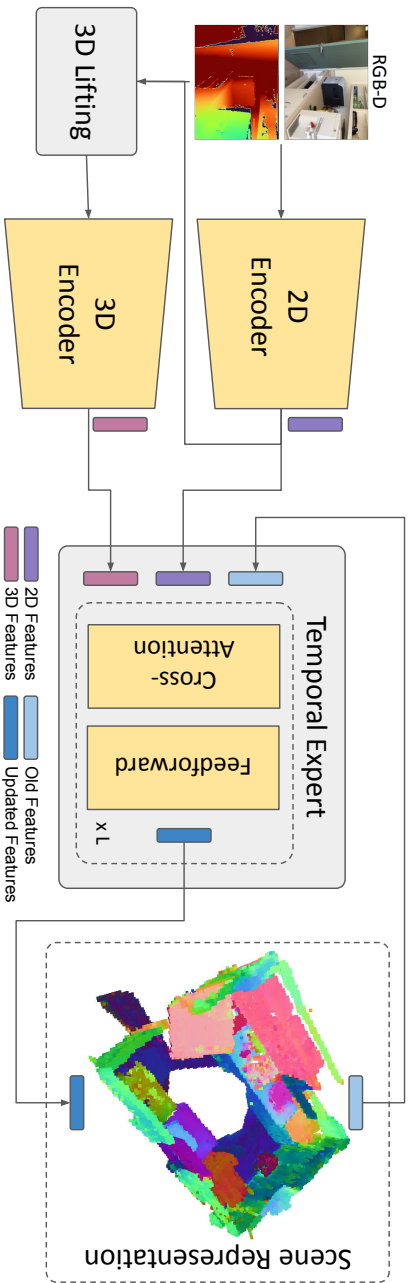


Figure 10.2: Pipeline Overview: Our pipeline consists of three main stages. The 2D encoder extracts information from incoming RGB-D imagery. This information is enhanced with 3D information using a light-weight 3D encoder. The information from these two sources is combined with existing information in the learned scene representation using the temporal expert.

2D encoder is partially fine-tuned using an auxiliary semantic segmentation head enforcing consistent performance across frames and for regularization of the joint feature space.

10.1.4 3D Encoder

We additionally process the incoming information in 3D as geometry is complementary to 2D appearance. This processing is particularly motivated by the possible reasoning about hidden geometric object boundaries occluded in the current 2D frame. To this end, we lift the obtained 2D feature map f_t^{2D} to 3D point clouds by projecting the depth map D_t using the known, gravity-aligned camera orientation $R \in SO(3)$ and intrinsics. Due to noisy depth estimates, we additionally filter out points that are more than 3 m away from the camera. The resulting *local* 3D feature volume is refined using a light-weight U-Net [Ronneberger et al., 2015] F^{3D} yielding a 3D feature map f_t^{3D} with the same feature dimension $D_F = 40$ as the 2D feature map f_t^{2D} . The UNet consists of four blocks and each block consists of two residual layers with interleaved batch norm and ReLU layers. The output of this layer is downsampled to the resolution of the next block using a convolutional layer with a kernel size and stride of two. The output of the lower block is processed by a transposed convolution and combined with the skip connection of the current level. Similar to the 2D encoder, the 3D encoder is also supervised using an auxiliary head for semantic segmentation enforcing an as good performance as possible from a single 3D frame.

10.1.5 Spatio-Temporal Expert

In the previous stages, 2D features are extracted and enhanced with 3D information. In the next step, this information is integrated into the existing global scene representation S_{t-1} . To this end, we propose the spatio-temporal expert network $F^{\Delta T}$ with weights $\theta^{\Delta T}$. The task of the spatio-temporal expert network $F^{\Delta T}$ is to update the features stored in the learned scene representation given the new information from f_t^{2D} , f_t^{3D} , and the existing information f_{t-1}^{global} . The feature volume f_{t-1}^{global} is a crop of the relevant local sub-volume from the global scene

representation S_{t-1} using the known camera pose $[R|t] \in SE(3)$. The overall mapping is computed as:

$$f_t^{\text{global}} = F^{\Delta T} \left(\left[f_{t-1}^{\text{global}}, f_t^{3D}, f_t^{2D} \right]; \theta^{\Delta T} \right) \quad (10.2)$$

The resulting local volume f_t^{global} is then written back, using the inverse camera pose, to obtain the new global scene representation S_t . By providing access to both the 2D and 3D features in parallel, the expert network can learn where it is beneficial to rely more on 2D appearance features or where it is advantageous to trust the 3D geometry-based features, see Fig. 10.4 for an illustration. The 3D features reveal geometrical details while the 2D features provide textural information in flat areas with little geometric information.

$F^{\Delta T}$ is implemented as a transformer consisting of cross-attention and feed-forward layers (see Fig. 10.3). The task of the cross-attention layer is to extract relevant information from the three sources of information (f_t^{2D} , f_t^{3D} , f_{t-1}^{global}) using f_{t-1}^{global} as query features. The attention is defined as $f = w_{f_{t-1}^{\text{global}}} v_{f_{t-1}^{\text{global}}} + w_{f_t^{2D}} v_{f_t^{2D}} + w_{f_t^{3D}} v_{f_t^{3D}}$ and the weights ($w_{f_{t-1}^{\text{global}}}$, $w_{f_t^{2D}}$, $w_{f_t^{3D}}$) are defined as:

$$w_{f^k} = Q \left(f_{t-1}^{\text{global}} \right)^T K \left(f^k \right), \quad (10.3)$$

where $f^k \in \{f_{t-1}^{\text{global}}, f_t^{3D}, f_t^{2D}\}$. The values v_{f^k} are obtained using a linear projection layer $v_{f^k} = V(f^k)$.

The features obtained from the cross-attention layer are first normalized using layer norm and then refined using a feed-forward layer as it is standard in transformer architecture. Furthermore, both layers are augmented with a skip connection guaranteeing healthy gradients during training. The refined features that consist of information extracted from the three sources f_t^{2D} , f_t^{3D} , and f_{t-1}^{global} are written back into the global scene representation. The temporal expert is also supervised by a point-wise segmentation loss that enforces optimal segmentation given the currently available 3D scene information.

Positional encoding of encoder features. To enable the spatio-temporal expert network to make optimal fusion decisions, it has to learn from which encoder

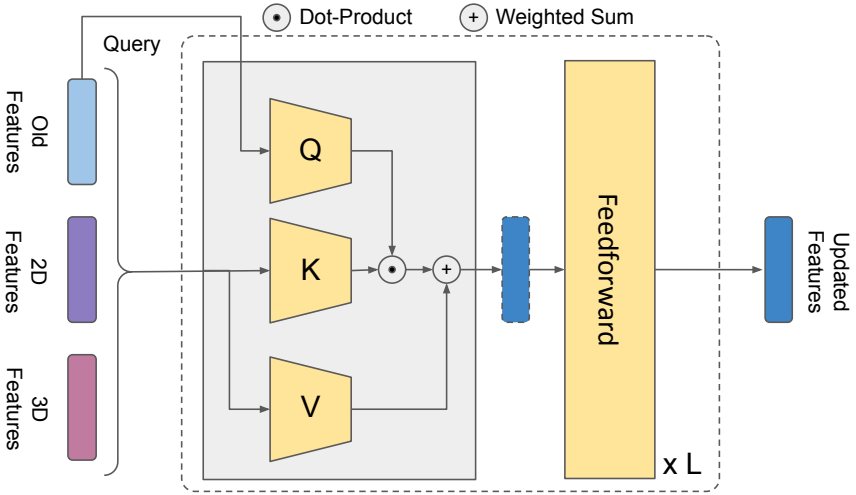


Figure 10.3: Temporal expert network. The temporal expert network takes the three features (f_{i-1}^{global} , 2D, and 3D) as input, and iteratively refines the old feature vector to obtain the update feature that can be stored in the scene representation. The old feature is used as the query in the attention mechanism.

to select information from. However, transformer architectures are invariant to permutations in their input sequence. This means that the network cannot learn from which source a specific feature in the input is obtained. Thus, we have to encode the source information into the feature that is passed to the network. As we keep the ordering of the different sources fixed in the input sequence, we can treat the source information like positional information. Therefore, we encode the source information using positional encoding as it is standardized in transformer networks. This positional encoding allows to inform the temporal expert network from which source a specific feature is coming.

10.1.6 Loss Function and Training Details

The pipeline is trained using focal loss [Lin et al., 2017] at several stages in the pipeline. These losses are applied after the 2D encoder F^{2D} , the 3D encoder F^{3D} , and the temporal expert network $F^{\Delta T}$. These auxiliary supervision signals are required to constrain the feature space that encodes the information throughout

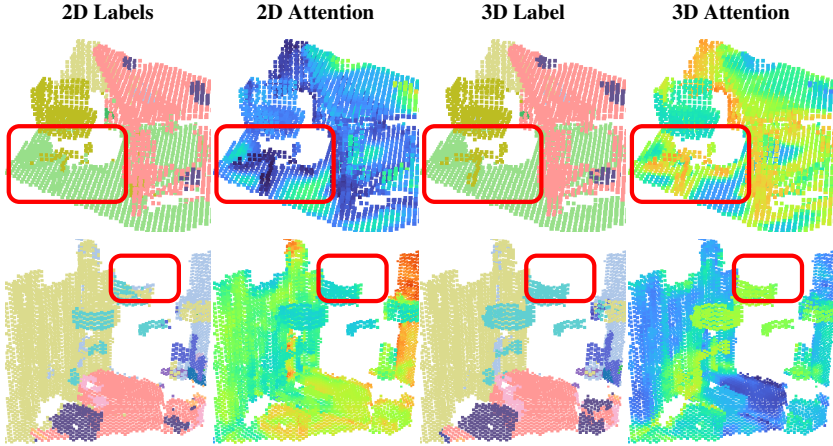


Figure 10.4: Temporal expert attention maps. We visualize the attention maps for the 2D and 3D input features in the expert network. The expert attends to the 3D features to refine the segmentation of fine details (legs of the chair, lamps), while it attends to the 2D features to predict the label for large areas (table, walls, *etc.*).

the entire pipeline. Further, these auxiliary losses ensure that each stage solves the task of semantic segmentation as good as possible for themselves providing the temporal expert network with valuable information. The overall loss is the sum of these losses:

$$\mathcal{L} = \lambda_{2D}\mathcal{L}_{2D} + \lambda_{3D}\mathcal{L}_{3D} + \lambda_{\text{Expert}}\mathcal{L}_{\text{Expert}} \quad (10.4)$$

where each term \mathcal{L}_{2D} , \mathcal{L}_{3D} , $\mathcal{L}_{\text{Expert}}$ is a focal loss, defined as:

$$\mathcal{L} = (1 - \hat{y})^\gamma \text{CE}(\hat{y}, y) \quad (10.5)$$

As the loss is applied on a per-voxel level, the corresponding groundtruth labels first need to be mapped from the ground truth polygon mesh to a voxelized representation. To this end, we first voxelize all scenes using the target resolution and assign the label of the closest vertex of the mesh. Closest points are efficiently found using KD-tree-based nearest neighbor search.

Auxiliary heads. After each sub-network, we leverage a additional classification head to supervise the pipeline using an auxiliary semantic loss. The additional head consumes the features predicted by the network and passes them through a

two-layer MLP that is interleaved with batch normalization and ReLU activation. For all sub-networks, we use the exact same architecture in order to uniformly constrain the feature space. The goal of these additional heads is to enable an auxiliary loss that enforces the sub-networks to encode as much semantic information as they possibly can extract from the given input.

Avoiding feature drift. When using learned features during iterative fusion of sensor measurements, the problem of feature drift can arise. Feature drift is a problem during inference. During inference, features might drift outside the distribution they have been trained on. Thus, the trained neural networks fail to handle them properly as they represent unknown information. Oftentimes, this drift manifests itself by the growing scale of the norm of the individual features. In our pipeline, this is primarily caused by the bias of the different layers in the 3D encoder and the spatio-temporal expert network that are iteratively added on top of the features. Thus, we have to prevent that during training and testing.

While other works, *e.g.*, INS-Conv [Liu et al., 2022b], address this issue by performing network passes over the entire scene (which can limit scalability to large-scale scenes), we approach this problem such that it is more suitable for online fusion: Given the observation that the growing norm is the main driver of feature drift, we utilize a simple normalization strategy. To this end, after all sub-networks – 2D encoder, 3D encoder, and spatio-temporal expert network – we pass the final features through a one-dimensional layer norm. This yields normalized features and prevents their norm from growing outside the training distribution.

Sequential Training. The temporal expert network needs to learn how to fuse new information into the existing scene representation based on sequential data. A key challenge is catastrophic forgetting, where the network forgets what it has learned during the beginning of a sequence and only focuses on the last few frames along a camera trajectory. To overcome this challenge, it is critically important to randomly select camera views along each video trajectory, *i.e.*, a permutation of the original frame order. Similarly, to avoid that the model only sees fully reconstructed scenes after some initial training time, we randomly reset the reconstructed scenes so that the model always sees scenes at varying levels of

reconstruction.

10.2 Experiments

10.2.1 Implementation Details

We implement the proposed pipeline in PyTorch. We use the MinkowskiEngine [Choy et al., 2019a] for the sparse 3D convolutions in the 3D encoder, and Pytorch3D [Ravi et al., 2020] for the geometric projections. The entire pipeline is trained with the Adam optimizer and a OneCycle [Smith and Topin, 2017] learning rate scheduler. Due to memory constraints, the batch size is 4 but we obtain an effective batch size of 8 by aggregating the gradients across two batches. We set the maximum learning rate to 0.001 for the 3D and temporal expert networks, while the maximum learning rate for the pre-trained 2D encoder is set to $1e - 05$. We equally weight the different terms in the loss function setting $\lambda_{3D} = \lambda_{2D} = \lambda_{\text{Expert.}} = 1$. Further, we set the parameter of the focal loss $\gamma = 1$. We use five layers in the temporal expert transformer with a hidden dimension of $D_{\text{hidden}} = 128$ in the feed-forward layers. The voxel grid resolution for the entire pipeline is set to 4 cm.

10.2.2 Online Methods in Comparison

FusionAware [Zhang et al., 2020]. Unlike our voxel-based representation, FusionAware is a *point*-based online 3D semantic segmentation method. The method aggregates measurements in 3D space using point convolutions and computes intra- and inter-frame features.

SVCNN [Huang et al., 2021b]. Similar to ours, Supervoxel Convolution (SVCNN) is another candidate from the space of voxel-based approaches. SVCNN uses dedicated convolutional operators that operate directly on supervoxels and aggregate multi-view features during online reconstruction.

Method	Processing mIoU \uparrow	Bed	Bathroom	Cabinet	Chair	Counter	Curtain	Desk	Door	Floor	Other	Picture	Fridge	Shower	Sink	Sofa	Table	Toilet	Wall	Window	
Mix3D [Nekrasov et al., 2021]	78.1	85.5	84.3	78.1	85.8	57.5	83.1	68.5	71.4	97.9	59.4	31.0	80.1	89.2	84.1	81.9	72.3	94.0	88.7	72.5	
VirtualMVFusion [Kimdu et al., 2020]	74.6	77.1	81.0	84.8	70.2	86.5	39.7	89.9	69.9	66.4	94.8	58.8	33.0	74.6	85.1	76.4	79.6	70.4	93.5	86.6	72.8
Minkowski [Choy et al., 2019a]	73.6	85.9	81.8	83.2	70.9	84.0	52.1	85.3	66.0	64.3	95.1	54.4	28.6	73.1	89.3	67.5	77.2	68.3	87.4	85.2	72.7
PanopticFusion [Narita et al., 2019]	52.9	49.1	68.8	60.4	38.6	63.2	22.5	70.5	43.4	29.3	81.5	34.8	24.1	49.9	66.9	50.7	64.9	44.2	79.6	60.2	56.1
INS-Conv [Liu et al., 2022b]	71.7	75.1	75.9	81.2	70.4	86.8	53.7	84.2	60.9	60.8	95.3	53.4	29.3	61.6	86.4	71.9	79.3	64.0	93.3	84.5	66.3
FusionAware [Zhang et al., 2020]	63.0	60.4	74.1	76.6	59.0	74.7	50.1	73.4	50.3	52.7	91.9	45.4	32.3	55.0	42.0	67.8	68.8	54.4	89.6	79.5	62.7
SVCNN [Huang et al., 2021b]	63.5	65.6	71.1	71.9	61.3	75.7	44.4	79.5	53.4	56.6	92.8	47.8	27.2	63.6	53.1	66.4	64.5	50.8	86.4	79.2	61.1
ALSTER (Ours)	66.8	82.2	77.1	49.6	65.1	83.3	54.1	76.1	55.5	61.1	96.6	48.9	37.0	38.8	58.0	77.6	75.1	57.0	95.6	81.7	64.6

Table 10.1: 3D Semantic Segmentation on ScanNet [Dai et al., 2017a] test set. Offline baselines predict semantic labels using a-priori 3D scene reconstructions and **global** passes over the entire scene. Online but **global** baselines do online reasoning but require global passes over the full scene. **Local** methods are online and reason only on local information within the viewing frustum and on currently updated points. Among local methods, our proposed approach improves over existing baselines by at least +3.3 mIoU.

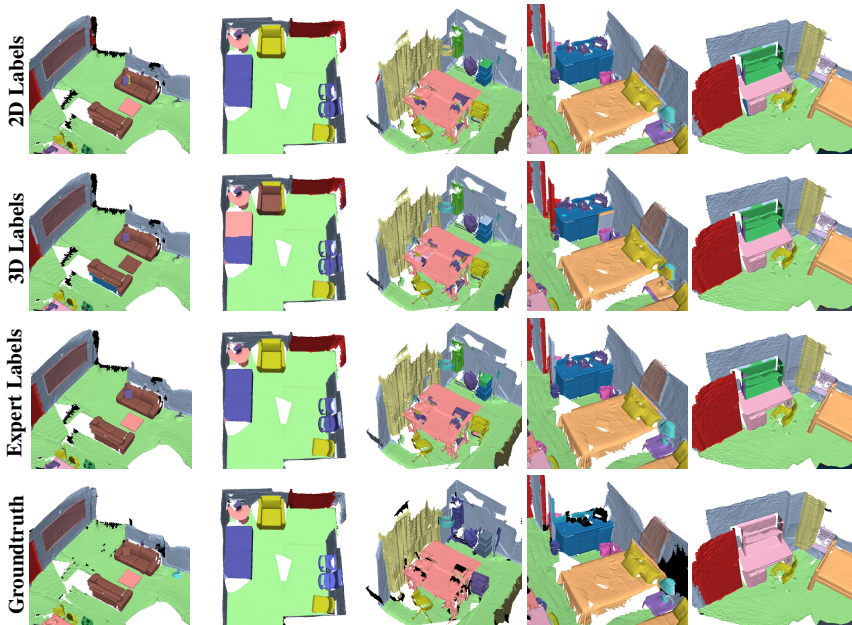


Figure 10.5: Qualitative results of our proposed method and comparison between the different stages. The expert successfully selects the correct information from the two encoders. The 2D encoder is focused on object-level decisions (*e.g.*, table in column 1, bench in column 2) while the 3D information is used for fine details (*e.g.* lamp in column 3, desk in column 5).

10.2.3 Datasets and Metrics

ScanNet [Dai et al., 2017a]. consists of 1513 scans from 707 unique indoor scenes containing 2.5M RGB-D frames. All scenes provide dense 3D semantic annotations mapped to the NYU40 class labels. Each scene is recorded up to three times using an iPad equipped with an Occipital depth sensor. The camera poses and dense reconstruction of the scenes are obtained using BundleFusion [Dai et al., 2017b]. The 3D labels are projected into all 2D frames to provide the 2D labels.

SceneNN [Hua et al., 2016]. SceneNN consists of 76 scenes with semantic annotations and corresponding posed RGB-D data. We follow [Huang et al., 2021b, Liu et al., 2022b] and demonstrate the generalization capabilities of our

method by training on ScanNet and evaluating on the 76 SceneNN scenes.

Metrics. We follow the standard metrics of the ScanNet and SceneNN datasets. In particular, we compute the mean and per-class intersection over union (IoU) on ScanNet, as well as the mean accuracy (mAcc) and weighted intersection over union (wIoU) on SceneNN.

10.2.4 3D Semantic Segmentation

Table 10.1 reports 3D semantic segmentation scores of our and recent methods on the well-established ScanNet [Dai et al., 2017a] test benchmark. We compare online and offline methods, as well as local and global methods. While offline methods rely on a pre-computed 3D scene reconstruction in the form of a point cloud or polygon mesh, online methods are able to reconstruct the 3D scene on the fly as new frames become available. This functionality is attractive for online applications in robotics or AR/VR devices, however they cannot rely on the full scene context which makes semantic reasoning harder and semantic scores are generally higher for offline methods [Nekrasov et al., 2021, Kundu et al., 2020, Choy et al., 2019a]. In the group of online methods, our approach improves over the existing local methods like SVCNN [Huang et al., 2021b] and FusionAware [Zhang et al., 2020] by at least +3.3 mIoU. Local methods operate on a local window defined by the newly incoming frames and are therefore memory and computationally efficient, both attractive qualities for real-time processing. Global methods require global passes over the reconstructed scenes either by CRF regularization or neural network processing. This step takes increasingly more time as the reconstructed scene becomes larger in size. These methods are therefore less applicable for real-time applications. Specifically no upper bound on the processing time can be guaranteed.

In Table 10.3, we compare our method to existing baselines on the ScanNet validation set as well as SceneNN. For ScanNet, we report the mIoU over all benchmark classes. For SceneNN, we compute the weighted intersection over union (wIoU) and the mean accuracy (mAcc) for all annotated NYU40 classes. In addition to the quantitative metrics, we also report the voxel resolution for all methods where it is available. A smaller voxel size generally results in better

Method	mIoU \uparrow	Wall	Floor	Cabinet	Bed	Chair	Sofa	Table	Door	Window	Bookshelf	Picture	Counter	Desk	Curtain	Fridge	Shower Curt.	Toilet	Sink	Bathtub	Other Furn.
Ours - 2D	68.2	83.4	96.3	60.1	75.0	85.8	77.0	68.7	65.7	60.3	66.7	28.2	64.8	56.1	68.3	61.8	58.8	86.6	62.7	86.0	52.1
Ours - 3D	<u>69.0</u>	<u>85.0</u>	96.8	59.2	73.9	87.3	<u>75.8</u>	<u>69.7</u>	69.5	<u>61.3</u>	62.9	33.1	65.7	<u>56.1</u>	68.4	61.2	<u>60.2</u>	89.0	64.1	87.0	52.8
Ours - Temp	70.6	85.6	96.9	<u>59.6</u>	77.1	87.3	75.0	71.9	<u>68.4</u>	65.8	<u>65.9</u>	36.6	67.2	61.1	69.4	62.6	65.0	89.8	65.2	87.1	53.8
$D_F = 64$	67.9	<u>84.4</u>	<u>96.2</u>	59.4	76.9	85.6	73.0	69.5	<u>66.3</u>	63.1	36.4	24.8	65.6	61.6	71.3	<u>61.5</u>	67.1	89.1	64.7	89.4	52.9
$D_F = 40$	70.6	85.6	96.9	<u>59.6</u>	77.1	87.3	75.0	71.9	68.4	65.8	65.9	36.6	67.2	<u>61.1</u>	69.4	62.6	<u>65.0</u>	89.8	<u>65.2</u>	87.1	<u>53.8</u>
$D_F = 32$	68.2	83.0	94.9	60.8	78.5	86.3	76.0	69.8	61.6	58.7	63.6	24.6	64.5	61.1	70.0	60.9	50.4	90.6	67.5	87.5	53.0
$D_F = 16$	68.1	83.2	95.2	58.5	<u>77.9</u>	84.9	75.8	69.4	61.5	<u>63.8</u>	52.9	<u>33.0</u>	60.1	58.2	64.9	<u>61.5</u>	64.7	90.8	63.9	85.9	55.0
Xception	70.6	85.6	96.9	59.6	77.1	87.3	75.0	71.9	68.4	65.8	65.9	36.6	67.2	61.1	69.4	62.6	65.0	89.8	65.2	87.1	53.8
MoblieNet	66.0	81.2	95.9	57.8	73.9	83.1	70.8	68.0	60.3	50.2	63.9	37.3	63.1	61.9	63.4	56.3	50.1	88.6	58.5	86.2	50.0

Table 10.2: Ablating different aspects of our pipeline on ScanNet [Dai et al., 2017a] validation set. We show that the expert selects valuable information from the two encoders and the existing scene representation by evaluating the individual pipeline outputs. The expert network consistently improves upon the two other stages in terms of IoU. We also evaluate the impact of the stored feature dimension D_F on the overall result. While the smaller feature sizes suffer from compression due to limited capacity the larger features ($D_F = 64$) suffer from slight overfitting. Finally, we compare the Xception 2D encoder to the lighter MoblieNet ($\times 20$ fewer parameters). Unsurprisingly, the smaller encoder leads to a slight deterioration of performance, but the results indicate potential for runtime-accuracy tradeoffs in time-critical applications.

3D Semantic Segmentation					
	Processing	Res. [cm]	ScanNet	SceneNN	
			val. mIoU	wIoU	mAcc
SemanticFusion [McCormac et al., 2017]	Global	N/A	42.3	47.1	<u>58.5</u>
PanopticFusion [Narita et al., 2019]	Global	2.4	<u>53.1</u>	–	–
InsConv [Liu et al., 2022b]	Global	2	72.4	–	79.5
SemanticReconstruction [Jeon et al., 2018]	Local	N/A	44.0	–	–
ProgressiveFusion [Pham et al., 2019]	Local	0.8	55.0	52.2	61.6
FusionAware [Zhang et al., 2020]	Local	N/A	67.2	63.9	71.7
SVCNN [Huang et al., 2021b]	Local	N/A	<u>68.3</u>	69.0	76.9
ALSTER (Ours)	Local	4	70.6	<u>67.8</u>	76.9

Table 10.3: 3D Semantic Segmentation on ScanNet and SceneNN. Scores are mean intersection over union (mIoU) on ScanNet [Dai et al., 2017a] validation, the mean accuracy (mAcc) and weighted IoU (wIoU) on SceneNN [Pham et al., 2019]. All other scores are as reported in [Huang et al., 2021b] and [Liu et al., 2022b].

scores since finer details can be represented, however this comes at increased memory costs. Our proposed method performs best among all local methods on both ScanNet and SceneNN (with a close second on the wIoU metric). When also compared to global methods, INS-Conv [Liu et al., 2022b] performs only marginally better on ScanNet, even when using a voxel resolution that is twice as high, highlighting the memory efficiency of our method.

Qualitative Results.

In Figure 10.6 and 10.7, we show additional qualitative results emphasizing the benefits of leveraging the complementary information obtained from 2D and 3D. As it can be seen from comparing the outputs of the 2D and 3D networks, the 2D network is mainly responsible for avoiding object-scale confusion of the semantic class. *E.g.*, a desk that this erroneously confused as a table by the 3D network is correctly classified by 2D network. The spatio-temporal expert network learns to select the correct information and corrects the errors. On the other hand, the 3D network mainly refines low-level misclassifications of points in the scene and acts as a neural regularizer making sure that all points belonging to the same object are consistently labelled. *E.g.*, this can be seen in the case of the chair or desks, where only partially wrong predictions made by the 2D network, which are subsequently refined by the 3D network. The spatio-temporal expert learns to leverage this information from the 3D network. In summary, the output of the

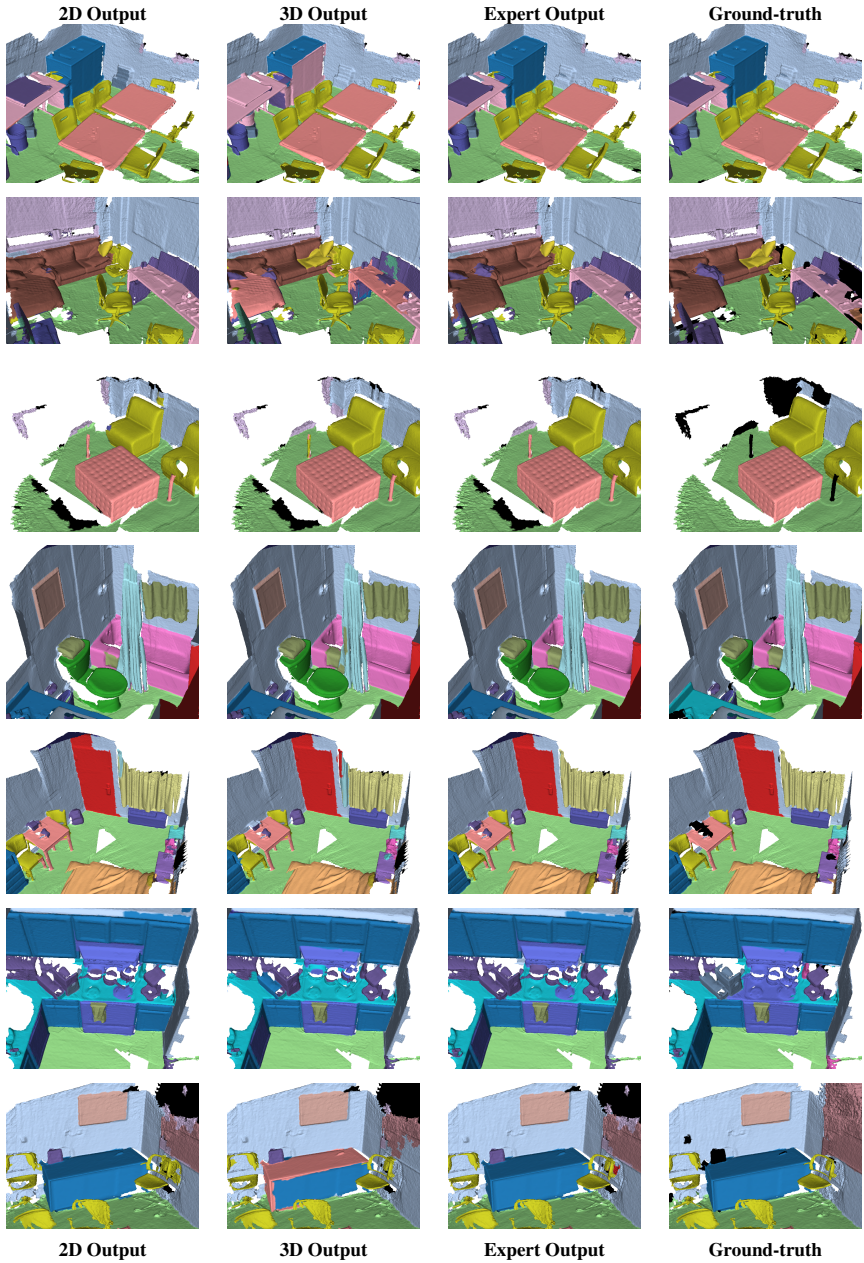


Figure 10.6: Additional qualitative results on the ScanNetv2 validation set.

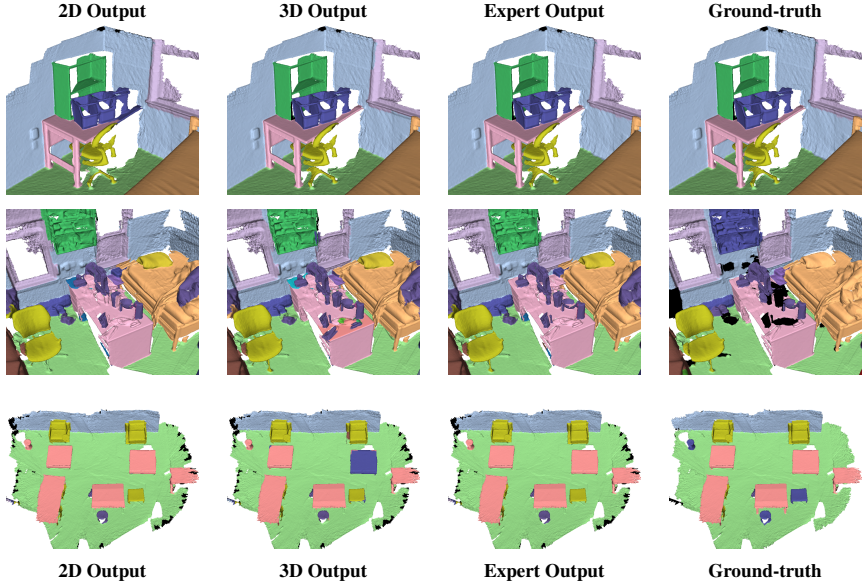


Figure 10.7: Additional qualitative results on the ScanNetv2 validation set.

spatio-temporal expert network is consistently better than the output of the two separate branches by fusing their complementary information with the already integrated information.

10.2.5 Ablation Studies

Does the temporal expert network improve upon the individual 2D and 3D networks? In Table 10.2, we report the numbers for the different stages in our pipeline. For each stage, we compute the per-class intersection over union on the ScanNet validation set. The per-stage labels are obtained from the auxiliary heads used during training to constrain the joint feature space and aggregated using a simple voting mechanism. These labels are then mapped to the ScanNet groundtruth and evaluated using their evaluation pipeline. The numbers show that the expert is consistently better than the individual branches (Ours - 2D and Ours - 3D). Further, the fact that sometimes the 2D labeling is better than the 3D labels and the significant margin between 3D and expert indicate that 1) bypassing the 2D information around the 3D encoder is useful and 2) our attention-based fusion

mechanism allows better reasoning over time than simple voting. We also show the differences between the different stages in Figure 10.5, where one can see the benefits of selecting information from the two different encoders.

How does our method handle limited data? In order to evaluate the performance of our method on limited data, we randomly selected 10 scenes from the ScanNet validation set. Then, we subsampled the trajectories with different set sizes in order to simulate limited data. In Figure 10.8, we show the performance for these different step sizes. For small step sizes, the performance only slightly changes. For larger step sizes (step size > 10), the performance starts to drop to a subpar performance. However, even for a very large step size (step size = 50), the performance does not completely deteriorate.

What does the temporal expert network attend to? In Figure 10.4, we visualize the attention maps for different frames during the fusion process together with the corresponding predicted labels. We qualitatively show that they learn to leverage the two different encoder according to their individual strengths. The temporal expert network attends to the 3D network for fine-details usually refining edges and geometric details (*e.g.*, legs of a chair, edge of a table) while it attends to the 2D feature for information about large regions. Further, we observe that the fusion with the old scene representation happens in later layers while earlier layers combine the 2D and 3D information.

What is the impact of the feature dimension D_F ? The dimension D_F of the features stored in the learned scene representation is a key hyperparameter of the pipeline. Thus, we evaluate its impact on the overall performance in Table 10.2. One can see that for D_F below 40 (the default value), the performance is slightly deteriorated due to the required compression of the semantic information. For $D_F = 64$ the main reason for the slight performance drop is the increased overfitting to the training data due to the increased capacity of the features.

How fast is our method? An average step through our entire (non-optimized) pipeline takes 116.1 ms (8.6 FPS) on an NVIDIA RTX 2080 and 3.6 GHz Intel CPU i9-9900K. To identify the main bottleneck, we analyse the runtime of the individual components in Fig. 10.9. One pass through the 2D network DeepLabV3 [Chen et al., 2017b] plus lifting takes on average 83.5 ms. One

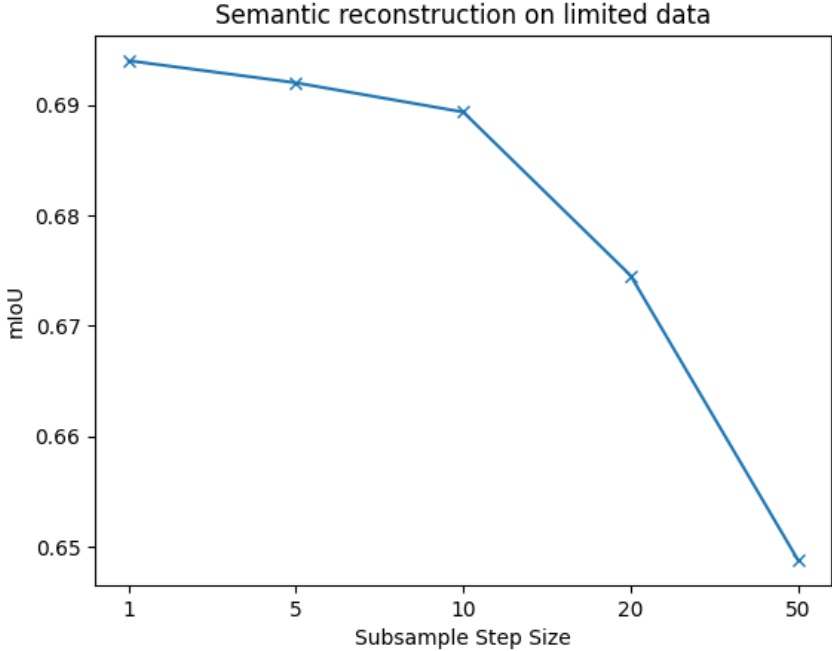


Figure 10.8: Our model on limited data. We evaluate the performance of the proposed pipeline on limited data by subsampling the trajectories with different step sizes. We can see that up to a subsampling step size of 10, the performance drop is insignificant while above that it starts to drop. However, also under these conditions the performance does not completely deteriorate.

pass through our light-weight 3D UNet takes 28.6 ms on average, and the temporal expert network operates at 4 ms per frame. These numbers reveal the 2D DeepLabV3 as the main bottleneck. Thus, we also report the performance of MobileNet [Howard et al., 2017] in Tab. 10.2 instead of the standard Xception encoder. This reduces the 2D processing time to 45.2 ms and boosts the overall runtime to 12.9 FPS.

How many parameters does our pipeline have? Our pipeline consists in $51 \cdot 10^6$ parameters in total. The largest share is due to the the DeepLabV3 (Xception) model, which consists of $41 \cdot 10^6$ parameters. The 3D U-Net consists of $10 \cdot 10^6$ parameters. Compared to both encoders, the expert model takes a relatively small share of $92 \cdot 10^3$ parameters. This further justifies our architecture

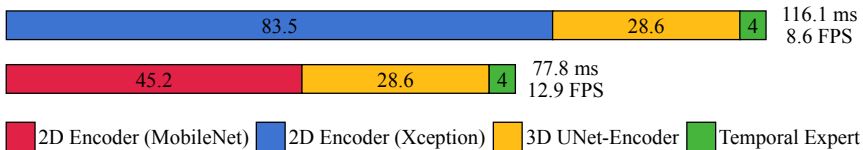


Figure 10.9: Runtime analysis for different 2D backbones. A smaller 2D encoder trades runtime for accuracy (*c.f.* Tab. 10.2).

design, with a marginal increase in model size and runtime, we obtain a notable boost in performance (+1.6 mIoU, see Tab. 10.2)

10.3 Discussion

10.3.1 Discussion of Baseline Comparison

Firstly, we would like to further detail the differences to the main baselines. While our pipeline purely operates in a local frame, some other baselines aggregate global information at some stage in the pipeline. *E.g.*, INS-Conv [Liu et al., 2022b] runs a network pass over the entire reconstruction of the scene in order to avoid feature drift. This is expensive when the pipeline is applied in the reconstruction of large-scale scenes. There it can become a severe bottleneck to run global processing to avoid feature drift. We improve this issue by carefully designing the pipeline with normalization layers in order to avoid this feature drift without requiring global processing. Secondly, due to the difficulty of unavailable code for the global methods, we cannot get more qualitative and quantitative results than the information in their papers.

10.3.2 Why is there no qualitative comparison to baselines?

As the reader might have noted, there is no qualitative comparison to one of the baselines. This is due to the fact that it is not standard practice to publish the full pipeline source code for the proposed methods¹. Thus, it is not possible to

¹INS-Conv [Liu et al., 2022b] has released a code snippet on <https://github.com/THU-luvision/INS-Conv> for their proposed model architecture, but not the full training and evaluation pipeline. SVCNN [Huang et al., 2021b] has released the training code for their neural network, but not the full model on https://github.com/shishenghuang/SVNet_jitter

show qualitative results on reconstructed data in a baseline comparison. With this paper, we aim to change this practice by releasing the source code on acceptance of this manuscript in order to foster more research in this field.

10.3.3 Limitations

One main limitation of our method is the sensitivity to labeling errors in the groundtruth annotations. In Figure 10.10, we visualize the confusion matrix for our pipeline on the ScanNetv2 dataset. To better visualize the differences, the values are normalized in each column, which is the relevant axis for evaluation. A difficult class on ScanNetv2 is the ‘picture’ category. One can see from the confusion matrix that this class is oftentimes confused for the wall class. However, this is also a class where the complementary benefit of 2D and 3D information is striking. We can show a significant improvement over the two encoders as well as to all baselines. We also address the problems with the two classes where a significant drop in performance is visible on the test set. We first focus on the bookshelf class. There are two issues with the bookshelf class. The first issue is caused by the hierarchical nature of the labels. *I.e.*, books inside a bookshelf are oftentimes labelled as bookshelf. However, our method predominantly “correctly” classifies these books as books. This causes a significant drop in performance. The second issue is caused by the confusion between bookshelf and shelf. These are not consistently labelled in the dataset. Thus, our pipeline does not robustly learn this distinction and miss-classifies this category. The second class are refrigerators. For this class, the confusion is fuzzier. *I.e.*, refrigerators are confused with different classes (wall, other furniture, window, and wall).

Similar to the previously presented annotation pipeline, a second limitation of the presented method is its fixed set of labels. As discussed, the world can be hardly grouped in a fixed number of categories as these categories not only change from situation to situation but are also strongly dependent on the context. At the moment, if the proposed pipeline is deployed into a different, potentially more specialized or more general setting than indoor scenes, it has to be retrained on a different dataset. An alternative avenue would be to explore the recent advances in coupling language with scene understanding for online mapping. This would

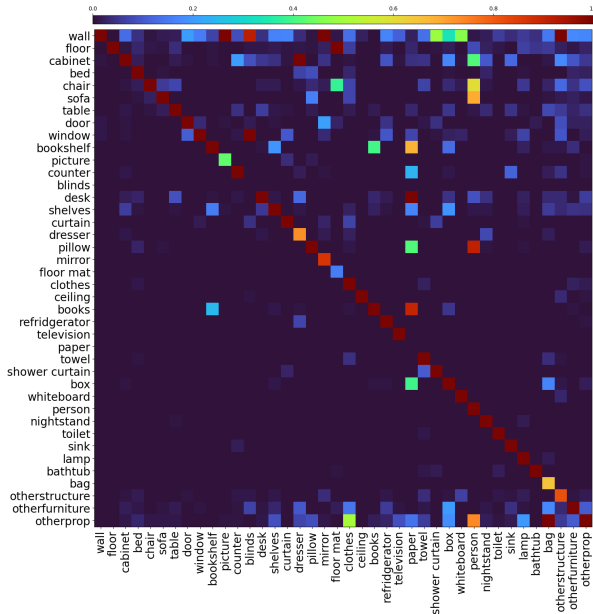


Figure 10.10: Confusion matrix of ScanNet validation set results.

potentially allow to train the pipeline only once and deploy it to any environment afterwards without retraining.

10.3.4 Summary

We presented a novel pipeline for online joint geometric and semantic 3D reconstruction. The pipeline consists of three components and a learned scene representation that represents the scene as sparse voxel grid. In order to leverage the complementary nature of 2D and 3D information, the first two stages encode 2D RGB-D data and enhance the encoded features with 3D spatial information. At the heart of your pipeline sits a spatio-temporal expert network that fuses the different sources of information and sequentially updates the learned scene representation. This network uses an attention mechanism to the 2D, 3D, and existing features to extract relevant information for the updates. We experimentally show that this design improves the performance on semantic segmentation upon the two individual branches.

Part IV

3D Scene Editing

Introduction

Throughout history, humans have sought the means to reshape and redefine reality, a desire that spans from ancient mythologies to cutting-edge technological innovations. This quest for altering reality extends seamlessly into the realm of augmented reality, where through technology like computer vision and computer graphics the possibilities are boundless. The ability to manipulate the fabric of reality holds the potential to revolutionize various domains, ranging from entertainment and education to enhancing privacy. Consider the scenario where you are in search of inspiration for new furnishings to adorn your home. Traditionally, this endeavor entails a visit to a furniture store, where you look at beautiful pieces displayed in the showroom. However, envisioning how these pieces might fit harmoniously within the confines of your own living room can be an elusive skill truly mastered by only a few. For most of us, this imaginative leap is challenging, if not impossible. This is where the power of altering reality through 3D scene editing comes into play. 3D scene editing allows to visualize how a particular piece of furniture, or any object for that matter, would look and feel within your own home. Achieving this visualization necessitates manipulating the real world through digital means.

These manipulations are the focus of 3D scene editing. In essence, 3D scene editing involves a set of fundamental operations that make it all possible. These operations encompass removing existing objects from a scene, adding new elements, resizing objects, and transforming them into different poses and positions. In theory, one can reduce all these operations to a combination of removal and

addition of objects to the scene. In doing so, we unlock the potential to reshape our surroundings and explore new possibilities with technology like augmented reality.

To convincingly edit reality, the changes must seamlessly integrate with the three-dimensional world in a manner that does not disrupt the viewer's immersion. This implies the need for edits that allow individuals to move within the real world and explore the augmented reality. Augmented reality (AR) holds the key to making this theoretical concept a practical reality. However, achieving this vision presents a multitude of challenges that must be overcome. These hurdles span various domains of computer vision, including reconstruction, understanding, and editing. While we have discussed reconstruction and understanding in parts [II](#) and [III](#), we address the problem of editing in this section. Solving this problem is essential to realize the full potential of augmented reality as a means to truly alter and enhance our perception of the world around us.

To delve into the intricacies of this challenge, it is crucial to understand the inherent complexities in convincingly editing scene. Let us take the example of removing an object from a scene, simplifying the discussion for clarity. Imagine the task of removing a sofa from your living room. In reality, this operation triggers a cascade of transformations: 1) By removing the sofa, previously concealed aspects of the room's geometry come into view. For instance, the walls and the floor hidden behind and beneath the sofa are suddenly exposed. 2) The absence of the sofa leads to a change in the lighting conditions within the room. Light rays that were once obstructed by the sofa can now illuminate areas that were previously in shadow. This, in turn, reshapes the patterns of shadows and the overall lighting ambiance across the scene. In the physical world, these changes occur organically when an object is removed. However, when undertaking this task in the digital space, such as removing the sofa from the living room, these transformations must be computationally modeled. In essence, we must recreate the real-world effects of object removal digitally, requiring a sophisticated blend of techniques from computer vision and computer graphics. In the scope of this part, we focus on the first problem, the problem of digitally removing an object from a scene.

The endeavor to digitally manipulate reality involves the manipulation of measurements that define our perception of the world using the operations described above. This necessitates a thoughtful choice of the modality in which the operations are executed. As digital image processing gained prominence, the ability to modify photographs became increasingly accessible to the masses. Consumer software like Photoshop [Adobe Systems, Inc., 1990] empowered users to wield creative control over static images, granting them the power to manipulate photographs to suit their preferences. However, these manipulations were confined to the 2D space, offering a mere snapshot of reality.

While the manipulation of 2D images suffices for certain applications, more complex scenarios demand a higher degree of sophistication. The next challenge emerges when striving to consistently edit every frame within a video sequence. In a static 2D image, the primary focus revolves around solving the aforementioned problem — removing the object and filling the created hole. However, when it comes to video, the complexity increases. Achieving a convincing removal and inpainting necessitates not only solving these challenges for an individual frame but also maintaining consistency across all frames. This consistency is paramount for convincing the viewer of the seamless edit.

Yet, an inherent limitation of video arises — it is constrained to a predetermined trajectory through the scene. It confines our ability to freely navigate and explore the edited space along various paths. To overcome this limitation and enable unrestricted exploration, we must adopt a scene representation that permits rendering from any viewpoint within the scene, while still facilitating the desired edits, such as the removal of an object. Neural radiance fields [Mildenhall et al., 2020a] have emerged as a popular technique to represent scenes and render novel views from it. Therefore, we explore the potential of neural radiance fields for 3D scene editing in this part. We propose a method that allows to remove objects from a neural radiance field and thereby enabling one category of edits that are required for altering reality.

Background

12.1 Overview

As we have learned above, people have always wanted to change and control reality. With the rise of computer technology, especially in computer vision, this dream has become more attainable than ever before. We have also learned that most edits can be reduced to the form of subtraction and addition. As we focus on object removal in this part, we limit ourselves to discussing the background and literature for object removal. Subtraction requires to cut out an object of the scene and plausible fill the created hole with content such that - ideally - the viewer does not realize that the image has been edited.

12.1.1 Image Inpainting

First works in object removal was based on a stream of research that investigated the task of texture synthesis [Heeger and Bergen, 1995, Efros and Leung, 1999]. *I.e.*, an object gets cut out and the hole is filled with a synthesized texture based on the surroundings of matching texture from somewhere else [Bornard et al., 2002]. [Criminisi et al., 2003] combine texture synthesis with structurally guided inpainting. Yet, all these works look in the same image for patterns and textures that might fit to fill the hole created by removing the object. [Hays and Efros, 2007] is the first data-driven approach that looks for matching patches in millions of images to fill the gap. Another popular approach is [Barnes et al.,

2009a], in which the authors propose to look for matching patches in the same image. The method enables interactive edits of images by random sampling and fast match propagation that exploits the natural coherence of images. Similar to the two previously discussed fields, the field of image editing was turned upside down by deep learning. All of the approaches before the revolution were hand-crafted optimization algorithms with the only way to leverage data are similarities in the same image or in an existing database of images. This changed in 2012. Context encoders [Pathak et al., 2016] first proposed to use neural networks for image inpainting. This was followed by a long line of work that used different neural network architectures and loss functions to better solve the problem [Liu et al., 2018, Zeng et al., 2019]. One big challenge in inpainting is to incorporate both - local and global context. This can be done using dilated convolutions [Iizuka et al., 2017], multiple branches with different receptive fields [Wang et al., 2018], or an attention mechanism [Yu et al., 2018]. One elegant approach is LaMa [Suvorov et al., 2022]. This method leverages Fourier convolutions to encode global context in to the inpainting method. As datasets are ever growing larger, the size of the inpainting networks become larger and larger and thus more powerful. Most recently, an especially data hungry approach took over in the field of inpainting - diffusion models. The learn to denoise a normal Gaussian noise to a real-image. [Lugmayr et al., 2022, Zhang et al., 2023a] showed that this can be applied to image inpainting leveraging the strong priors in these diffusion models. Moreover, it has been shown that the inpainting task can also be guided by language to give the user more control over the inpainting [Xie et al., 2023, Nichol et al., 2021] In addition to RGB image inpainting, the inpainting operation was also extended to depth images by [Fujii et al., 2020]. Despite all this progress in image inpainting, these methods all suffer from the same shortcoming - multi-view consistency. This will be addressed in the next sections.

12.1.2 Video Inpainting

How are objects removed consistently across frames? With more computing power and data, convincingly subtracting content from videos became into the reach of the possible. The goal of this task is to convincingly remove an object

from a video sequence. In image inpainting the created hole only needs to be filled once. However, in video inpainting the hole needs to be consistently filled across all frames of the sequence. The consistency is crucial as inconsistent inpaintings lead to flickering artifacts in the video and the final result is not convincing to the viewer. Thus, the main focus of video inpainting is on this consistency. As with images, there has been an extensive body of software that allowed professionals to labourously edit videos with the desired content. However, this is very time consuming and thus motivated research into automated tooling to speed it up or even fully automate this task. The first step was to extend methods that were used for image inpainting to videos. This was done by [Patwardhan et al., 2005]. [Wexler et al., 2007] also leverages texture synthesis from local patches, but enforces global consistency in the optimization as this is key for video inpainting. [Granados et al., 2012] leverage the fact that moving objects are occluding different parts of the scene at different times. This fact can be used for inpainting by propagating the information about occluded regions from frames where they have not been occluded. [Newson et al., 2014] employ another image inpainting technique for video inpainting namely patch-based inpainting. This approach has been extended by a flow-based term in the cost function by [Huang et al., 2016] leading to more consistent results. Due to the complexity of handling video data, it took more time until learning-based approaches tackled the problem of video inpainting. Not until 2019 was deep learning considered for the task, when [Wang et al., 2019a, Kim et al., 2019] proposed to leverage neural networks for the task. This is typically achieved with guidance from nearby frames, *e.g.* via estimating flow [Gao et al., 2020, Xu et al., 2019b, Huang et al., 2016, Li et al., 2022d], sometimes using depth [Liao et al., 2020, Bešić and Valada, 2022], or an attention mechanism [Zeng et al., 2020]. The approaches presented so far solve the inpainting problem in an offline manner. [Herling and Broll, 2014] proposes a video inpainting method that allows for the manipulation of live video streams. Perhaps counter-intuitively, moving objects make the task easier, since their movement disoccludes the background, making most parts of the scene visible in at least one of the frames. This is not the case for static objects.

Removing Static Objects from Videos. Where occluded pixels are visible in

other frames in the sequence, these can be used to inpaint regions [Lepetit et al., 2001, Mori et al., 2020, Mori et al., 2022]. For example, [Liu et al., 2020b, Liu et al., 2021b] remove static foreground distractors from videos, *e.g.* fences and raindrops. However, there are typically still pixels which cannot be seen in other views, for which some other method is required to fill them in. For example, [Herling and Broll, 2010] propagates patches from visible pixels into the region to inpaint, and [Mori et al., 2020, Thonat et al., 2016] inpaint missing pixels via PatchMatch. Kim *et al.* [Kim et al., 2021] rely on a pre-computed mesh of each scene for object removal. Our key difference to these methods is that our inpaintings can be extrapolated to novel viewpoints.

12.1.3 3D Editing

We already discussed above that editing images or videos has inherent limitations. Editing images allows us to only edit reality from a single viewpoint. Editing videos allows us to edit reality along a fixed trajectory. While there are use cases for both of these problems, ideally we want to alter reality in a way such that a user can explore it from any view point along any trajectory. This requires to create 2D renderings (images) from a 3D scene with ideally known lighting. Thus, altering this, requires 3D editing. There are works that propose the completion of scenes on a geometric and semantic level [Song et al., 2017, Dai et al., 2018]. However, for altering reality this is not sufficient and oftentimes these works try to reconstruct the missing object. In order to alter reality, geometry and appearance have to be inpainted in a plausible way. [Kawai et al., 2015] leverages background geometry for image inpainting yet the inpainting is still in 2D. [Thonat et al., 2016] were one of the first works that tackled this problem by representing the scene as a 3D mesh and solve the inpainting using an EM algorithm based PatchMatch [Barnes et al., 2009b]

Given the limitations of existing 3D editing methods, we propose a NeRF-based scene editing approach in this part. Our approach allows us to a) remove objects from a scene and b) re-render novel views of this scenes.

12.2 Novel View Synthesis

12.2.1 Neural Radiance Fields

Neural radiance fields (or NeRF) [Mildenhall et al., 2020a] is a highly popular image-based rendering method which uses a differentiable volume-rendering formulation to represent a scene; a multi-layer perceptron (MLP) is trained to regress the color and opacity given a 3D coordinate and ray viewing direction. The volume rendering equation is given by

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i \quad \text{where} \quad T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right). \quad (12.1)$$

The camera ray \mathbf{r} is parameterized using origin \mathbf{o} and direction \mathbf{d} as $\mathbf{r} = \mathbf{o} + t\mathbf{d}$. The color \mathbf{c}_i and density are the outputs of the MLP $F(\mathbf{x}_i, \mathbf{d}; \theta)$ at the a particular 3D position $\mathbf{x}_i = \mathbf{o} + t_i\mathbf{d}$ observed from direction \mathbf{d} . The beauty of this formulation is that it can be simply optimized from a set of input images with known poses using a pixel-wise loss function such as

$$\mathcal{L}_2 = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} \|C(\mathbf{r}) - \hat{C}(\mathbf{r})\|_2^2 \quad (12.2)$$

NeRF combined works on implicit 3D scene representations [Liu et al., 2019a, Park et al., 2019, Mescheder et al., 2019, Runz et al., 2020, Saito et al., 2019, Saito et al., 2020, Chibane et al., 2020] discussed in part II, with light-field rendering [Davis et al., 2012] and novel view synthesis [Hedman et al., 2018, Liu et al., 2019b, Xu et al., 2019c]. Extensions include work that reduces aliasing artifacts [Barron et al., 2021], can cope with unbounded scenes [Barron et al., 2022], reconstructs a scene from only sparse views [Chen et al., 2021, Niemeyer et al., 2022, Long et al., 2022, Kim et al., 2022] or makes NeRFs more efficient, *e.g.* by using octrees [Tremblay et al., 2022, Yu et al., 2021] or other sparse structures [Sara Fridovich-Keil and Alex Yu et al., 2021].

Depth-aware Neural Radiance Fields To overcome NeRFs requirement for

dense views and the limits in the quality of the reconstructed geometry, depth can be used in training [Deng et al., 2022, Roessle et al., 2022]. Sparse depth is usually available when a structure from motion framework is used to obtain camera poses [Schönberger and Frahm, 2016, Schönberger et al., 2016]. Alternatively, depth from sensors can be used in implicit representations [Rematas et al., 2022, Sucar et al., 2021].

Object-centric and Semantic NeRFs for Editing One direction of progress in NeRFs is the decomposition of the scene into its constituent objects [Ost et al., 2021, Yang et al., 2021, Wu et al., 2022]. This is done based on motion for dynamic scenes [Ost et al., 2021], or instance segmentation for static scenes [Yang et al., 2021]. Both lines of work also model the background of the scene as a separate model. However, similar to video inpainting, dynamic scenes allow a better modeling of the background since more of it is visible. In contrast, visual artifacts can be seen in the background representation of [Yang et al., 2021, Wu et al., 2022], which model static scenes. Methods that decompose the scene based on semantics [Zhi et al., 2021a, Kobayashi et al., 2022] can also be used to remove objects. However, they do not try to complete the scene when a semantic part is removed and, for example, [Kobayashi et al., 2022] discusses how “the background behind the deleted objects can be noisy or have a hole because it lacks observation”.

12.2.2 Generative Models for Novel View Synthesis

3D aware generative models can be used to synthesize views of an object or scene from different viewpoints, in a 3D consistent manner [Nguyen-Phuoc et al., 2019, Schwarz et al., 2020, DeVries et al., 2021, Rockwell et al., 2021]. In contrast with NeRF models, which only have a test time component and “overfit” to a specific scene, generative models can be used to hallucinate views of novel objects by sampling in the latent variable space. There has also been some interest in 3D generative models that work for full indoor scenes [DeVries et al., 2021, Rockwell et al., 2021, Li et al., 2022c]. However, their capacity to fit the source views (or memorization) can be limited, as shown in the qualitative results of [DeVries et al., 2021]. To train the generative model, [DeVries et al., 2021, Li et al.,

2022c, Rockwell et al., 2021] require a large dataset of indoor scenes with RGB and camera poses and in some cases depth [DeVries et al., 2021, Li et al., 2022c]. In contrast, we use a pre-trained 2D inpainting network, which can be trained on any image, is less dependent on the existence of training data and less constrained to indoor scenarios.

12.2.3 Inpainting in Novel View Synthesis

Inpainting is often used as a *component* of novel view synthesis, to estimate textures for regions unobserved in the inputs [Shih et al., 2020, Wang et al., 2022a], *e.g.* for panorama generation [Koh et al., 2021, Hsu et al., 2021]. [Philip and Drettakis, 2018] enable object removal from image-based-rendering, but with an assumption that background regions are locally planar. Similarly to our approach, two concurrent works [Liu et al., 2022a, Mirzaei et al., 2023] leverage single image inpaintings to remove objects from NeRFs. To deal with multi-view inconsistencies, [Liu et al., 2022a] manually selects a single view to use in the NeRF optimization inside the mask, while [Mirzaei et al., 2023] uses a perceptual loss.

Removing Objects from Scenes

Removing objects from 3D scenes and coupling it with novel view synthesis for use-cases such as augmented reality is challenging for several reasons. First, holes that are created by the removal of the object have to be filled with consistent geometry and appearance. Second, the rendered images have to be multi-view consistent and be of high quality in order to provide a convincing and immersive experience to a user.

In this chapter, we approach the problem of jointly removing objects from scenes and render novel views using neural radiance fields. Since the initial publication of Neural Radiance Fields (NeRFs) [Mildenhall et al., 2020a], there has been an explosion of extensions to the original framework, *e.g.* [Barron et al., 2021, Barron et al., 2022, Chen et al., 2021, Deng et al., 2022, Kim et al., 2022, Liu et al., 2021a, Long et al., 2022, Mildenhall et al., 2020a]. NeRFs are being used beyond the initial task of novel view synthesis. It is already appealing to get them into the hands of non-expert users for novel applications, *e.g.* for NeRF editing [Yuan et al., 2022] or live capture and training [Müller et al., 2022], and these more casual use cases are driving interesting new technical issues.

One of those issues is how to seamlessly remove parts of the rendered scene as described above. Removing parts of the scene can be desirable for a variety of reasons. For example, a house scan being shared on a property selling website

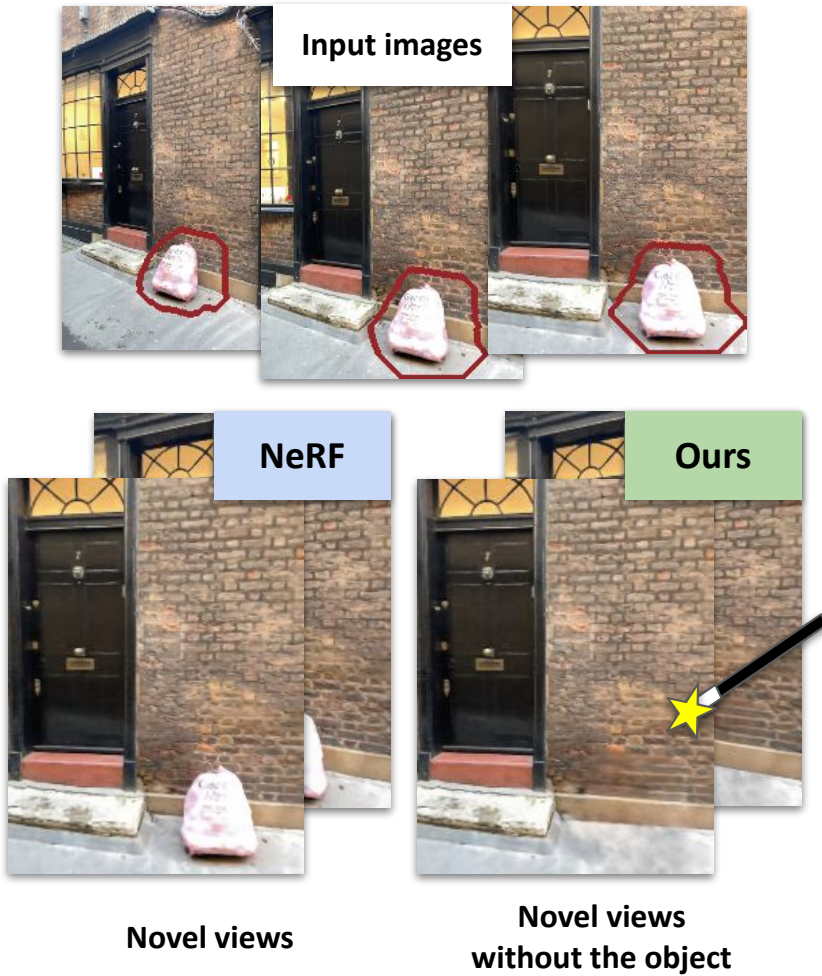


Figure 13.1: Removal of unsightly objects. Our method allows for objects to be plausibly removed from NeRF reconstructions, inpainting missing regions whilst preserving multi-view coherence.

may need unappealing or personally identifiable objects to be removed [Sulaiman et al., 2020]. Similarly, objects could be removed so they can be replaced in an augmented reality application, *e.g.* removing a chair from a scan to see how a new chair fits in the environment [Ozturkcan, 2021]. Removing objects might also be desirable when a NeRF is part of a traditional computer vision pipeline, *e.g.* removing parked cars from scans that are going to be used for relocalization [Moreau et al., 2022].

Some editing of NeRFs has already been explored. For example, object-centric representations disentangle labeled objects from the background, which allows editing of the trained scene with user-guided transformations [Yang et al., 2021, Wu et al., 2022], while semantic decomposition allows selective editing and transparency for certain semantic parts of the scene [Kobayashi et al., 2022]. However, these previous approaches only augment information from the input scan, limiting their generative capabilities, *i.e.* the hallucination of elements that have not been observed from any view.

In this chapter, we tackle the problem of removing objects from scenes, while realistically filling the resulting holes, as shown in Fig. 13.1. Solving this problem requires: a) exploiting multi-view information when parts of the scene are observed in some frames but occluded in others and, b) leveraging a generative process to fill areas that are never observed. To this end, we pair the multi-view consistency of NeRFs with the generative power of 2D inpainting models [Suvorov et al., 2022] that are trained on large scale 2D image datasets. Such 2D inpaintings are not multi-view consistent by construction, and may contain severe artifacts. Using these inpaintings directly causes corrupted reconstructions, so we design a new confidence-based view-selection scheme that iteratively removes inconsistent inpaintings from the optimization. We validate our approach on a new dataset and show that we outperform existing approaches for novel view synthesis on standard metrics of image quality, as well as producing multi-view consistent results.

In summary, we make the following contributions in this chapter:

- We propose the first approach focusing on inpainting NeRFs by leveraging the power of single image inpainting.
- We introduce a novel view-selection mechanism that automatically removes



Figure 13.2: Per-frame inpainting can give plausible results for each frame, but they are not consistent between viewpoints and sometimes contain severe artifacts corrupting the optimization.

inconsistent views from the optimization.

- We present a new dataset for evaluating object removal and inpainting in indoor and outdoor scenes.

13.1 Method

We assume we are given an RGB-D sequence with camera poses and intrinsics. Depths and poses could be acquired, for example, using a dense structure-from-motion pipeline [Schönberger et al., 2016, Schönberger and Frahm, 2016]. For most of our experiments we capture posed RGB-D sequences directly using Apple’s ARKit framework [Apple, 2017], but we also show that we can relax this requirement through use of a multi-view stereo method to estimate depth for an RGB sequence. Along the way, we also assume access to a per-frame mask of the

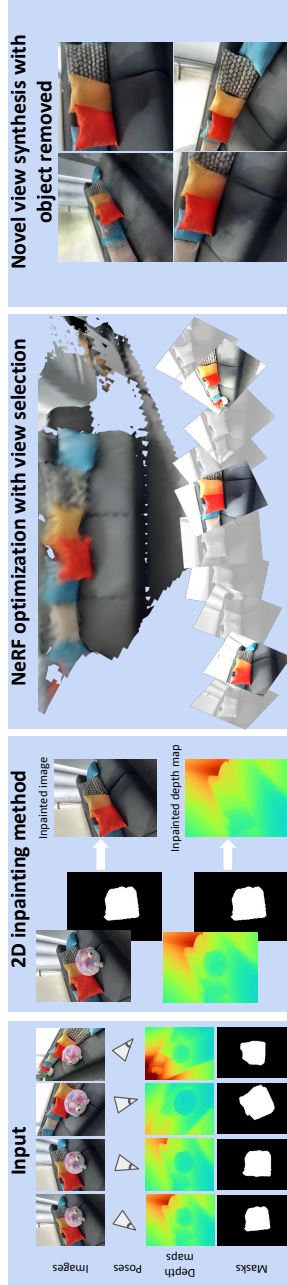


Figure 13.3: An overview of our method We take a sequence of posed RGB-D images together with corresponding 2D masks as input. The 2D frames are inpainted using [Suvorov et al., 2022] and then used to optimize a neural radiance field. During optimization, our confidence-based view selection automatically removes inconsistent views from the optimization preventing unwanted artifacts in the final result. Finally, novel views can be rendered from the scene, where the object has been removed.

object to be removed. The goal is to learn a NeRF model from this input, which can be used to synthesize consistent novel views, where the per-frame masked region should be plausibly inpainted. An overview of our method is shown in Figure 13.3.

13.1.1 RGB and Depth Inpainting Network

Our method relies on a 2D image inpainting method to inpaint each RGB image individually. We motivate this choice with the abundance of training data available for training 2D inpainting methods. By training on this data, the model can incorporate strong priors from the data. Moreover, the inpainting task can be simply modelled by masking out regions in the input image, while keeping the the training data as natural as possible. In contrast, achieving the same results in 3D would require capturing large-scale 3D datasets. Especially the naive masking would not necessarily lead to the same results as the 2D training procedure. *I.e.*, if you randomly mask out you rather model to reconstruct or complete the (partially) removed object than inpaint the background. Thus, we decide to leverage 2D inpainting models for our method. Furthermore, we also require a depth inpainting network. We use both networks as black-boxes and our approach is agnostic to the method chosen. Future improvements in 2D image inpainting can be directly translated to improvements to our method.

Given an image I_n and corresponding mask M_n , the per-image inpainting algorithm produces a new image \tilde{I}_n . Similarly, the depth inpainting algorithm produces an inpainted depth map \tilde{D}_n . We show some results for the 2D inpainting network in Figure 13.2.

13.1.2 Background on NeRFs

Following the original NeRF paper [Mildenhall et al., 2020a], we represent the scene as an MLP F_Θ that predicts color $\mathbf{c} = [r, g, b]$ and density σ , for a 5 dimensional input containing x, y, z position and two viewing directions. The predicted color for pixel \mathbf{r} , $\hat{I}_n(\mathbf{r})$, is obtained by volume rendering along its

associated ray, so

$$\hat{I}_n(\mathbf{r}) = \sum_{i=1}^K \underbrace{T_i(1 - \exp(-\sigma_i \delta_i))}_{w_i} \mathbf{c}_i, \quad (13.1)$$

where

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right). \quad (13.2)$$

K is the number of samples along the ray, t_i is a sample location, $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples, and w_i is the alpha compositing weight which, by construction, sum to less than or equal to 1.

The NeRF loss operates on training images as

$$\mathcal{L}_{\text{RGB}} = \sum_{n=1}^N \sum_{\mathbf{r} \in \Omega_n} \left\| I_n(\mathbf{r}) - \hat{I}_n(\mathbf{r}) \right\|^2, \quad (13.3)$$

where $I_n(\mathbf{r})$ is the input RGB value for pixel \mathbf{r} , and $\hat{I}_n(\mathbf{r})$ is its predicted color. Ω_n indicates the 2D domain of image n . The parameters of the MLP, Θ , are optimized to minimize this loss. Similarly to [Roessle et al., 2022], if input depth is available, then an additional loss can be added,

$$\mathcal{L}_{\text{depth}} = \sum_{n=1}^N \sum_{\mathbf{r} \in \Omega_n} \left| D_n(\mathbf{r}) - \hat{D}_n(\mathbf{r}) \right|, \text{ with } \hat{D}_n(\mathbf{r}) = \sum_{i=1}^K w_i t_i, \quad (13.4)$$

where $D_n(\mathbf{r})$ is the input depth for pixel \mathbf{r} , and $\hat{D}_n(\mathbf{r})$ is the corresponding predicted depth.

Finally, a distortion regularizer loss was introduced in [Barron et al., 2022] to better constrain the NeRF optimization and remove “floaters”. It encourages the non-zero compositing weights w_i to be concentrated in a small region along the ray, so for each pixel \mathbf{r} ,

$$l_{\text{dist}}(\mathbf{r}) = \sum_{i,j} w_i w_j \left| \frac{t_i + t_{i+1}}{2} - \frac{t_j + t_{j+1}}{2} \right| + \frac{1}{3} \sum_i w_i^2 \delta_i. \quad (13.5)$$

13.1.3 Confidence-based View Selection

Despite most of the individual inpainted RGB images \tilde{I}_n looking realistic, they still suffer from two issues: 1) some of the inpaintings are incorrect, and 2) despite individual plausibility, they are not multi-view consistent, *i.e.* the same area observed in multiple views is not necessarily completed in a consistent way (Figure 13.2). For this reason, we propose a confidence-based view selection scheme, that automatically chooses which views are used in the NeRF optimization. We associate to each image I_n a non-negative uncertainty measure u_n . The corresponding per-image confidence, e^{-u_n} , is used to re-weight the NeRF losses. This confidence value can be seen as a loss attenuation term, similar to the aleatoric uncertainty prediction term in [Kendall and Gal, 2017].

The RGB loss for our model is then set out as

$$\mathcal{L}_{\text{RGB}}^{\mathcal{P}}(\hat{I}) = \sum_{n=1}^N \sum_{\mathbf{r} \in \Omega_n \setminus M_n} \left\| I_n(\mathbf{r}) - \hat{I}_n(\mathbf{r}) \right\|^2 + \sum_{n \in \mathcal{P}} e^{-u_n} \sum_{\mathbf{r} \in M_n} \left\| \tilde{I}_n(\mathbf{r}) - \hat{I}_n(\mathbf{r}) \right\|^2 \quad (13.6)$$

where the color for pixels \mathbf{r} is supervised by the inpainted image for pixels inside the mask, and by the input RGB image for pixels outside the mask. Note that the second term of this loss is only computed over a restricted set of images \mathcal{P} , where $\mathcal{P} \subseteq \{1, \dots, N\}$. This is indicated by the superscript \mathcal{P} in the loss term $\mathcal{L}_{\text{RGB}}^{\mathcal{P}}$. In practice, that means that only some inpainted regions are used in the NeRF optimization. We discuss below how we choose the set \mathcal{P} .

We use a similar split into pixels inside and outside the mask for the depth loss, so

$$\mathcal{L}_{\text{depth}}^{\mathcal{P}} = \sum_{n=1}^N \sum_{\mathbf{r} \in \Omega_n \setminus M_n} \left| D_n(\mathbf{r}) - \hat{D}_n(\mathbf{r}) \right| + \sum_{n \in \mathcal{P}} e^{-u_n} \sum_{\mathbf{r} \in M_n} \left| \tilde{D}_n(\mathbf{r}) - \hat{D}_n(\mathbf{r}) \right|. \quad (13.7)$$

Finally, we include two regularizers. One is on the uncertainty weights $\mathcal{L}_{\text{reg}}^{\mathcal{P}} = \sum_{n \in \mathcal{P}} u_n$, to prevent a trivial solution where e^{-u_n} is 0. The other is a distortion regularizer, based on [Barron et al., 2022], using the loss detailed in

Equation (13.5), so

$$\mathcal{L}_{\text{dist}}^{\mathcal{P}} = \sum_{n=1}^N \sum_{\mathbf{r} \in \Omega_n \setminus M_n} l_{\text{dist}}(\mathbf{r}) + \sum_{n \in \mathcal{P}} \sum_{\mathbf{r} \in M_n} l_{\text{dist}}(\mathbf{r}). \quad (13.8)$$

View Direction and Multi-view Consistency. When optimizing the NeRF, we made three observations: a) the multi-view inconsistencies in the inpaintings are modelled by the network using the viewing direction; b) we can enforce multi-view consistency by removing the viewing direction from the input; and c) the inconsistencies introduce cloud-like artifacts in the density when not using the viewing direction as input. To prevent a) and c) and correctly optimize the variables u_n that capture the uncertainty about the inpaintings \tilde{I}_n , we propose: 1) adding an auxiliary network head, F_{Θ}^{MV} , to the NeRF that does not take the viewing direction as input and, 2) stopping the gradient from the color inpainting and F_{Θ}^{MV} to the density, leaving the uncertainty variable u_n as the only view-dependent input. This design forces the model to encode the inconsistencies between inpaintings into the uncertainty prediction while keeping the model consistent across views. F_{Θ}^{MV} has a loss term based on Equation 13.6: $\mathcal{L}_{\text{RGB}}^{\mathcal{P}}(\hat{I}^{MV})$. The outputs of the auxiliary head F_{Θ}^{MV} are not used in the final rendering. Instead, the loss associated to this extra head is a regularisation term. See Figure 13.4 for an illustration of our architecture.

Our final loss is then

$$\mathcal{L}^{\mathcal{P}} = \lambda_{\text{RGB}} \mathcal{L}_{\text{RGB}}^{\mathcal{P}}(\hat{I}) + \lambda_{\text{RGB}} \mathcal{L}_{\text{RGB}}^{\mathcal{P}}(\hat{I}^{MV}) + \lambda_{\text{depth}} \mathcal{L}_{\text{depth}}^{\mathcal{P}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}^{\mathcal{P}} + \lambda_{\text{dist}} \mathcal{L}_{\text{dist}}^{\mathcal{P}}, \quad (13.9)$$

which is optimized over the MLP parameters $\Theta = \{\Theta^{\sigma}, \Theta^c, \Theta^{MV}\}$ and the uncertainty weights $\mathbf{U}^{\mathcal{P}} = \{u_n, n \in \mathcal{P}\}$. The confidence of all images is initialized to 1, *i.e.* $u_n := 0$.

Iterative Refinement. We use the predicted per-image uncertainty, u_n , in an iterative scheme that progressively removes non-confident images from the NeRF optimization, *i.e.* we iteratively update the set \mathcal{P} of images that contribute to the loss in masked regions. After K_{grad} steps of optimizing $\mathcal{L}^{\mathcal{P}}$, we find the median estimated confidence value m . We then remove from the training set all 2D

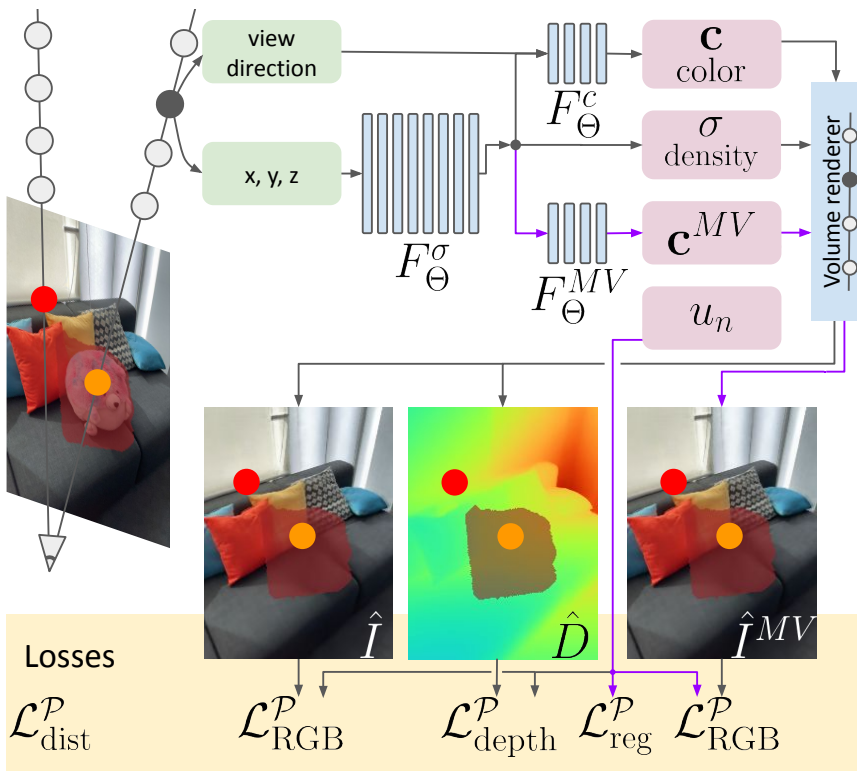


Figure 13.4: Full network architecture. The backbone of our NeRF is a MLP consuming the 3D coordinate as input. This branches into two parts, where one takes the view-direction as input and the other does not. The output are the density, a view-dependent and view-independent color. These outputs are used to volume render two images. The one is view-dependent while the other \hat{I}^{MV} is multi-view consistent. The losses $\mathcal{L}_{\text{RGB}}^{\mathcal{P}}$ and $\mathcal{L}_{\text{depth}}^{\mathcal{P}}$ are weighted by the view-specific uncertainty variable u_n , which only receives gradient from \hat{I}^{MV} .

Algorithm 1: Iterative refinement using confidence based view selection.

Data: Input images I_n , Inpainted images \tilde{I}_n , Depth maps D_n , Inpainted depth maps \tilde{D}_n , Masks M_n

Result: Trained NeRF model F_Θ with object removed

```

/* Set of images used for training NeRF is initialized with all
images. */
 $\mathcal{P} \leftarrow \{1, \dots, N\}$  and  $u_n \leftarrow 0, n \in \mathcal{P}$ 
for  $i \leftarrow 0$  to  $K_{\text{outer}}$  do
     $\Theta \leftarrow$  randomly initialized
    /* Gradient iterations of NeRF training. */
    for  $j \leftarrow 0$  to  $K_{\text{grad}}$  do
         $\Theta \leftarrow \Theta - \nabla_{\Theta} \mathcal{L}^{\mathcal{P}}$ 
         $\mathbf{U}^{\mathcal{P}} \leftarrow \mathbf{U}^{\mathcal{P}} - \nabla_{\mathbf{U}^{\mathcal{P}}} \mathcal{L}^{\mathcal{P}}$ 
        /* Calculate median of confidence values. */
         $m = \text{median}(\{e^{-u_n}, n \in \mathcal{P}\})$ 
        /* Update  $\mathcal{P}$  by removing images with small confidence. */
        for  $n \in \mathcal{P}$  do
            if  $e^{-u_n} < m$  then
                 $\mathcal{P} \leftarrow \mathcal{P} \setminus n$ 
    
```

inpainted regions which have associated confidence scores less than m . We then retrain the NeRF with the updated training set, and repeat these steps K_{outer} times. Note that images excluded from \mathcal{P} still participate in the optimization, but only for rays in the unmasked regions as they contain valuable information about the scene. This is summarized in Algorithm 1.

13.1.4 Implementation Details

Masking the Object to be Removed. Similarly to other inpainting methods, our method requires per-frame masks as input. Manually annotating each frame with a 2D mask, as done in other inpainting methods [Suvorov et al., 2022, Xu et al., 2019b, Li et al., 2022d], is time consuming. Instead, we manually annotate a 3D box that contains the object using MeshLab [Cignoni et al., 2008] to visualise and annotate a 3D point cloud. This only has to be done once per scene. Alternatively, we could have relied on 2D object segmentation methods, e.g. [He et al., 2017], or 3D object bounding box detectors, e.g. one of the baselines in [Ahmadyan et al., 2021].

Mask Refinement In practice, we observe that masks obtained from the annotated 3D bounding boxes can be quite coarse and include large parts of the background.

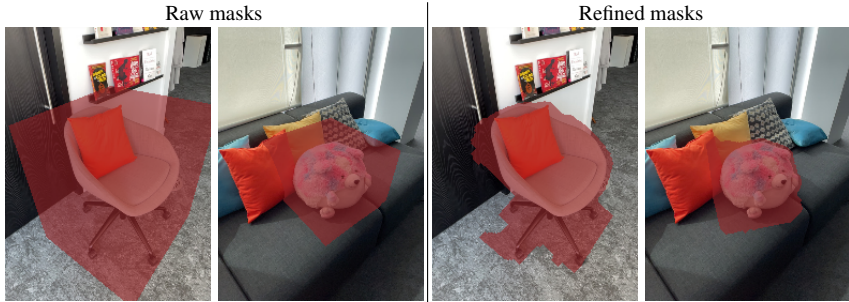


Figure 13.5: Mask refinement Our mask refinement leads to smaller masks and therefore higher quality inpainting.

Since large masks have a negative effect on the inpainting quality, we propose a mask refinement step to obtain masks which are tighter around the object. This step is not required if input masks are already tight. Intuitively, this mask refinement step removes parts of the 3D bounding box that are empty space. We start by taking all points in the reconstructed 3D point cloud that are inside the 3D bounding box. The refined mask is then obtained by rendering these points into each image and performing a simple comparison with the depth map to check occlusions in the current image. The resulting mask is cleaned up by dilating any pixel leaks caused by sensor noise using binary dilation and erosion. The effect of our mask refinement step can be seen in Figure 13.5.

Inpainting Network We used [Suvorov et al., 2022] for inpainting both RGB and depth. The inpainting of RGB images and depth maps is done independently and we use the reference network provided by the authors of [Suvorov et al., 2022] for both. Our depth maps are preprocessed by clipping to 5 m and linearly mapping depths in $[0 \text{ m}, 5 \text{ m}]$ to pixel values of $[0, 255]$. We observed empirically that this approach provided good results, but an inpainting method specific for depth maps could improve over this baseline.

NeRF Estimation The implementation of our method is built upon [Niemeyer et al., 2022] and [Barron et al., 2021]. We weight the terms in the loss function with $\lambda_{\text{RGB}} = \lambda_{\text{depth}} = \lambda_{\text{dist}} = 1$, and $\lambda_{\text{reg}} = 0.005$. We do a filtering step, where we remove low confidence images every $K_{\text{grad}} = 50,000$ steps, resulting in $K_{\text{outer}} = 4$ filtering steps. Timings for our method are comparable to those of a

standard NeRF, in our case [Niemeyer et al., 2022] and [Barron et al., 2021].

13.2 Experiments

13.2.1 Datasets

While previous approaches have tackled static object removal from videos, no standard dataset/metrics to evaluate these systems has been proposed, to our knowledge. This work introduces an RGB-D dataset of real scenes, designed to evaluate the quality of object removal. Our dataset has two variants, which are used differently when benchmarking. Please see Appendix B for a full overview about our dataset.

Ours - Real Objects. This dataset comprises 17 scenes focusing on a small area with one object of interest. They vary in difficulty in terms of background texture, size of the object, and complexity of scene geometry. For each scene, we collected two sequences, one with and the other without the object that we want to remove. The sequences are collected using ARKit [Apple, 2017] on an iPhone 12 Pro with Lidar, and contain RGB-D images and poses. The masks are annotated and refined as described in Section 13.1.4. For each scene, we use the sequence with the object and corresponding masks for training the NeRF model, and the sequence without the object for testing. The use of real objects makes it easier to evaluate how the systems deal with real shadows and reflections, as well as novel view synthesis.

Ours - Synthetic Objects. Most video and image inpainting methods, *e.g.* [Suvorov et al., 2022, Li et al., 2022d], do not perform novel view synthesis, meaning such methods cannot be fairly evaluated on our ‘Real objects’ dataset. We therefore introduce a separate synthetically-augmented variant of our dataset. This uses the same scenes as the real objects dataset, but we only use the sequence without the object. We then manually position a 3D object mesh from ShapeNet [Chang et al., 2015] in each scene. The object is placed so that it has a plausible location and size, *e.g.* a laptop on a table. The masks are obtained by projecting the mesh into the input images, which is the only use we make of the 3D object mesh. For this synthetic dataset, following *e.g.* [Barron et al., 2022], we use every

	Synthetic objects - Masked					Real objects - Masked				
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	Depth L_1 \downarrow	Depth L_2 \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	Depth L_1 \downarrow	Depth L_2 \downarrow
Image and video inpainting baselines										
LatMa [Suvorov et al., 2022] [†]	27.999	0.898	0.060	0.070	0.075	-	-	-	-	-
E ² FCVI [Li et al., 2022a] ^{†‡}	24.568	0.874	0.102	-	-	-	-	-	-	-
3D scene completion baselines										
PixelSynth [Rockwell et al., 2021] [†]	25.481	0.887	0.116	-	-	25.438	0.851	0.152	-	-
CompNVS [Li et al., 2022c]	17.389	0.823	0.171	1.697	3.641	-	-	-	-	-
Object compositional NeRF [Yang et al., 2021] [*]	-	-	-	-	-	21.757	0.836	0.134	0.312	0.341
Ablations										
Masked NeRF	26.126	0.882	0.093	0.084	0.105	21.644	0.815	0.142	0.096	0.054
Inpainted NeRF	28.760	0.905	0.086	0.278	0.400	23.705	0.848	0.134	0.145	0.121
Inpainted NeRF + inpainted Depth	27.568	0.898	0.094	0.231	0.318	23.652	0.844	0.136	0.313	0.387
Ours - no depth	28.290	0.906	0.079	0.296	0.335	24.228	0.848	0.130	0.345	0.288
Ours - depth predicted using [Sayed et al., 2022]	26.540	0.895	0.087	0.112	0.118	25.010	0.856	0.128	0.142	0.140
Our method	29.437	0.916	<u>0.078</u>	0.069	<u>0.096</u>	<u>25.271</u>	0.859	0.125	0.071	0.044

Table 13.1: Comparison with baselines and state of the art methods. Our method is either **best** or **second-best** compared to other novel-view synthesis baselines in inpainting the missing regions of the scene, by propagating multi-view information and leveraging 2D inpainting information. Notes: [†]These methods can't be evaluated on the proposed real dataset as they do not synthesize novel views. [‡]These methods do not produce depth maps. ^{*}[Yang et al., 2021] requires the actual object therefore it cannot be evaluated on the proposed synthetic dataset.

8th frame for testing and the rest of the frames for training the NeRF model.

ARKitScenes. We further validate our approach qualitatively on ARKitScenes [Baruch et al., 2021a]. This is an RGB-D dataset of 1,661 scenes, where depth was captured via iPhone Lidar.



Figure 13.6: Results on ARKitScenes [Baruch et al., 2021a]. We can successfully remove objects from casually captured sequences in indoor scenes.

13.2.2 Metrics

To evaluate the object removal and inpainting quality, we compare our system’s output image against the ground truth image, for each test image in the dataset. All metrics in the paper are only computed inside the masked region. We use the three standard metrics for NeRF evaluation [Mildenhall et al., 2020a]: PSNR [Hore and Ziou, 2010], SSIM [Wang et al., 2004] and LPIPS [Zhang et al., 2018].

To evaluate the geometric completion, we compute the L_1 and L_2 error between the rendered and the ground-truth depth maps inside masked regions. The metrics are averaged over all frames of a sequence, and then averaged over all sequences.

13.2.3 Ablations and Comparison with Baselines

In Table 13.1, we compare our approach with alternative methods for object removal, with a focus on methods that use an underlying NeRF representation.

Image and Video Inpainting Baselines. We compare with two state-of-the-art methods for image LaMa [Suvorov et al., 2022] and video inpainting E²FGVI [Li et al., 2022d]. In both cases, we use their reference implementation and provided trained network. Neither of these methods allows novel view synthesis, so they are only evaluated on the synthetic objects dataset.

3D Scene Completion Baselines. We compare with several published works for 3D scene completion. Note that none of these baselines specifically targets inpainting, so results should be viewed with this in mind. For all of them, we use their publicly available implementation. PixelSynth [Rockwell et al., 2021] and CompNVS [Li et al., 2022c] were both proposed for scene *outpainting*. Given one or a few frames of a scene their goal is to complete the scene to enable novel view synthesis. Both of these methods rely on a generative model of indoor scenes and neither requires test-time optimization. Both methods are adapted to use our masks as input. Object compositional NeRF [Yang et al., 2021] editing of objects in NeRFs via pose transformations. We adapt their code for object *removal* by setting a transformation that moves the object outside the camera’s field of view.

Ablations. We compare different ablations of our method, including different ways of training a baseline NeRF model. Masked NeRF corresponds to training a NeRF using the full input RGB-D data, but pixels and depths in the masked regions are ignored in the NeRF losses. Inpainted NeRF is a NeRF trained with all inpainted images, but not using the inpainted depth maps, while Inpainted NeRF + Inpainted Depth uses all the inpainted images and inpainted depth maps. This baseline corresponds to “All views” in Table 13.2. We also present results for our method, *i.e.* training the NeRF with the confidence-based view selection step, but without using depth maps as input (“Ours - no depth” and using depth maps

from a state-of-the-art multi-view depth prediction method [Sayed et al., 2022] to show that we do not necessarily rely on sensor depth (“Ours - depth predicted using [Sayed et al., 2022]”).

Finally, “Our method” is our proposed approach, which uses the method described in Section 13.1.

As shown in Table 13.1, our method is superior to other novel-view synthesis baselines across most appearance and depth metrics. Moreover, as opposed to the single image inpainting LaMa [Suvorov et al., 2022], our method is close to multi-view consistent, significantly reducing inter-frame flickering. Our method also outperforms the naive baselines, which train a NeRF with a masked version of the image, or all the inpainted images. Training our method without using depth maps leads to comparable performance in terms of image metrics, while the depth metrics are considerably worse indicating a degrade in the quality of the recovered 3D shape. Using predicted depth maps from [Sayed et al., 2022] gives competitive results, while pointing to an interesting direction for future research.

Qualitative Comparison. In Figure 13.7, we show that the proposed method successfully removes the selected object compared to the baselines. While Masked NeRF fails to complete large holes and Inpainted NeRF suffers from bad inpaintings in the training set, our method can leverage the 2D inpaintings, while avoiding integrating artifacts by removing those input frames. In contrast to Object compositional NeRF [Yang et al., 2021], leveraging the inpaintings also helps to mitigate the appearance of artifacts below the object’s surface. Compared to [Yang et al., 2021] and [Rockwell et al., 2021], our method is better able to generate plausible scene completions. We also show results from the ARKitScenes dataset in Figure 13.6.

View Selection Ablation Here we validate that our view selection procedure from Section 13.1.3 contributes to improved results. We compare our method with different view selection strategies in Table 13.2 on our synthetic object dataset. “All views” uses all the inpainting views when training the NeRF model. The other baselines use a subset of views to train the NeRF model, spaced at regular intervals: every 10th frame for 1/10th; every 50th frame for 1/50th, and single middle frame for Single view. The number of views used for each sequence

Method	# of views	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	L_1 \downarrow	L_2 \downarrow
All views	82 - 382	27.568	0.898	0.094	0.231	0.318
1/10th	8 - 38	27.098	0.900	0.079	0.202	0.291
1/50th	1 - 7	26.718	0.893	0.087	0.229	0.309
Single view	1	26.232	0.892	0.079	0.133	0.198
Ours	10 - 185	29.437	0.916	0.078	0.069	0.096

Table 13.2: Ablation on view selection methods. We validate our view selection formulation by comparing to alternative approaches. Ours consistently produces better performing models.

varies depending on the length of the sequence. We outperform these baselines, suggesting that our proposed strategy for view selection is effective in choosing a good set of views to include in the NeRF optimization.

13.3 Discussion

13.3.1 Limitations

The main limitation of our method is that it is upper bounded by the performance of the 2D inpainting method. Context is important for inpainting models as the 2D context is the main responsible for defining how the hole in the image is inpainted. Therefore, the 2D inpainting fails if the masks are too large not enough context is provided to do plausible hole filling. If this is the case along the entire trajectory, our method cannot inpaint the hole and the object removal fails as there needs to be a minimum number of inpainted images that also agree with each other to successfully remove the object from the scene.

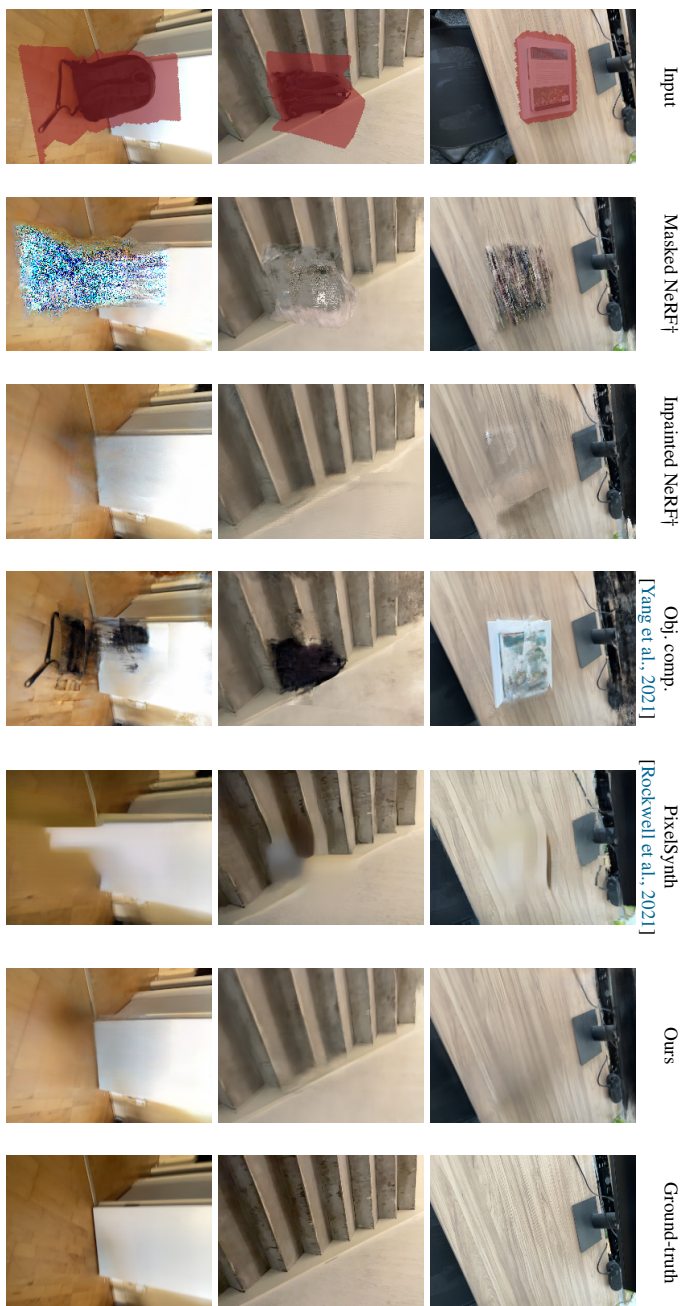
Second, the 2D inpainting networks do not consistently inpaint high-frequency textures. This is especially visible when looking at inpaintings of the inpainted frames. The inconsistency manifests itself by flickering high-frequency between frames of the sequence. Due to this flickering, our renderings sometimes suffer from blurring as the optimization can only resolve this low-level inconsistency by averaging the information from the different views.

Third, cast shadows or reflections of the object are not handled well. The chosen inpainting method [Suvorov et al., 2022] is remarkably good in reconstruct

shapes from its shadows. This leads that the reconstruction is sometimes contaminated by shadow artifacts that are introduced by the 2D model. This problem can be naively mitigated by increasing the mask. However, the larger the mask the higher the risk that we start to suffer from inpainting failure. Moreover, reflections are not handled with our method.

13.3.2 Summary

We presented a framework to train neural radiance fields, where objects are plausibly removed from the output renderings. Our method draws upon existing work in 2D inpainting, and introduces an automatic confidence-based view selection scheme to select single-view inpaintings with multi-view consistency. We experimentally validated that our proposed method improves novel-view synthesis from 3D inpainted scenes compared to existing work, despite suffering from blurring. We have also introduced a dataset for evaluating this work, which sets a benchmark for other researchers in the field.



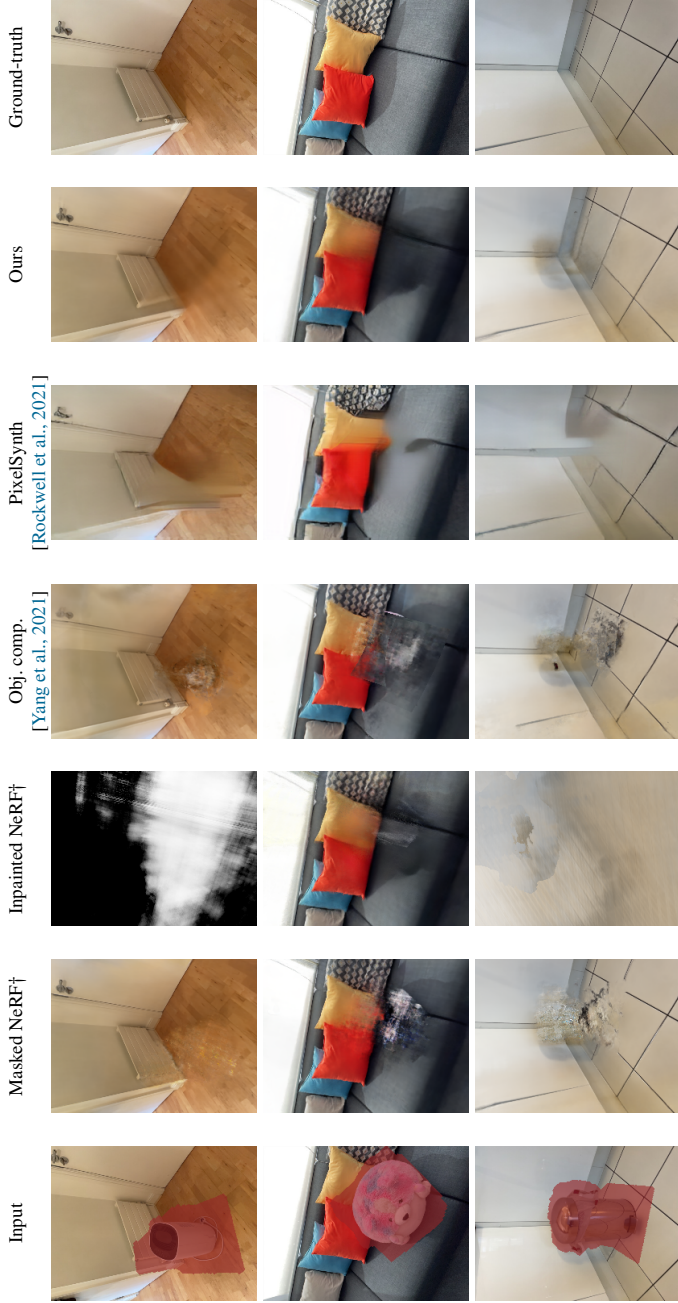


Figure 13.7: Qualitative comparisons with baseline. Our method significantly improves over 3D scene completion baselines for the task of removing objects from a scene using 3D inpainting. Further, it mitigates artifacts and stabilizes convergence compared to the inpainting baseline without automatic view selection. †: We also compare with two NeRF based baselines: Masked NeRF and Inpainted NeRF.

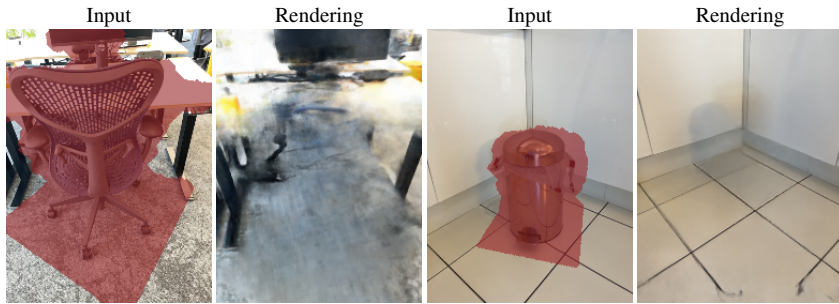


Figure 13.8: Failure cases and limitations. Our method can not recover when the 2D inpainting method fails all the frames, for example when the mask and covers a large part of the image. Further, our method keeps the shadows of the removed object, if they are not included in the object mask.

Part V

Conclusion

Summary

In conclusion, we explored three fundamental pillars within of augmented reality pipelines: 3D scene reconstruction, 3D scene understanding, and 3D scene editing. Reconstruction is the foundation of most AR applications providing essential spatial awareness. This spatial awareness, in turn, underpins critical functionalities like interaction and creation of immersive user experiences. Understanding, as the second pillar, proves indispensable for the creation of experiences in the physical world that necessitate the generation of contextually relevant content for augmented reality users. Lastly, 3D scene editing serves as the third essential component, allowing for alterations of reality in various forms. This capability finds application in diverse scenarios, ranging from visualizing new furniture arrangements in a living room to crafting captivating augmented reality gaming experiences. In concert, these three pillars represent the building blocks that enable augmented reality technologies to seamlessly integrate with our daily lives, transforming how we perceive and interact with the world around us.

3D Scene Reconstruction In the part dedicated to 3D scene reconstruction, our primary focus revolved around addressing the challenge of real-time scene updates using a continuous stream of RGB-D sensor data. We leveraged the power of machine learning for depth map fusion, demonstrating its superior accuracy compared to conventional hand-crafted methods. We started by addressing the challenge of handling depth-dependent noise that is common in depth sensing technology from a machine learning perspective while keeping the overall lightweight

and efficient. To this end, we extracted the current state of the reconstruction that is observed by the new measurement into a view-dependent local grid, update this local grid with the new information, and integrate the updated local geometry into the global scene. We showed that fusing depth maps using this approach increases accuracy compared to hand-crafted methods such as [Curless and Levoy, 1996a]. Further, we tackled the persistent issue of outliers contaminating reconstructed geometry, a common affliction of 3D reconstruction algorithms like two-view dense stereo. This approach involved moving the fusion process to a latent space, subsequently translating this latent representation into the final output geometry using neural networks. This technique not only facilitated the fusion of all available information but also efficiently eliminated outliers during the translation process. Remarkably, our method achieved a significant reduction in outlier contamination without compromising the completeness of the reconstruction, a persistent challenge encountered by traditional post-processing-based outlier removal methods. Furthermore, we ventured into the territory of efficient and accurate RGB-D fusion. Here, we introduced a scene representation termed "DeepSurfels," combining the efficiency of surfels with learned features. Complementing this novel representation, we proposed a learning-based fusion pipeline to seamlessly integrate RGB-D data into the DeepSurfels scene representation. Our empirical experiments confirmed the effectiveness of this combined approach, highlighting the effectiveness of DeepSurfels in enhancing the quality of the rendered images.

3D Scene Understanding. In the realm of 3D scene understanding, we confronted two pivotal questions. Firstly, we grappled with the challenge of automating the scaling of annotations for 3D scene understanding datasets. Our solution entailed the development of an automated annotation pipeline that commenced by generating semantic predictions for individual frames through state-of-the-art segmentation models. These predictions were subsequently merged into a unified consensus prediction. Navigating the complexity of varying label spaces employed by different models, we carefully crafted a framework for mapping these diverse label spaces into a unified label space. This endeavor involved two critical components: a) the deliberate design of a unified label space, which not

only resolved ambiguities but also boasted high resolution to capture the intricate nuances of the real world, and b) the construction of label space translations to align each unique label space with the unified one. To lift our approach to three dimensions, we mapped the 2D consensus into 3D, introducing an additional layer of aggregation and denoising. This lifting also facilitated the creation of multi-view consistent 2D maps through the rendering of 3D annotations back into 2D views. Our empirical evaluation demonstrated that the proposed pipeline not only rivaled human-annotated labeling approaches but also significantly outperformed them when human labels were integrated into the pipeline. This method empowers the scalability of 3D semantic segmentation to large-scale datasets, enabling the training of expansive models and the evaluation of existing ones with unprecedented efficiency and accuracy.

Secondly, we tackled the challenge of incremental semantic mapping with only using local information. To this end, we proposed a spatio-temporal expert network that combines information from the 2D and 3D domain with existing information stored in the learned scene representation. The key is that incremental mapping methods have to incorporate an as large receptive field as possible yet they should be efficient in terms of runtime and memory. Therefore, we kept the 3D processing local and avoid expensive recomputation of intermediate representations across the entire scene. Instead, we extracted valuable global context from the current 2D view, refined it in the local 3D window and combined it with previously aggregated information. In an extensive experimental evaluation, we showed that this approach outperforms all existing local methods while only being slightly inferior to state-of-the-art methods that all require global context to achieve their results.

3D Scene Editing. Within the domain of 3D scene editing, we tackled the challenge of object removal from neural radiance fields - a widely popular scene representation known for its versatility in novel view synthesis, geometry reconstruction, and appearance modeling. Numerous practical scenarios call for the ability to seamlessly eliminate objects from reconstructed scenes, whether driven by privacy concerns, the desire to remove unsightly elements, or the need to focus on the fundamental underlying geometry (e.g., the room).

To this end, we introduced a method that automates the removal of objects from these neural radiance fields. Leveraging strong priors derived from 2D inpainting models, we designed on a two stage process. Initially, we inpainted 2D images, harnessing the output frames within the optimization of the neural radiance field. However, our initial attempts revealed unsatisfactory results due to inconsistencies among the inpainted frames.

To surmount this challenge, we proposed an uncertainty-based optimization framework. This framework assigned a confidence value to each view, encapsulating the consistency of the inpainted region with all other frames. Based on this confidence measure, we automatically identified and excluded frames that exhibited inconsistencies with the overall set. To evaluate this newly introduced task, we further introduced a novel dataset. The goal is to have accurate ground-truth for the removal task. Therefore, we captured RGB-D sequences using a custom scanning app. To simulate the removal of objects, we scanned the scene with the object, removed the object while keeping the scanning session running, and re-scanned the scene to capture the ground-truth for the removal task. Our empirical experiments on this dataset compellingly demonstrated the effectiveness of our proposed removal approach, showcasing significant improvements over existing methods and solidifying its contribution to the field of 3D scene editing.

Future Work

At the start of this thesis, we illuminated the boundless possibilities that augmented reality can unlock, underscoring its potential to reshape our daily experiences. We identified three fundamental pillars — 3D scene reconstruction, 3D scene understanding, and 3D scene editing — as the key to enable these transformative capabilities. Throughout this thesis, we have diligently tackled specific challenges within each of these subfields, contributing novel solutions and insights. Yet, it is crucial to acknowledge that the path toward the widespread integration of augmented reality into our everyday lives remains strewn with open questions and uncharted territories.

15.1 3D Scene Reconstruction

In part II, we have explored the application of machine learning to the problem of incremental RGB-D fusion to generate a globally consistent 3D model.

Scaling up learning. We showed the potential of using machine learning and that it generalizes well to unseen data. However, one main limitation of the proposed methods that we train on synthetic data only and require 3D ground-truth for training. Recent advances in machine learning have shown the power of large-scale training. Thus, an interesting avenue would be to learn the incremental mapping easily available RGB stream through coupling explicit mapping with with 2D self-supervision. This would allow learn incremental dense 3D reconstruction in an end-to-end framework with RGB as input and a dense 3D reconstruction as

output while only requiring 2D frames for supervision.

Test-time optimization A second interesting direction would be incorporating some form of test-time optimization into the fusion process. While strong priors can be learned from training on datasets, it is oftentimes reasonable to adjust the reconstruction that it better fits the input data. Signals that can be used for self-supervision mentioned above could also be used for optimizing the scene at test time to yield a higher accuracy of the reconstruction. This might become more and more popular by the increased power of compute systems available on portable devices.

15.2 3D Scene Understanding

In part III, we have addressed the challenge of automating data annotation and incremental semantic mapping.

Latent annotation pipeline We have proposed a fully automatic labeling pipeline that allows to effectively scale dense semantic 3D annotation to large datasets. One limitation is still that the annotations are fixed to our carefully curated unified label space. However, the set of required labels is not fixed and might change from application to application. One application might consider labels that are focused on indoor living spaces, another application is better served with labels for indoor office spaces, and a third is not interested into object categories but object affordances. Thus, it would be worthwhile to explore a labeling pipeline that can be easily adjustable to the scenario the labels are used in. To this end, it would be an interesting avenue, to tightly couple language features with 3D data similar to [Peng et al., 2023] yet leverage the power of a model ensemble.

Fusing language features The same limitation applies to incremental mapping. All mapping pipelines up to date either segment the scene into a fixed label set or produce weak segmentation results [Jatavallabhula et al., 2023]. Nevertheless, coupling online pipelines with powerful models and features embedding language information is an interesting direction of research. Especially, how can you deploy the full power of language models to systems that have limited compute and memory and require real-time capabilities? This is a question that would be

interesting to explore in the context of incremental semantic mapping.

15.3 3D Scene Editing

In part IV, we proposed a method that removes objects from neural radiance fields. The proposed pipeline mostly suffered from inpainting failures and high-frequency flickering. Thus, resolving these issues would be interesting directions to be explored in future work.

Confidence-based pixel selection and iterative refinement The current pipeline selects the inpainted views on a per-view basis. Yet, the quality of the inpainting is not uniform across the entire inpainted region. Therefore, it would be interesting to explore to optimize a dense pixel-wise confidence-map for each view. This would allow to iteratively shrink the mask produced by the removed object using the rendered image. In return, this information could be used to reduce the difficulty of the inpainting problem. That way, information is effectively propagated in between views and the inpainting can be bootstrapped using existing information.

Interactive editing One limitation of our 3D editing method is the runtime of the underlying neural radiance field used as a scene representation. However, the initially described use of editing in augmented reality oftentimes require on-device online editing to enable truly immersive experience. *I.e.*, you do not want to wait for hours until your living room is cleared to visualize new furniture. Hence, to enable interactive immersive applications, this shortcoming has to be addressed. One first step into this direction are fast neural radiance field architecture such as Instant-NGP [Müller et al., 2022] or just recently released Gaussian Splats [Kerbl et al., 2023]. However, these models are more local representations as opposed to neural radiance fields and the spatial priors are less strong. Therefore, an interesting direction is to couple local representation with 2D priors for object removal and inpainting to enable interactive 3D editing in neural radiance fields.

Appendix

Additional Results

In this section, we present all additional qualitative results for the different methods presented in this thesis.

A.1 Learning-based Depth Map Fusion

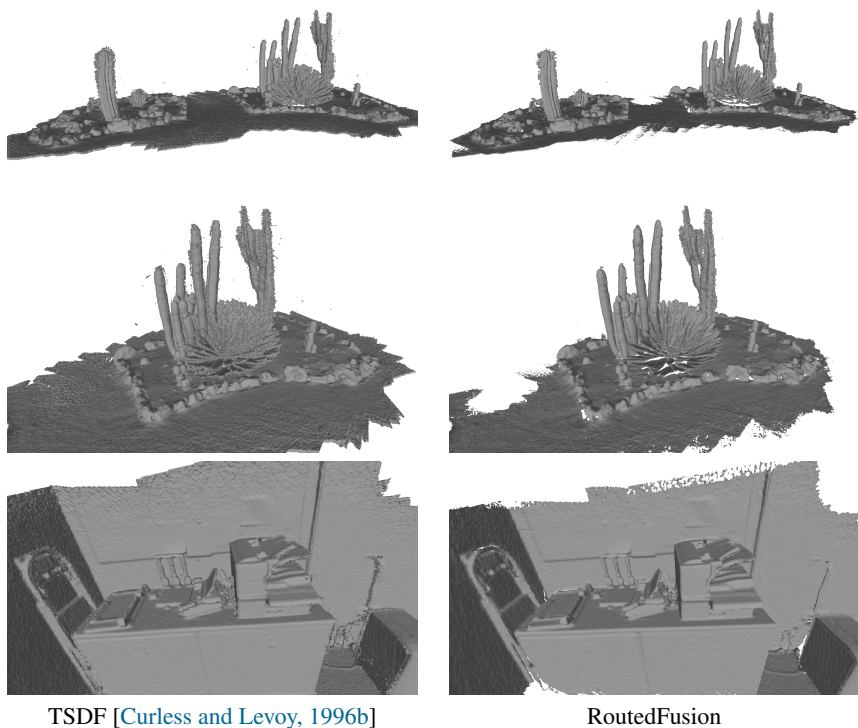


Figure. A.1: More qualitative results of standard TSDF and RoutedFusion on scene 3D data. They illustrate the significant performance difference in reconstructing fine geometries and clean edges.

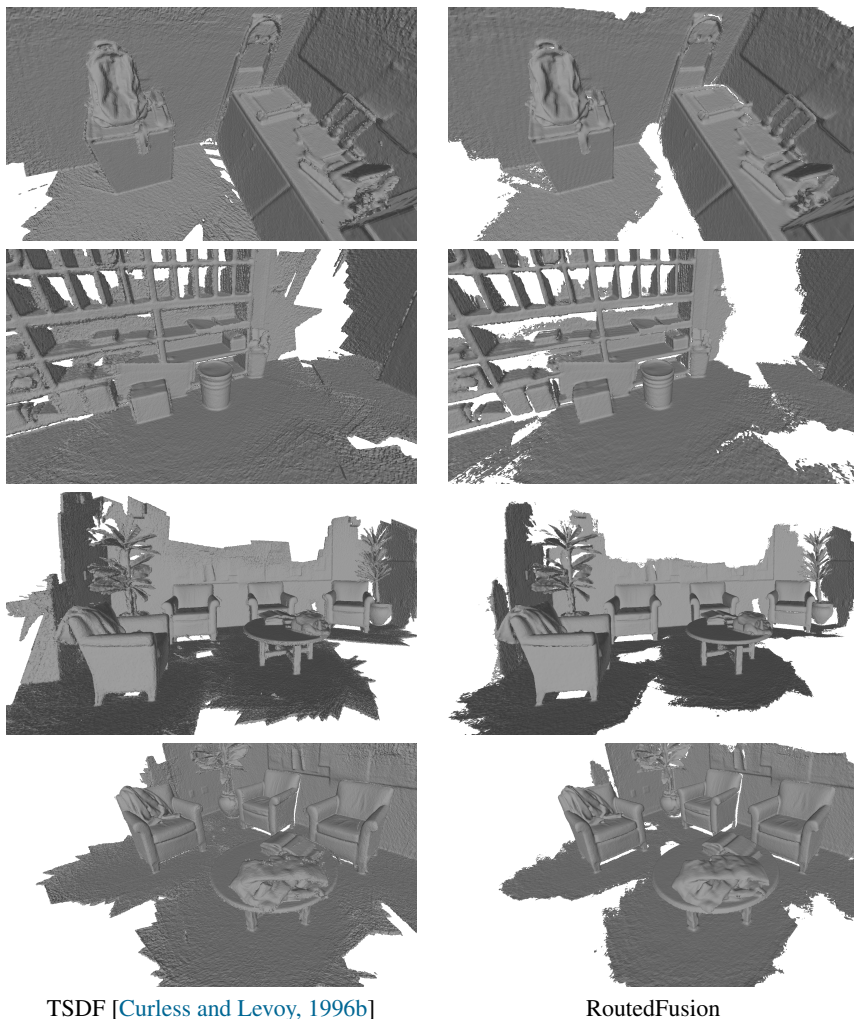


Figure. A.2: More qualitative results of standard TSDF and RoutedFusion on scene 3D data. They illustrate the significant performance difference in reconstructing fine geometries and clean edges.

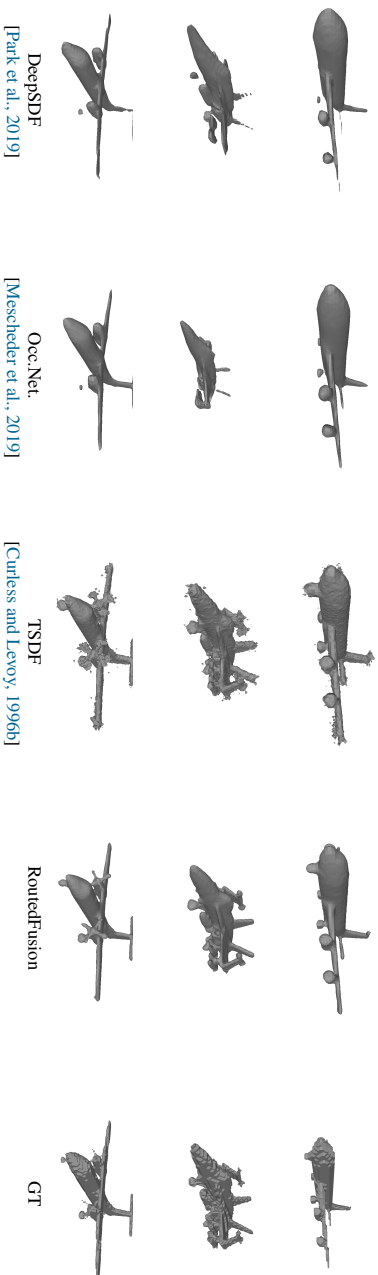


Figure. A.3: More qualitative results on ShapeNet test data. They illustrate the significant performance difference in reconstructing fine geometries and clean edges between Routefusion and standard TSDF as well as recent learning-based approaches.

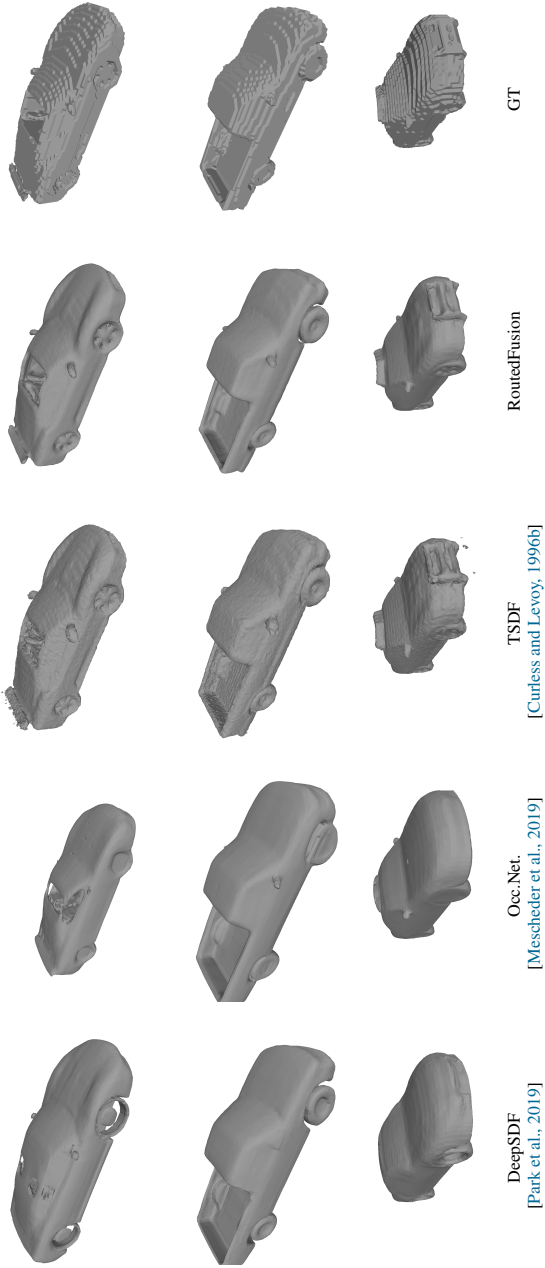


Figure. A-4: More qualitative results on ShapeNet test data They illustrate the significant performance difference in reconstructing fine geometries and clean edges between RoutedFusion and standard TSDF as well as recent learning-based approaches.

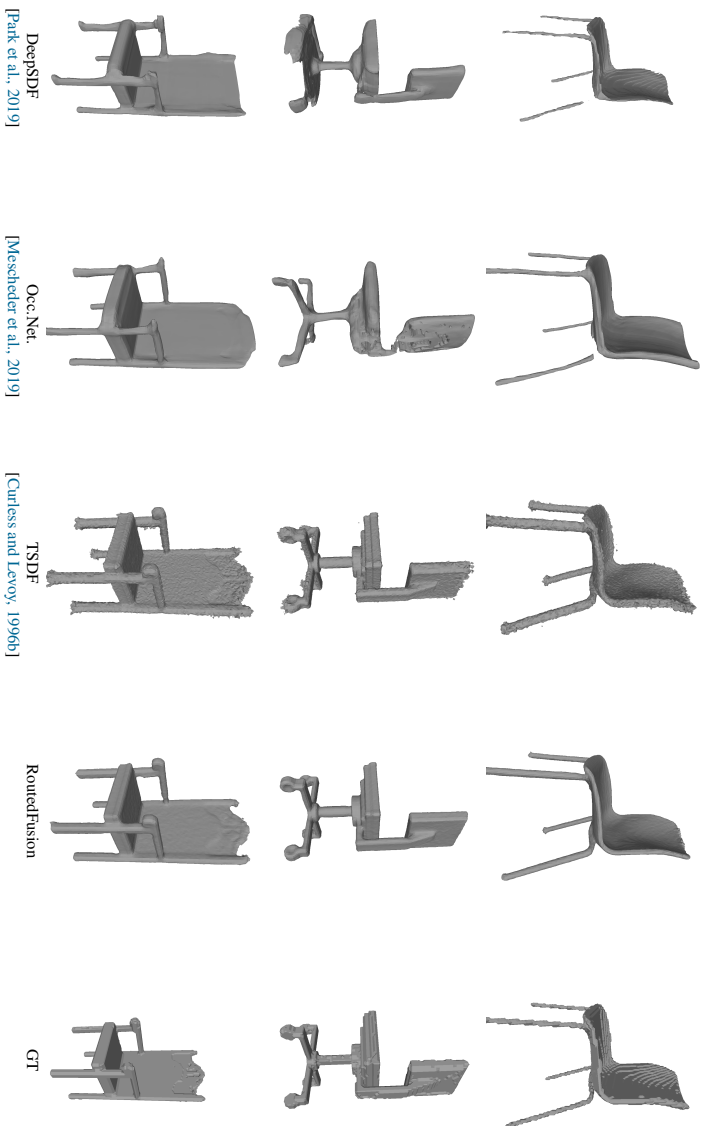


Figure. A.5: More qualitative results on ShapeNet test data. They illustrate the significant performance difference in reconstructing fine geometries and clean edges between RoutedFusion and standard TSDF as well as recent learning-based approaches.

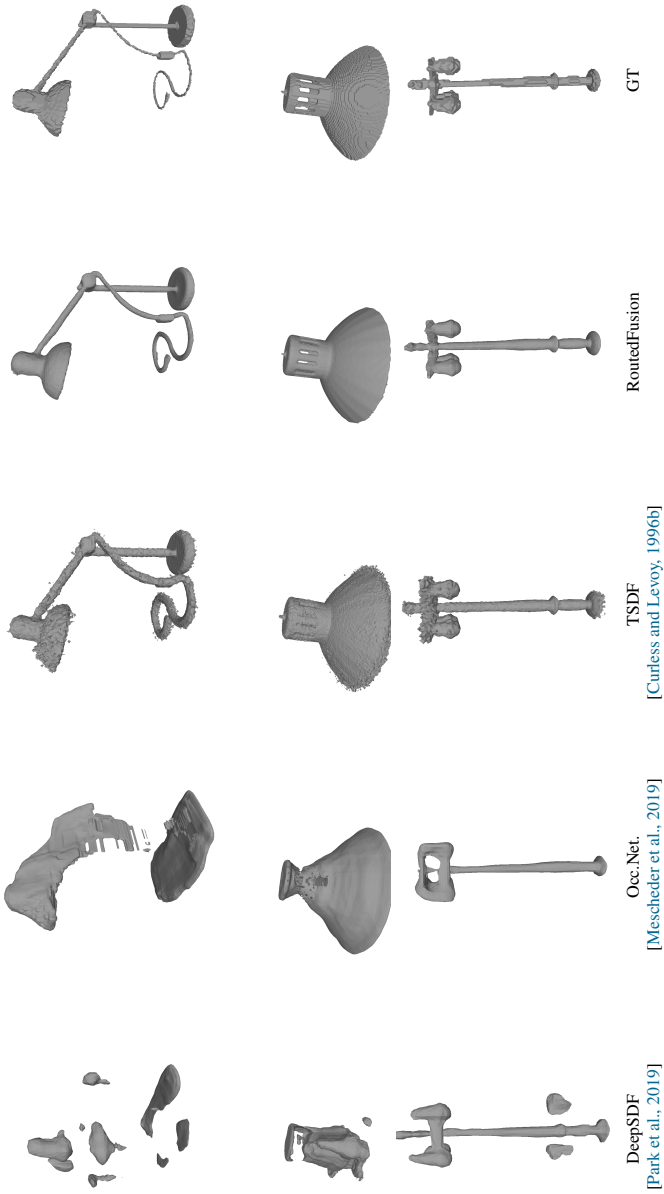


Figure. A.6: More qualitative results on ShapeNet test data. They illustrate the significant performance difference in reconstructing fine geometries and clean edges between RoutedFusion and standard TSDf as well as recent learning-based approaches

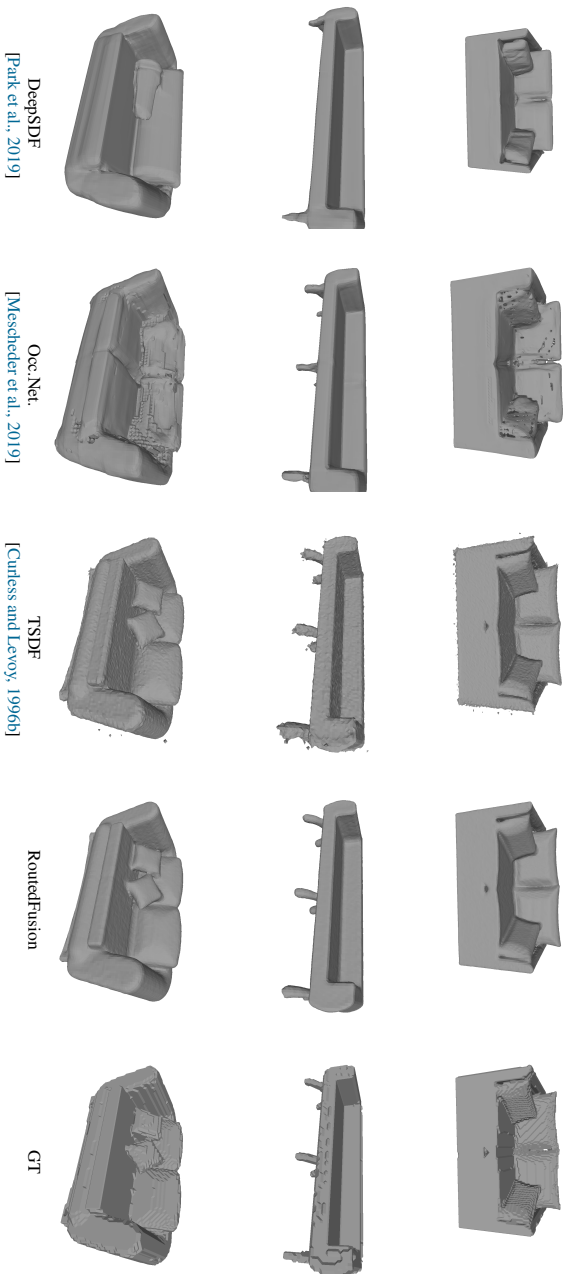


Figure. A.7: More qualitative results on ShapeNet test data They illustrate the significant performance difference in reconstructing fine geometries and clean edges between RoutedFusion and standard TSDF as well as recent learning-based approaches

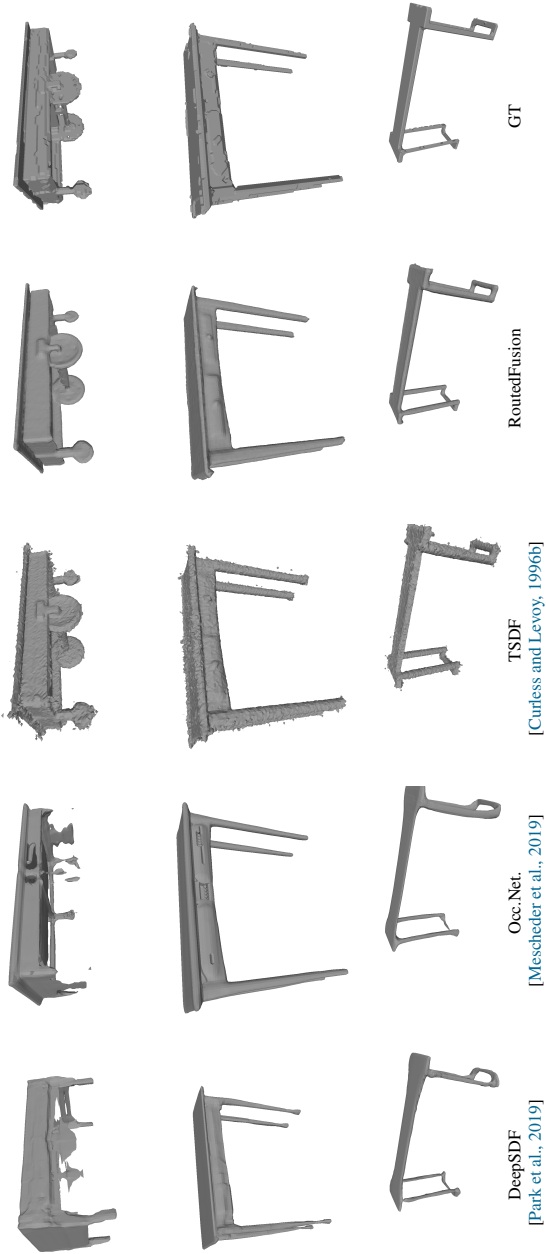


Figure. A.8: More qualitative results on ShapeNet test data. They illustrate the significant performance difference in reconstructing fine geometries and clean edges between RoutedFusion and standard TSDF as well as recent learning-based approaches

A.2 Moving the Fusion to a Learned Space

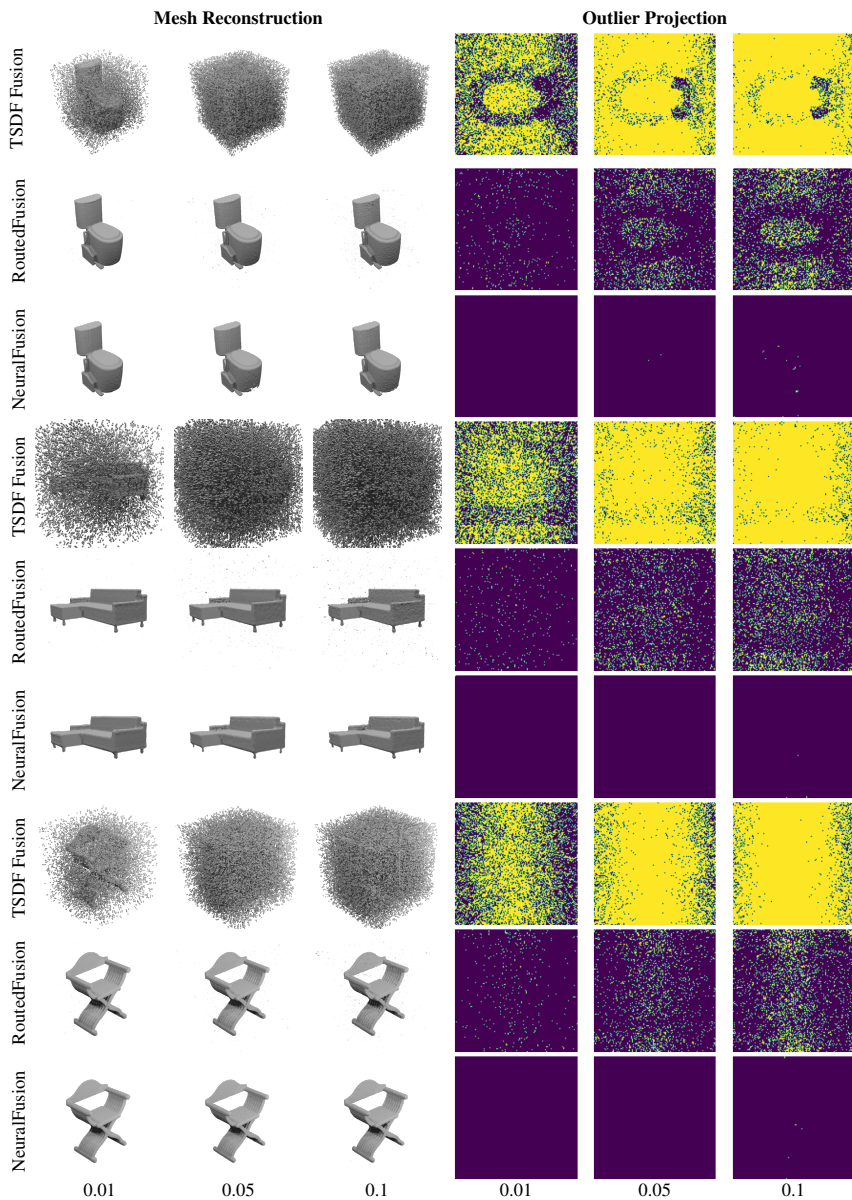


Figure. A.9: More qualitative results for different outlier fractions on ModelNet [Wu et al., 2015] examples. NeuralFusion consistently removes more outliers than existing depth map fusion methods compared to RotuedFusion and TSD F Fusion [Curless and Levoy, 1996a]. Even for large outlier fractions, our method successfully filters almost all of them.

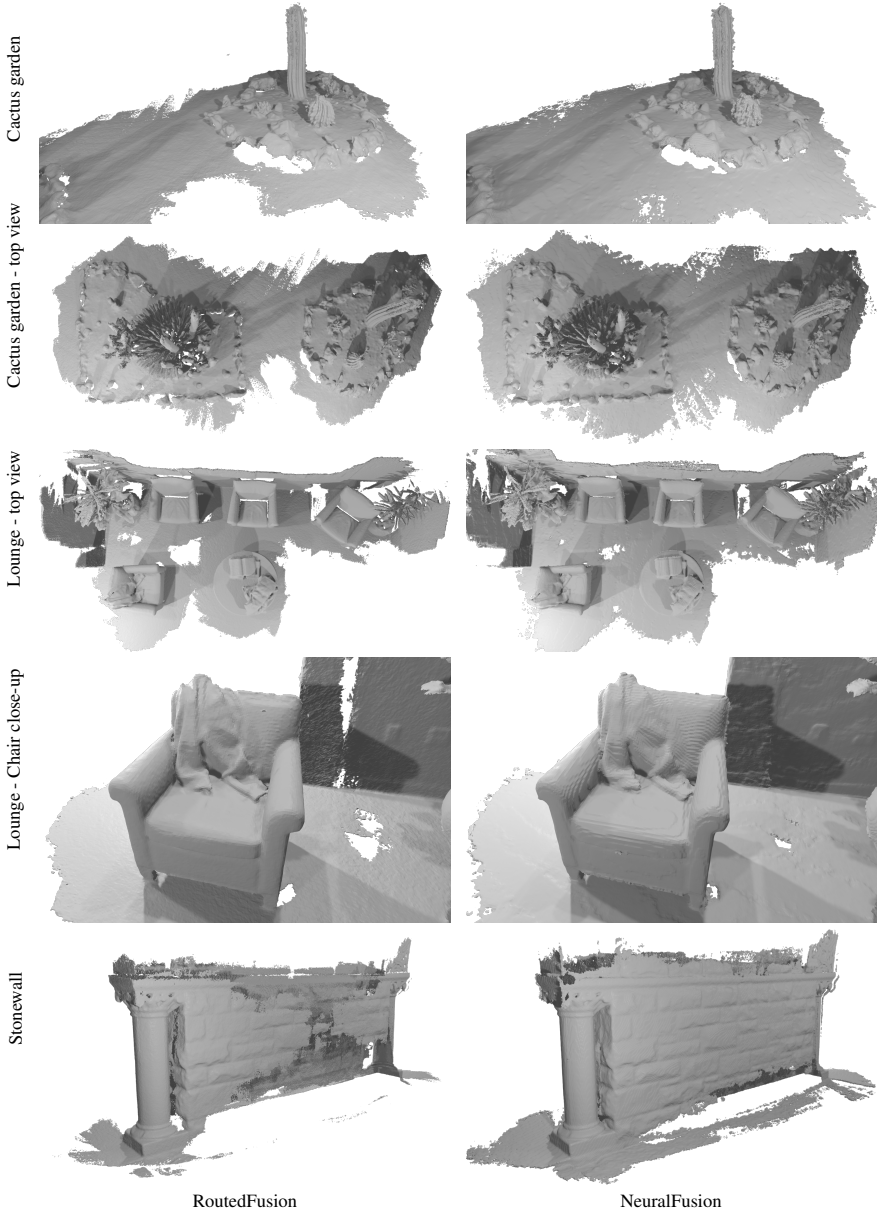


Figure. A.10: Additional results on Scene3D [Zhou and Koltun, 2013]. NeuralFusion reconstructs scenes with significantly higher completeness. This is due to the learned translation that can effectively discriminate between outliers and geometry. Furthermore, our method can filter large outlier blobs.

A.3 Learning-based Appearance Fusion

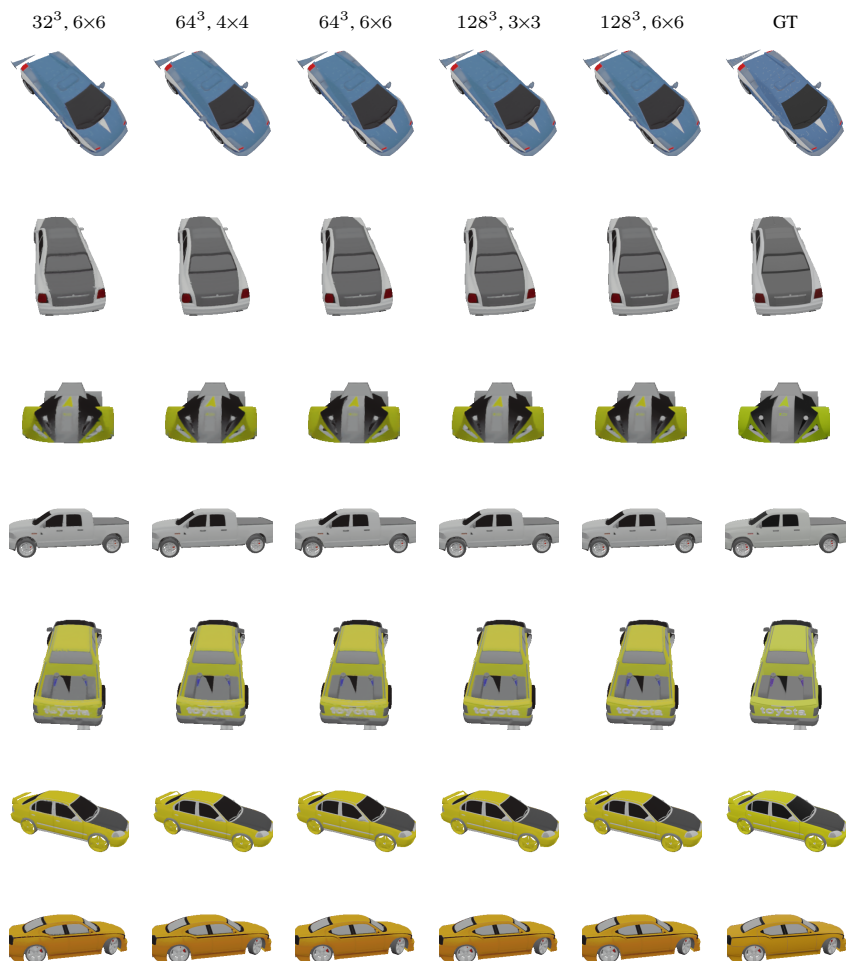


Figure. A.11: Qualitative results of our model on unseen ShapeNet [Chang et al., 2015] car scenes for different DeepSurfel parameters. The column names denote DeepSurfel grid and patch resolution respectively. We used DeepSurfels with 3 feature and 3 color channels (3+3 configuration).

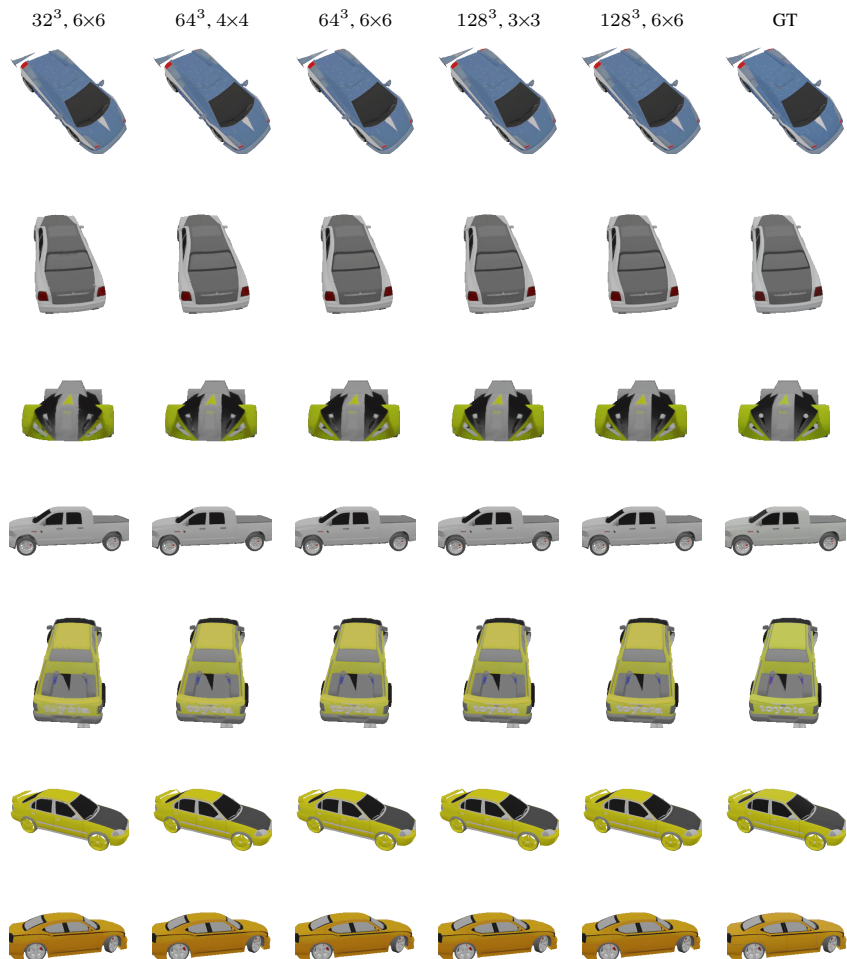


Figure. A.12: Qualitative results of our model on unseen ShapeNet [Chang et al., 2015] car scenes for different DeepSurfel parameters. DeepSurfels with 5 feature and 3 color channels (5+3 configuration) demonstrate better results compared to our method with less channels (3+3) displayed in Figure. A.11. The column name denotes DeepSurfel grid and patch resolution respectively.

Dataset for Object Removal

In this section, we present the dataset we proposed and used in part [IV](#).

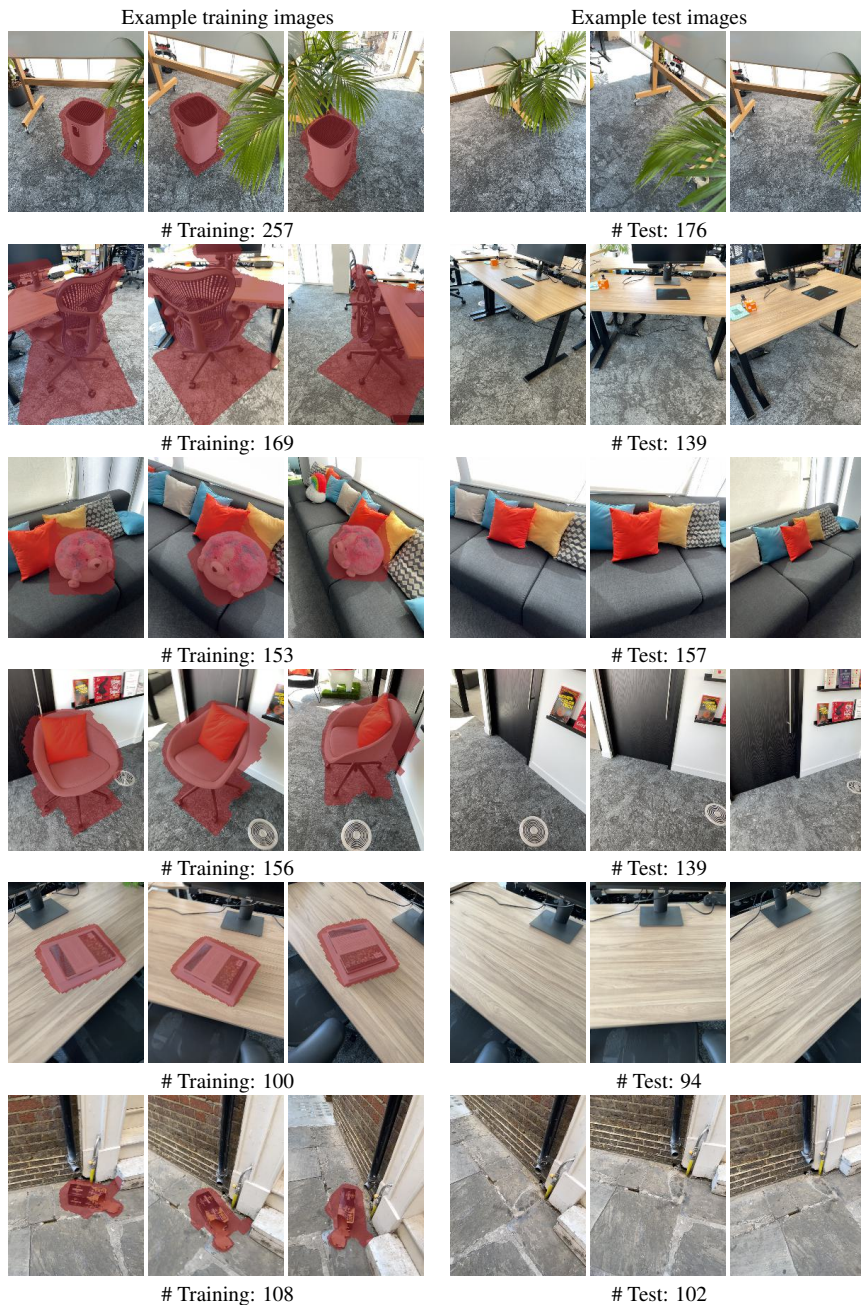
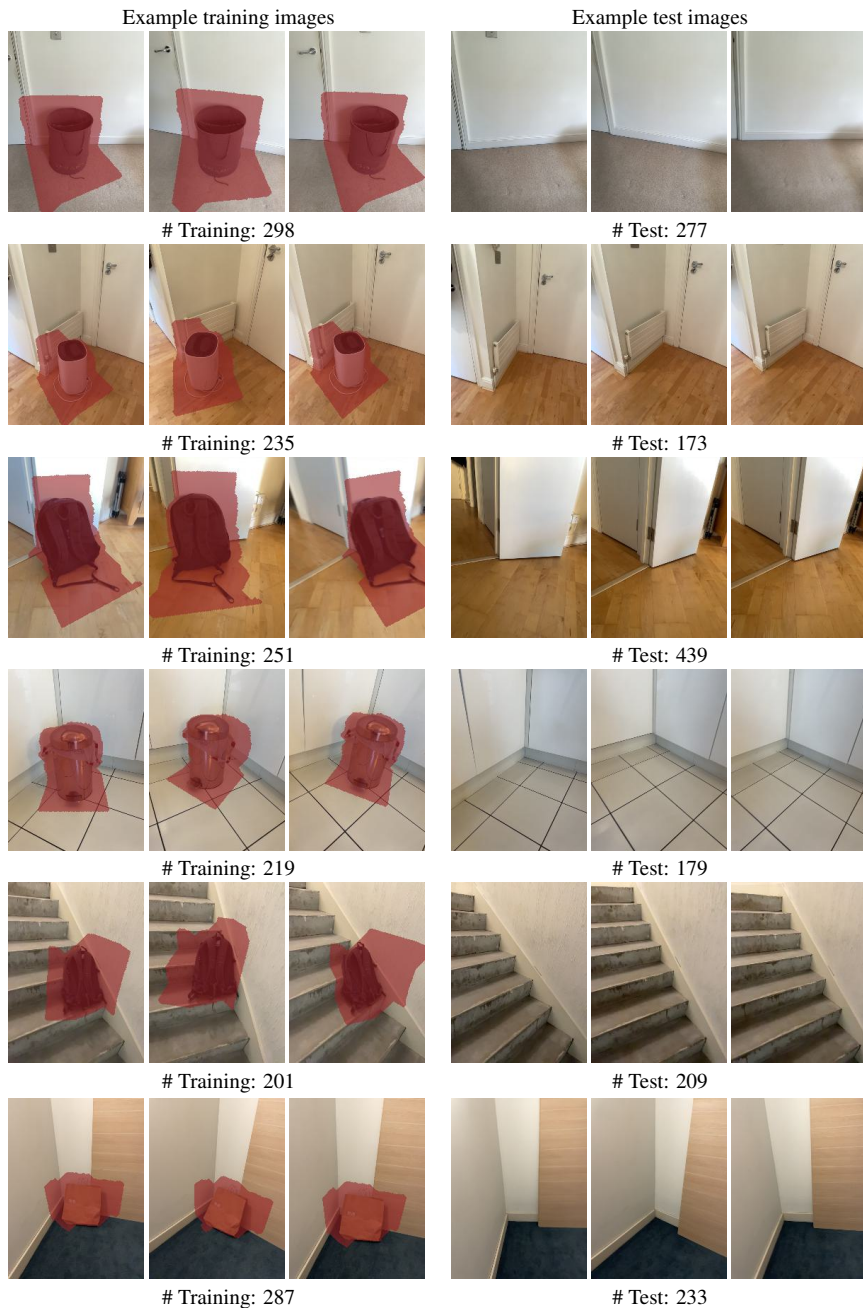


Figure. B.1: Our real objects dataset – Part 1 of 3

**Figure. B.2: Our real objects dataset – Part 2 of 3**

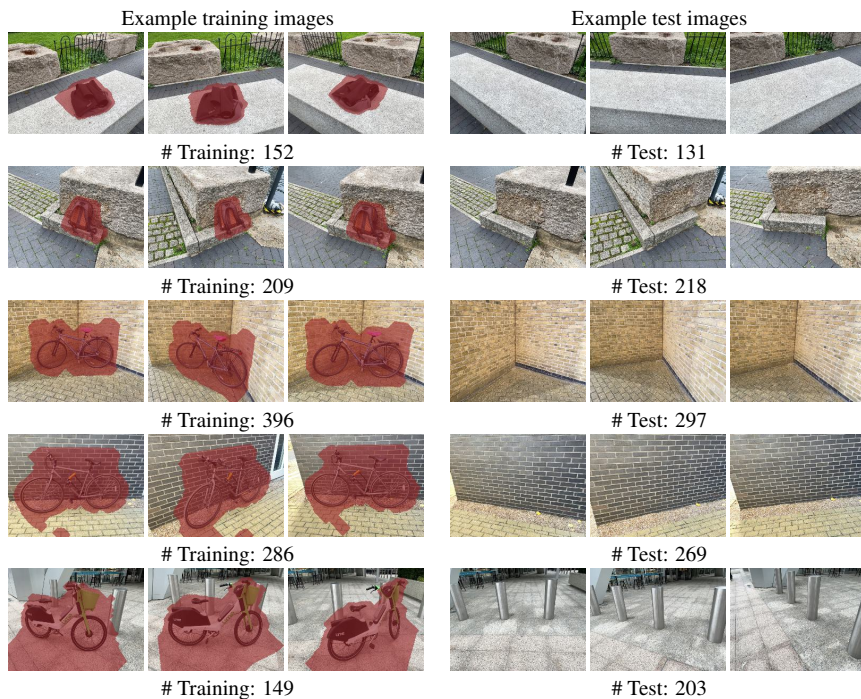


Figure. B.3: Our real objects dataset – Part 3 of 3

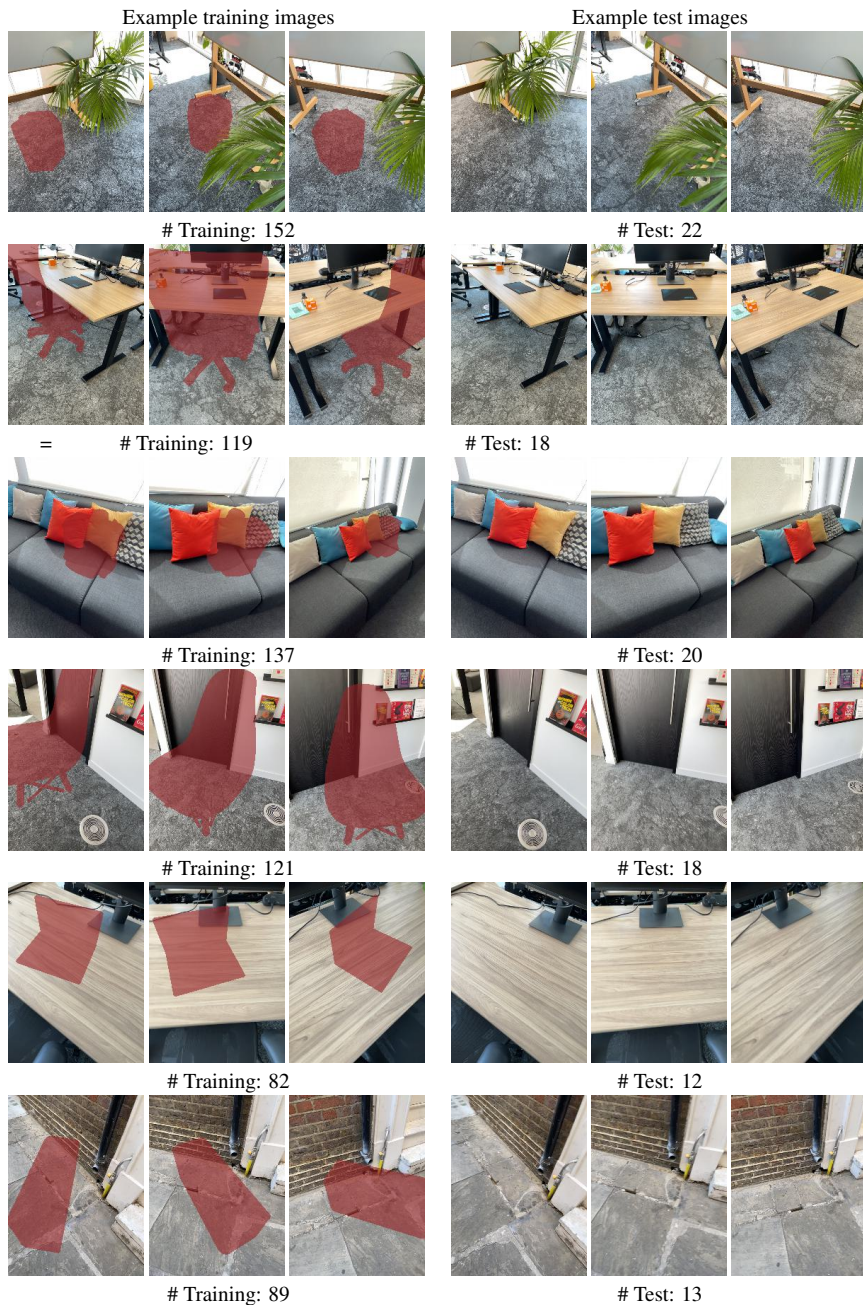


Figure. B.4: Our synthetic objects dataset – Part 1 of 3

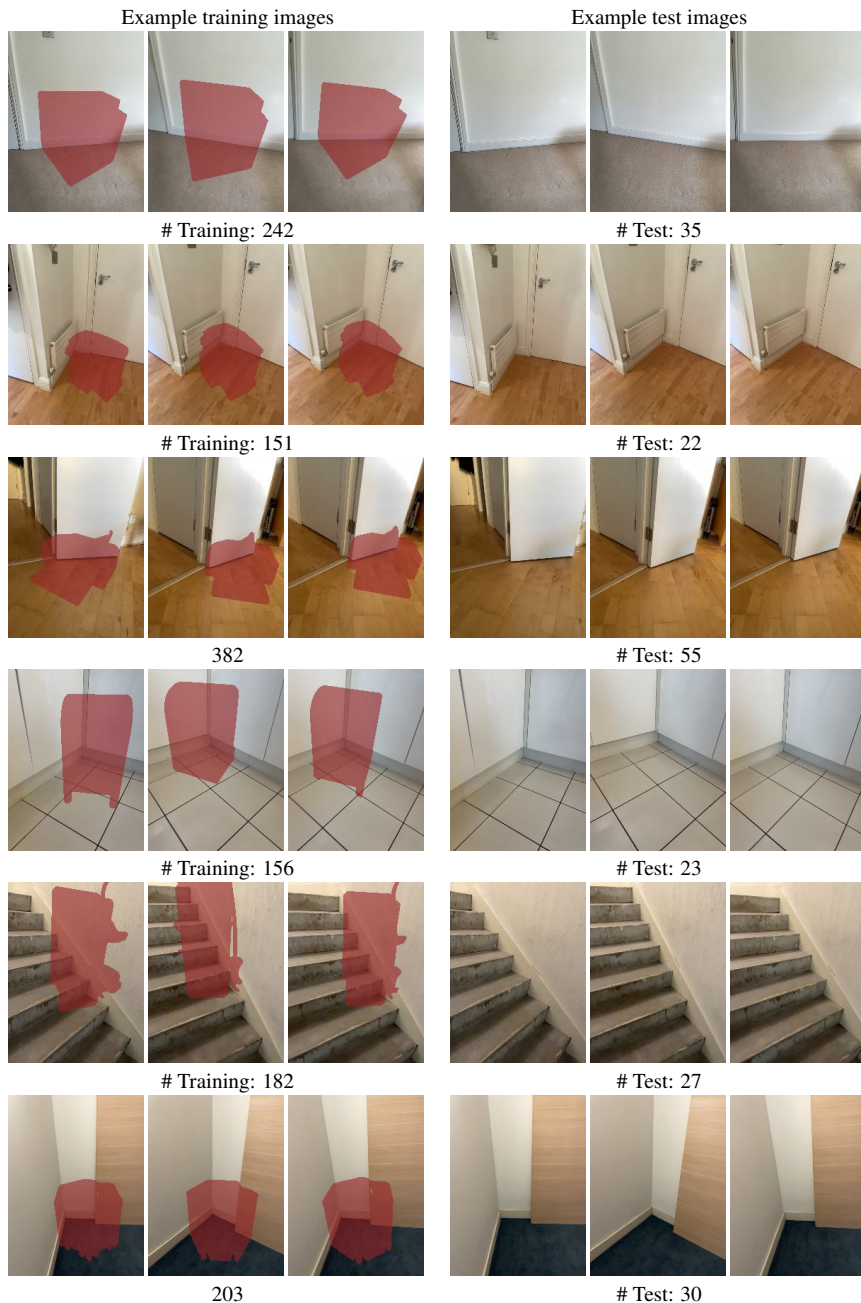


Figure. B.5: Our synthetic objects dataset – Part 2 of 3

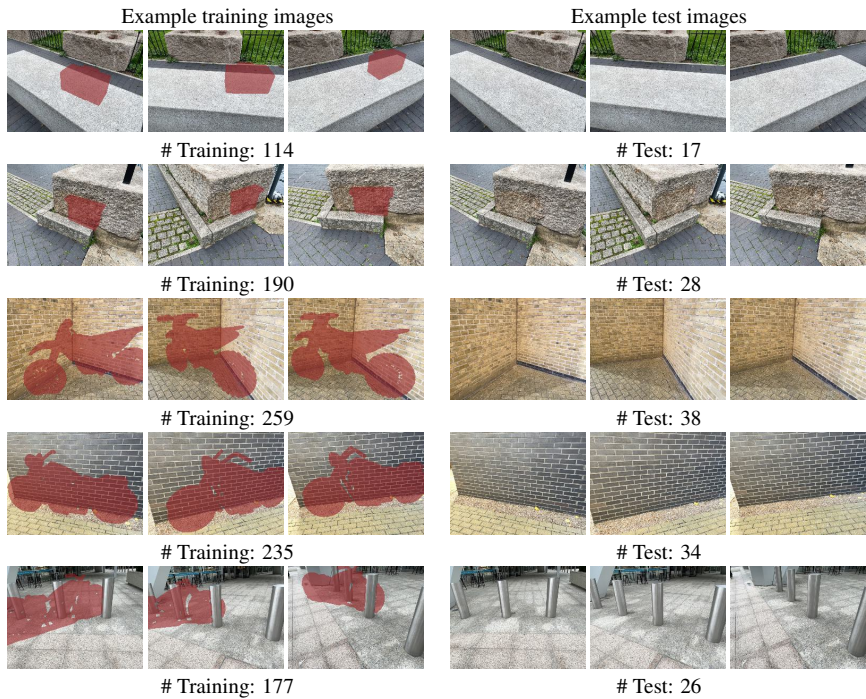


Figure. B.6: Our synthetic objects dataset – Part 3 of 3

Bibliography

- [Adobe Systems, Inc., 1990] Adobe Systems, Inc. (1990). Adobe photoshop. Computer software. [8](#), [171](#)
- [Agarwal et al., 2011] Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., and Szeliski, R. (2011). Building rome in a day. *Communications of the ACM*, 54(10):105–112. [20](#)
- [Ahmadyan et al., 2021] Ahmadyan, A., Zhang, L., Ablavatski, A., Wei, J., and Grundmann, M. (2021). Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *CVPR*. [191](#)
- [Aliev et al., 2020] Aliev, K.-A., Sevastopolsky, A., Kolos, M., Ulyanov, D., and Lempitsky, V. (2020). Neural point-based graphics. In *ECCV*. [27](#)
- [Allène et al., 2008] Allène, C., Pons, J.-P., and Keriven, R. (2008). Seamless image-based texture atlases using multi-band blending. In *ICCV*. [32](#)
- [Apple, 2017] Apple (2017). ARKit. Accessed: 14 October 2022. [21](#), [116](#), [184](#), [193](#)
- [Arbelaez et al., 2010] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2010). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916. [113](#)
- [Armando et al., 2019] Armando, M., Franco, J., and Boyer, E. (2019). Adaptive mesh texture for multi-view appearance modeling. In *International Conference on 3D Vision (3DV)*. [33](#)

- [Armeni et al., 2016] Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., and Savarese, S. (2016). 3D Semantic Parsing of Large-Scale Indoor Spaces. In *CVPR*. 116
- [Badki et al., 2020] Badki, A., Gallo, O., Kautz, J., and Sen, P. (2020). Meshlet priors for 3d mesh reconstruction. *CoRR*, abs/2001.01744. 26
- [Barnes et al., 2009a] Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009a). Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24. 174
- [Barnes et al., 2009b] Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009b). PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (ToG)*. 176
- [Barron et al., 2021] Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., and Srinivasan, P. P. (2021). Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*. 177, 181, 192, 193
- [Barron et al., 2022] Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2022). Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*. 177, 181, 187, 188, 193
- [Baruch et al., 2021a] Baruch, G., Chen, Z., Dehghan, A., Dimry, T., Feigin, Y., Fu, P., Gebauer, T., Joffe, B., Kurz, D., Schwartz, A., and Shulman, E. (2021a). ARKitscenes - a diverse real-world dataset for 3D indoor scene understanding using mobile RGB-D data. In *NeurIPS*. xvii, 195
- [Baruch et al., 2021b] Baruch, G., Chen, Z., Dehghan, A., Dimry, T., Feigin, Y., Fu, P., Gebauer, T., Joffe, B., Kurz, D., Schwartz, A., and Shulman, E. (2021b). ARKitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile RGB-d data. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. 116, 124, 125, 133, 141

- [Berger et al., 2014] Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Levine, J. A., Sharf, A., and Silva, C. T. (2014). State of the art in surface reconstruction from point clouds. In *35th Annual Conference of the European Association for Computer Graphics, Eurographics 2014-State of the Art Reports*, number CONF. The Eurographics Association. 23
- [Bernardini et al., 2001] Bernardini, F., Martin, I. M., and Rushmeier, H. E. (2001). High-quality texture reconstruction from multiple scans. *IEEE TVCG*. 32
- [Bešić and Valada, 2022] Bešić, B. and Valada, A. (2022). Dynamic object removal and spatio-temporal RGB-D inpainting via geometry-aware adversarial learning. *IEEE Transactions on Intelligent Vehicles*. 175
- [Bi et al., 2017] Bi, S., Kalantari, N. K., and Ramamoorthi, R. (2017). Patch-based optimization for image-based texture mapping. 32
- [Bi et al., 2020] Bi, S., Xu, Z., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., and Ramamoorthi, R. (2020). Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. In *ECCV*. 27
- [Bircher et al., 2015] Bircher, A., Alexis, K., Burri, M., Oettershagen, P., Omari, S., Mantel, T., and Siegwart, R. (2015). Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In *International Conference on Robotics and Automation (ICRA)*. 83
- [Bláha et al., 2016] Bláha, M., Vogel, C., Richard, A., Wegner, J. D., Pock, T., and Schindler, K. (2016). Large-scale semantic 3d reconstruction: an adaptive multi-resolution model for multi-class volumetric labeling. In *CVPR*. 35
- [Bleyer et al., 2011] Bleyer, M., Rhemann, C., and Rother, C. (2011). Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11, pages 1–11. 20
- [Bloesch et al., 2018] Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., and Davison, A. J. (2018). Codeslam - learning a compact, optimisable representation for dense visual SLAM. In *CVPR*, pages 2560–2568. 31

- [Borenstein and Ullman, 2002] Borenstein, E. and Ullman, S. (2002). Class-specific, top-down segmentation. In *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part II* 7, pages 109–122. Springer. 113
- [Bornard et al., 2002] Bornard, R., Lecan, E., Laborelli, L., and Chenot, J. (2002). Missing data correction in still images and image sequences. In Rowe, L. A., Merialdo, B., Mühlhäuser, M., Ross, K. W., and Dimitrova, N., editors, *Proceedings of the 10th ACM International Conference on Multimedia 2002, Juan les Pins, France, December 1-6, 2002*, pages 355–361. ACM. 173
- [Bozic et al., 2021] Bozic, A., Palafox, P., Thies, J., Dai, A., and Nießner, M. (2021). Transformerfusion: Monocular rgb scene reconstruction using transformers. *Advances in Neural Information Processing Systems*, 34:1403–1414. 31
- [Bréhéret, 2017] Bréhéret, A. (2017). Pixel Annotation Tool. <https://github.com/abreheret/PixelAnnotationTool>. 116
- [Breitenmoser and Siegwart, 2012] Breitenmoser, A. and Siegwart, R. (2012). Surface reconstruction and path planning for industrial inspection with a climbing robot. In *Proc. International Conference on Applied Robotics for the Power Industry (CARPI)*. 83
- [Caesar et al., 2018] Caesar, H., Uijlings, J., and Ferrari, V. (2018). Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1209–1218. 116, 124
- [Cao et al., 2018] Cao, Y., Liu, Z., Kuang, Z., Kobbelt, L., and Hu, S. (2018). Learning to reconstruct high-quality 3d shapes with cascaded fully convolutional networks. In *ECCV*, pages 626–643. 29
- [Carion et al., 2020] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer. 112

- [Chabra et al., 2020] Chabra, R., Lenssen, J. E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., and Newcombe, R. A. (2020). Deep local shapes: Learning local SDF priors for detailed 3d reconstruction. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J., editors, *ECCV*, volume 12374 of *Lecture Notes in Computer Science*, pages 608–625. Springer. 26
- [Chang et al., 2017] Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2017). Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*. 117, 124
- [Chang et al., 2015] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. (2015). Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*. xv, xvi, xvii, xviii, xix, 46, 47, 48, 49, 67, 69, 75, 95, 96, 101, 103, 193, 228, 229
- [Changpinyo et al., 2021] Changpinyo, S., Sharma, P., Ding, N., and Soricut, R. (2021). Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3558–3568. 126
- [Chen et al., 2021] Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., and Su, H. (2021). MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*. 177, 181
- [Chen et al., 2017a] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848. 114
- [Chen et al., 2017b] Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017b). Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*. 147, 162

- [Chen and Zhang, 2019] Chen, Z. and Zhang, H. (2019). Learning implicit fields for generative shape modeling. In *CVPR*. 26, 64
- [Cherabier et al., 2016] Cherabier, I., Häne, C., Oswald, M. R., and Pollefeys, M. (2016). Multi-label semantic 3d reconstruction using voxel blocks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 601–610. IEEE. 28, 114
- [Cherabier et al., 2018] Cherabier, I., Schönberger, J. L., Oswald, M. R., Pollefeys, M., and Geiger, A. (2018). Learning priors for semantic 3d reconstruction. In *European Conference on Computer Vision (ECCV)*. 29
- [Chibane et al., 2020] Chibane, J., Alldieck, T., and Pons-Moll, G. (2020). Implicit functions in feature space for 3D shape reconstruction and completion. In *CVPR*. 26, 67, 75, 76, 177
- [Choi et al., 2015] Choi, S., Zhou, Q., and Koltun, V. (2015). Robust reconstruction of indoor scenes. In *CVPR*, pages 5556–5565. 29
- [Chollet, 2017] Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. In *CVPR*. 147
- [Choy et al., 2019a] Choy, C., Gwak, J., and Savarese, S. (2019a). 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*. 115, 144, 154, 155, 157
- [Choy et al., 2019b] Choy, C., Gwak, J., and Savarese, S. (2019b). 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*. 119
- [Choy et al., 2016] Choy, C. B., Xu, D., Gwak, J., Chen, K., and Savarese, S. (2016). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 26

- [Chu et al., 2020] Chu, H., Ma, S., la Torre, F. D., Fidler, S., and Sheikh, Y. (2020). Expressive telepresence via modular codec avatars. In *ECCV, Lecture Notes in Computer Science*. 83
- [Cignoni et al., 2008] Cignoni, P., Corsini, M., and Ranzuglia, G. (2008). Mesh-Lab: an open-source 3D mesh processing system. *ERCIM News*. 191
- [Coleman and Andrews, 1979] Coleman, G. B. and Andrews, H. C. (1979). Image segmentation by clustering. *Proceedings of the IEEE*, 67(5):773–785. 112
- [Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619. 112
- [Community, 2020] Community, B. O. (2020). *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam. 95
- [Cook et al., 1984] Cook, R. L., Porter, T., and Carpenter, L. (1984). Distributed ray tracing. 91
- [Cordts et al., 2016] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. pages 3213–3223. 116, 124
- [Criminisi et al., 2003] Criminisi, A., Pérez, P., and Toyama, K. (2003). Object removal by exemplar-based inpainting. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), 16-22 June 2003, Madison, WI, USA*, pages 721–728. IEEE Computer Society. 173
- [Curless and Levoy, 1996a] Curless, B. and Levoy, M. (1996a). A volumetric method for building complex models from range images. 16, 27, 29, 30, 35, 36, 39, 46, 48, 49, 51, 53, 54, 55, 57, 59, 60, 68, 71, 75, 76, 81, 82, 95, 98, 103, 118, 120, 206, 225

- [Curless and Levoy, 1996b] Curless, B. and Levoy, M. (1996b). A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*, pages 303–312. [216](#), [217](#), [218](#), [219](#), [220](#), [221](#), [222](#), [223](#)
- [Czarnowski et al., 2020] Czarnowski, J., Laidlow, T., Clark, R., and Davison, A. (2020). Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 5:721–728. [31](#)
- [Dai et al., 2017a] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017a). ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *CVPR*. [xvi](#), [xix](#), [56](#), [116](#), [117](#), [124](#), [125](#), [127](#), [132](#), [133](#), [134](#), [135](#), [137](#), [138](#), [140](#), [146](#), [147](#), [155](#), [156](#), [157](#), [158](#), [159](#)
- [Dai and Nießner, 2018] Dai, A. and Nießner, M. (2018). 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 452–468. [28](#), [35](#), [114](#)
- [Dai et al., 2017b] Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., and Theobalt, C. (2017b). Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph.*, 36(3):24:1–24:18. [29](#), [35](#), [156](#)
- [Dai et al., 2018] Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., and Nießner, M. (2018). Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4578–4587. Computer Vision Foundation / IEEE Computer Society. [29](#), [35](#), [176](#)
- [Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee. [112](#)

- [Davis et al., 2012] Davis, A., Levoy, M., and Durand, F. (2012). Unstructured light fields. In *Computer Graphics Forum*. 177
- [Debevec et al., 1996] Debevec, P. E., Taylor, C. J., and Malik, J. (1996). Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. 32, 83, 86
- [Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*. 115, 147
- [Deng et al., 2022] Deng, K., Liu, A., Zhu, J.-Y., and Ramanan, D. (2022). Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*. 178, 181
- [DeVries et al., 2021] DeVries, T., Bautista, M. A., Srivastava, N., Taylor, G. W., and Susskind, J. M. (2021). Unconstrained scene generation with locally conditioned radiance fields. In *ICCV*. 178, 179
- [Dong et al., 2018] Dong, W., Wang, Q., Wang, X., and Zha, H. (2018). Psdf fusion: Probabilistic signed distance function for on-the-fly 3d data fusion and scene reconstruction. In *ECCV*. 16, 31, 46, 52, 54
- [Donné and Geiger, 2019] Donné, S. and Geiger, A. (2019). Defusr: Learning non-volumetric depth fusion using successive reprojections. In *CVPR*, pages 7634–7643. Computer Vision Foundation / IEEE. 31, 44
- [Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 114
- [Duan et al., 2012] Duan, Y., Pei, M., and Jia, Y. (2012). Probabilistic depth map fusion for real-time multi-view stereo. In *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*, pages 368–371. 30

- [Duzceker et al., 2021] Duzceker, A., Galliani, S., Vogel, C., Speciale, P., Dushmanu, M., and Pollefeys, M. (2021). Deepvideomvs: Multi-view stereo on video with recurrent spatio-temporal fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15324–15333. [20](#)
- [Efros and Leung, 1999] Efros, A. A. and Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999*, pages 1033–1038. IEEE Computer Society. [173](#)
- [Eftekhar et al., 2021] Eftekhar, A., Sax, A., Malik, J., and Zamir, A. (2021). Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796. [127](#), [131](#)
- [Eigen and Fergus, 2015] Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2650–2658. IEEE Computer Society. [21](#), [128](#)
- [Eisemann et al., 2008] Eisemann, M., De Decker, B., Magnor, M., Bekaert, P., de Aguiar, E., Ahmed, N., Theobalt, C., and Sellent, A. (2008). Floating Textures. *Comput. Graph. Forum.* [32](#), [83](#), [85](#), [86](#)
- [Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338. [115](#)
- [Fei-Fei et al., 2006] Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611. [115](#)

- [Felzenszwalb and Huttenlocher, 2004] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, 59:167–181. [113](#)
- [Flynn et al., 2019] Flynn, J., Broxton, M., Debevec, P., DuVall, M., Fyffe, G., Overbeck, R., Snavely, N., and Tucker, R. (2019). Deepview: View synthesis with learned gradient descent. In *CVPR*. [27](#)
- [Fu et al., 2018a] Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018a). Deep ordinal regression network for monocular depth estimation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2002–2011. Computer Vision Foundation / IEEE Computer Society. [21](#)
- [Fu et al., 2020] Fu, Y., Yan, Q., Liao, J., and Xiao, C. (2020). Joint texture and geometry optimization for RGB-D reconstruction. In *CVPR*. [32](#)
- [Fu et al., 2018b] Fu, Y., Yan, Q., Yang, L., Liao, J., and Xiao, C. (2018b). Texture mapping for 3d reconstruction with RGB-D sensor. In *CVPR*. [32](#), [83](#), [85](#), [86](#), [94](#), [95](#), [103](#)
- [Fuhrmann and Goesele, 2011] Fuhrmann, S. and Goesele, M. (2011). Fusion of depth maps with multiple scales. *ACM Trans. Graph.*, 30(6):148:1–148:8. [29](#)
- [Fujii et al., 2020] Fujii, R., Hachiuma, R., and Saito, H. (2020). RGB-D image inpainting using generative adversarial network with a late fusion approach. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*. [174](#)
- [Gal et al., 2010] Gal, R., Wexler, Y., Ofek, E., Hoppe, H., and Cohen-Or, D. (2010). Seamless montage for texturing models. *Comput. Graph. Forum*. [32](#)
- [Gallup et al., 2007] Gallup, D., Frahm, J.-M., Mordohai, P., Yang, Q., and Pollefeys, M. (2007). Real-time plane-sweeping stereo with multiple sweeping directions. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE. [20](#)

- [Gao et al., 2020] Gao, C., Saraf, A., Huang, J.-B., and Kopf, J. (2020). Flow-edge guided video completion. In *ECCV*. 175
- [Garrido et al., 2013] Garrido, S., Malfaz, M., and Blanco, D. (2013). Application of the fast marching method for outdoor motion planning in robotics. *Robotics and Autonomous Systems*. 83
- [Genova et al., 2019a] Genova, K., Cole, F., Sud, A., Sarna, A., and Funkhouser, T. A. (2019a). Deep structured implicit functions. *CoRR*, abs/1912.06126. 26
- [Genova et al., 2019b] Genova, K., Cole, F., Vlastic, D., Sarna, A., Freeman, W. T., and Funkhouser, T. A. (2019b). Learning shape templates with structured implicit functions. *CoRR*, abs/1904.06447. 26
- [Geosystems, 2016] Geosystems, L. (2016). Blk360 laser scanner. <https://leica-geosystems.com/de-ch/products/laser-scanners/scanners/blk360>. Accessed: 2023-10-02. 21
- [Ghiasi et al., 2021] Ghiasi, G., Gu, X., Cui, Y., and Lin, T.-Y. (2021). Scaling Open-Vocabulary Image Segmentation with Image-Level Labels. In *ECCV*. 123
- [Ghiasi et al., 2022] Ghiasi, G., Gu, X., Cui, Y., and Lin, T.-Y. (2022). Scaling open-vocabulary image segmentation with image-level labels. In *European Conference on Computer Vision*, pages 540–557. Springer. 114
- [Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448. 112
- [Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587. 112
- [Gkioxari et al., 2019] Gkioxari, G., Malik, J., and Johnson, J. (2019). Mesh r-cnn. In *ICCV*. 29

- [Goldlücke et al., 2014] Goldlücke, B., Aubry, M., Kolev, K., and Cremers, D. (2014). A super-resolution framework for high-accuracy multiview reconstruction. *IJCV*. 32
- [Graham et al., 2018] Graham, B., Engelcke, M., and Van Der Maaten, L. (2018). 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232. 115, 119
- [Granados et al., 2012] Granados, M., Tompkin, J., Kim, K. I., Grau, O., Kautz, J., and Theobalt, C. (2012). How not to be seen - object removal from videos of crowded scenes. *Comput. Graph. Forum*, 31(2pt1):219–228. 175
- [Griffin et al., 2007] Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset. 115
- [Groueix et al., 2018] Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. (2018). Atlasnet: A papier-mâché approach to learning 3d surface generation. In *CVPR*, pages 216–224. 29
- [Gupta et al., 2014] Gupta, S., Girshick, R. B., Arbeláez, P. A., and Malik, J. (2014). Learning rich features from RGB-D images for object detection and segmentation. In Fleet, D. J., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII*, volume 8695 of *Lecture Notes in Computer Science*, pages 345–360. Springer. 147
- [Han et al., 2020] Han, L., Zheng, T., Xu, L., and Fang, L. (2020). OccuSeg: Occupancy-Aware 3D Instance Segmentation. In *CVPR*. 120, 144
- [Häne et al., 2017] Häne, C., Heng, L., Lee, G. H., Fraundorfer, F., Furgale, P., Sattler, T., and Pollefeys, M. (2017). 3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing (IVC)*. 85

- [Häne et al., 2013] Häne, C., Zach, C., Cohen, A., Angst, R., and Pollefeys, M. (2013). Joint 3d scene reconstruction and class segmentation. In *CVPR*, pages 97–104. [28](#), [114](#)
- [Häne et al., 2017] Häne, C., Zach, C., Cohen, A., and Pollefeys, M. (2017). Dense semantic 3d reconstruction. *39(9):1730–1743*. [28](#)
- [Hays and Efros, 2007] Hays, J. and Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Trans. Graph.*, 26(3):4. [173](#)
- [Hazirbas et al., 2016] Hazirbas, C., Ma, L., Domokos, C., and Cremers, D. (2016). Fusetnet: Incorporating depth into semantic segmentation via fusion-based CNN architecture. In Lai, S., Lepetit, V., Nishino, K., and Sato, Y., editors, *Computer Vision - ACCV 2016 - 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part I*, volume 10111 of *Lecture Notes in Computer Science*, pages 213–228. Springer. [147](#)
- [He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *ICCV*. [191](#)
- [Hedman et al., 2018] Hedman, P., Philip, J., Price, T., Frahm, J.-M., Drettakis, G., and Brostow, G. (2018). Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6). [177](#)
- [Heeger and Bergen, 1995] Heeger, D. J. and Bergen, J. R. (1995). Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. [173](#)
- [Herling and Broll, 2010] Herling, J. and Broll, W. (2010). Advanced self-contained object removal for realizing real-time diminished reality in unconstrained environments. In *ISMAR*. [176](#)
- [Herling and Broll, 2014] Herling, J. and Broll, W. (2014). High-quality real-time video inpainting with PixMix. *IEEE TVCG*. [175](#)

- [Hoppe et al., 1992] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. A., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. In Thomas, J. J., editor, *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1992, Chicago, IL, USA, July 27-31, 1992*, pages 71–78. ACM. [23](#)
- [Hore and Ziou, 2010] Hore, A. and Ziou, D. (2010). Image quality metrics: PSNR vs. SSIM. In *ICPR*. [195](#)
- [Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861. [112](#), [163](#)
- [Hsu et al., 2021] Hsu, C.-Y., Sun, C., and Chen, H.-T. (2021). Moving in a 360 world: Synthesizing panoramic parallaxes from a single panorama. *arXiv:2106.10859*. [179](#)
- [Hu et al., 2021a] Hu, W., Zhao, H., Jiang, L., Jia, J., and Wong, T.-T. (2021a). Bidirectional Projection Network for Cross Dimension Scene Understanding. In *CVPR*. [119](#)
- [Hu et al., 2021b] Hu, Z., Bai, X., Shang, J., Zhang, R., Dong, J., Wang, X., Sun, G., Fu, H., and Tai, C.-L. (2021b). Vmnet: Voxel-mesh network for geodesic-aware 3d semantic segmentation. In *ICCV*. [119](#)
- [Hua et al., 2016] Hua, B.-S., Pham, Q.-H., Nguyen, D. T., Tran, M.-K., Yu, L.-F., and Yeung, S.-K. (2016). SceneNN: A Scene Meshes Dataset with Annotations. In *International Conference on 3D Vision (3DV)*. [116](#), [156](#)
- [Huang et al., 2021a] Huang, J., Huang, S.-S., Song, H., and Hu, S.-M. (2021a). Di-fusion: Online implicit 3d reconstruction with deep priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8932–8941. [31](#)

- [Huang et al., 2020] Huang, J., Thies, J., Dai, A., Kundu, A., Jiang, C., Guibas, L. J., Nießner, M., and Funkhouser, T. (2020). Adversarial texture optimization from RGB-D scans. In *CVPR*. 27
- [Huang et al., 2016] Huang, J.-B., Kang, S. B., Ahuja, N., and Kopf, J. (2016). Temporally coherent completion of dynamic video. *ACM Transactions on Graphics (ToG)*. 175
- [Huang et al., 2021b] Huang, S.-S., Ma, Z.-Y., Mu, T.-J., Fu, H., and Hu, S.-M. (2021b). Supervoxel Convolution for Online 3D Semantic Segmentation. *ACM TOG*. 121, 145, 154, 155, 156, 157, 159, 164
- [Huang et al., 2018] Huang, Z., Li, T., Chen, W., Zhao, Y., Xing, J., LeGendre, C., Luo, L., Ma, C., and Li, H. (2018). Deep volumetric video from very sparse multi-view performance capture. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *ECCV*, volume 11220 of *Lecture Notes in Computer Science*, pages 351–369. Springer. 26
- [Iizuka et al., 2017] Iizuka, S., Simo-Serra, E., and Ishikawa, H. (2017). Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14. 174
- [Intel, 2014] Intel (2014). Intel realsense. <https://www.intelrealsense.com/stereo-depth/>. Accessed: 2023-10-17. 21
- [Izadi et al., 2011] Izadi, S., Newcombe, R. A., Kim, D., Hilliges, O., Molyneaux, D., Hodges, S., Kohli, P., Shotton, J., Davison, A. J., and Fitzgibbon, A. W. (2011). Kinectfusion: real-time dynamic 3d surface reconstruction and interaction. In *International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2011, Vancouver, BC, Canada, August 7-11, 2011, Talks Proceedings*, page 23. 29, 35
- [Jatavallabhula et al., 2023] Jatavallabhula, K., Kuwajerwala, A., Gu, Q., Omama, M., Chen, T., Li, S., Iyer, G., Saryazdi, S., Keetha, N., Tewari, A., Tenenbaum, J., de Melo, C., Krishna, M., Paull, L., Shkurti, F., and Torralba, A. (2023). ConceptFusion: Open-Set Multimodal 3D Mapping. *arXiv*. 210

- [Jeon et al., 2018] Jeon, J., Jung, J., Kim, J., and Lee, S. (2018). Semantic Reconstruction: Reconstruction of Semantically Segmented 3D Meshes via Volumetric Semantic Fusion. In *Computer Graphics Forum*. 120, 159
- [Ji et al., 2017] Ji, M., Gall, J., Zheng, H., Liu, Y., and Fang, L. (2017). Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *ICCV*, pages 2326–2334. 28
- [Jiang et al., 2020] Jiang, C. M., Sud, A., Makadia, A., Huang, J., Nießner, M., and Funkhouser, T. A. (2020). Local implicit grid representations for 3d scenes. In *CVPR*, pages 6000–6009. IEEE. 26
- [Johanna Wald, 2019] Johanna Wald, Armen Avetisyan, N. N. F. T. M. N. (2019). Rio: 3d object instance re-localization in changing indoor environments. 116
- [Kähler et al., 2015] Kähler, O., Prisacariu, V. A., Ren, C. Y., Sun, X., Torr, P. H. S., and Murray, D. W. (2015). Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Trans. Vis. Comput. Graph.*, 21(11):1241–1250. 29
- [Kähler et al., 2016] Kähler, O., Prisacariu, V. A., Valentin, J. P. C., and Murray, D. W. (2016). Hierarchical voxel block hashing for efficient integration of depth images. *IEEE Robotics and Automation Letters*, 1(1):192–197. 29
- [Kar et al., 2017] Kar, A., Häne, C., and Malik, J. (2017). Learning a multi-view stereo machine. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 365–376. 26
- [Kawai et al., 2015] Kawai, N., Sato, T., and Yokoya, N. (2015). Diminished reality based on image inpainting considering background geometry. *IEEE transactions on visualization and computer graphics*, 22(3):1236–1247. 176

- [Kazhdan et al., 2006] Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, page 0. [23](#)
- [Kazhdan and Hoppe, 2013] Kazhdan, M. M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13. [71](#), [72](#), [82](#)
- [Keller et al., 2013] Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., and Kolb, A. (2013). Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision, 3DV 2013, Seattle, Washington, USA, June 29 - July 1, 2013*, pages 1–8. [30](#)
- [Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5574–5584. [44](#), [188](#)
- [Kerbl et al., 2023] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14. [211](#)
- [Kim et al., 2019] Kim, D., Woo, S., Lee, J., and Kweon, I. S. (2019). Deep video inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 5792–5801. Computer Vision Foundation / IEEE. [175](#)
- [Kim et al., 2021] Kim, J., Hyeon, J., and Doh, N. (2021). Generative multiview inpainting for object removal in large indoor spaces. *International Journal of Advanced Robotic Systems*, 18(2). [176](#)
- [Kim et al., 2022] Kim, M., Seo, S., and Han, B. (2022). Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *CVPR*. [177](#), [181](#)
- [Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*. [66](#), [93](#)

- [Kirillov et al., 2023] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. (2023). Segment anything. *arXiv preprint arXiv:2304.02643*. [123](#)
- [Knapitsch et al., 2017] Knapitsch, A., Park, J., Zhou, Q.-Y., and Koltun, V. (2017). Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4). [xvi](#), [60](#), [71](#), [72](#), [78](#), [82](#)
- [Kobayashi et al., 2022] Kobayashi, S., Matsumoto, E., and Sitzmann, V. (2022). Decomposing NeRF for editing via feature field distillation. In *NeurIPS*. [178](#), [183](#)
- [Koh et al., 2021] Koh, J. Y., Lee, H., Yang, Y., Baldrige, J., and Anderson, P. (2021). Pathdreamer: A world model for indoor navigation. In *ICCV*. [179](#)
- [Kolev et al., 2009] Kolev, K., Klodt, M., Brox, T., and Cremers, D. (2009). Continuous global optimization in multiview 3d reconstruction. *IJCV*, 84(1):80–96. [28](#), [35](#)
- [Kontogianni et al., 2023] Kontogianni, T., Celikkan, E., Tang, S., and Schindler, K. (2023). Interactive object segmentation in 3d point clouds. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2891–2897. IEEE. [117](#), [132](#), [133](#)
- [Kundu et al., 2020] Kundu, A., Yin, X., Fathi, A., Ross, D., Brewington, B., Funkhouser, T., and Pantofaru, C. (2020). Virtual Multi-view Fusion for 3D Semantic Segmentation. In *ECCV*. [119](#), [144](#), [147](#), [155](#), [157](#)
- [Kutulakos and Seitz, 1999] Kutulakos, K. N. and Seitz, S. M. (1999). A theory of shape by space carving. In *ICCV*. [33](#)
- [labelme github contributors,] labelme github contributors. labelme: Image polygonal annotation with python. [116](#)
- [Ladicky et al., 2017] Ladicky, L., Saurer, O., Jeong, S., Maninchedda, F., and Pollefeys, M. (2017). From point clouds to mesh using regression. In *ICCV*, pages 3913–3922. [25](#), [26](#)

- [Lambert et al., 2020] Lambert, J., Liu, Z., Sener, O., Hays, J., and Koltun, V. (2020). MSeg: A Composite Dataset for Multi-Domain Semantic Segmentation. pages 2879–2888. [128](#)
- [Lee et al., 2020] Lee, J. H., Ha, H., Dong, Y., Tong, X., and Kim, M. H. (2020). Texturefusion: High-quality texture acquisition for real-time rgb-d scanning. In *CVPR*. [32](#), [33](#)
- [Lefloch et al., 2015] Lefloch, D., Weyrich, T., and Kolb, A. (2015). Anisotropic point-based fusion. In *18th International Conference on Information Fusion, FUSION 2015, Washington, DC, USA, July 6-9, 2015*, pages 2121–2128. [30](#)
- [Lempitsky and Ivanov, 2007] Lempitsky, V. and Ivanov, D. (2007). Seamless mosaicing of image-based texture maps. In *CVPR*. [32](#)
- [Lensch et al., 2001] Lensch, H. P. A., Heidrich, W., and Seidel, H.-P. (2001). A silhouette-based algorithm for texture registration and stitching. *Graphical Models*. [32](#)
- [Lepetit et al., 2001] Lepetit, V., Berger, M.-O., and Lorrain, L.-I. (2001). An intuitive tool for outlining objects in video sequences: Applications to augmented and diminished reality. In *ISMAR*. [176](#)
- [Leroy et al., 2018] Leroy, V., Franco, J., and Boyer, E. (2018). Shape reconstruction using volume sweeping and learned photoconsistency. In *ECCV*, pages 796–811. [28](#)
- [Li et al., 2022a] Li, B., Weinberger, K. Q., Belongie, S. J., Koltun, V., and Ranftl, R. (2022a). Language-driven semantic segmentation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. [114](#)
- [Li et al., 2022b] Li, Y., Mao, H., Girshick, R., and He, K. (2022b). Exploring plain vision transformer backbones for object detection. In *European Conference on Computer Vision*, pages 280–296. Springer. [112](#)

- [Li et al., 2019] Li, Y., Tsiminaki, V., Timofte, R., Pollefeys, M., and Gool, L. V. (2019). 3d appearance super-resolution with deep learning. In *CVPR*. 33
- [Li et al., 2022c] Li, Z., Fang, T., Li, Z., Cui, Z., Sato, Y., Pollefeys, M., and Oswald, M. R. (2022c). CompNVS: Novel view synthesis with scene completion. In *ECCV*. 178, 179, 194, 196
- [Li et al., 2022d] Li, Z., Lu, C.-Z., Qin, J., Guo, C.-L., and Cheng, M.-M. (2022d). Towards an end-to-end framework for flow-guided video inpainting. In *CVPR*. 175, 191, 193, 194, 196
- [Li and Snavely, 2018] Li, Z. and Snavely, N. (2018). Megadepth: Learning single-view depth prediction from internet photos. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2041–2050. Computer Vision Foundation / IEEE Computer Society. 21
- [Liang et al., 2023] Liang, F., Wu, B., Dai, X., Li, K., Zhao, Y., Zhang, H., Zhang, P., Vajda, P., and Marculescu, D. (2023). Open-vocabulary semantic segmentation with mask-adapted clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7061–7070. 114, 123, 124, 126
- [Liao et al., 2020] Liao, M., Lu, F., Zhou, D., Zhang, S., Li, W., and Yang, R. (2020). DVI: Depth guided video inpainting for autonomous driving. In *ECCV*. 175
- [Lin et al., 2017] Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007. IEEE Computer Society. 151
- [Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common

- objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer. [116](#)
- [Liu et al., 2018] Liu, G., Reda, F. A., Shih, K. J., Wang, T., Tao, A., and Catanzaro, B. (2018). Image inpainting for irregular holes using partial convolutions. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, volume 11215 of *Lecture Notes in Computer Science*, pages 89–105. Springer. [174](#)
- [Liu et al., 2022a] Liu, H.-K., Shen, I.-C., and Chen, B.-Y. (2022a). NeRF-In: Free-form NeRF inpainting with RGB-D priors. *arXiv*. [179](#)
- [Liu et al., 2022b] Liu, L., Zheng, T., Lin, Y., Ni, K., and Fang, L. (2022b). INS-Conv: Incremental Sparse Convolution for Online 3D Segmentation. In *CVPR*. [121](#), [145](#), [153](#), [155](#), [156](#), [159](#), [164](#)
- [Liu et al., 2019a] Liu, S., Saito, S., Chen, W., and Li, H. (2019a). Learning to infer implicit surfaces without 3D supervision. *Neural Information Processing Systems (NeurIPS)*. [26](#), [177](#)
- [Liu et al., 2021a] Liu, S., Zhang, X., Zhang, Z., Zhang, R., Zhu, J.-Y., and Russell, B. (2021a). Editing conditional radiance fields. In *ICCV*. [181](#)
- [Liu et al., 2020a] Liu, S., Zhang, Y., Peng, S., Shi, B., Pollefeys, M., and Cui, Z. (2020a). DIST: rendering deep implicit signed distance function with differentiable sphere tracing. In *CVPR*, pages 2016–2025. IEEE. [26](#)
- [Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer. [112](#)
- [Liu et al., 2019b] Liu, W., Piao, Z., Min, J., Luo, W., Ma, L., and Gao, S. (2019b). Liquid warping GAN: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *ICCV*. [177](#)

- [Liu et al., 2020b] Liu, Y.-L., Lai, W.-S., Yang, M.-H., Chuang, Y.-Y., and Huang, J.-B. (2020b). Learning to see through obstructions. In *CVPR*. 176
- [Liu et al., 2021b] Liu, Y.-L., Lai, W.-S., Yang, M.-H., Chuang, Y.-Y., and Huang, J.-B. (2021b). Learning to see through obstructions with layered decomposition. *IEEE TPAMI*. 176
- [Liu et al., 2019] Liu, Z., Cao, Y., Kuang, Z., Kobbelt, L., and Hu, S. (2019). High-quality textured 3d shape reconstruction with cascaded fully convolutional networks. *IEEE TVCG*. 27
- [Liu et al., 2023] Liu, Z., Milano, F., Frey, J., Siegwart, R., Blum, H., and Cadena, C. (2023). Unsupervised continual semantic adaptation through neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 124, 131
- [Lombardi et al., 2018] Lombardi, S., Saragih, J. M., Simon, T., and Sheikh, Y. (2018). Deep appearance models for face rendering. 83
- [Lombardi et al., 2019] Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., and Sheikh, Y. (2019). Neural volumes: Learning dynamic renderable volumes from images. 27, 83
- [Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440. 113
- [Long et al., 2022] Long, X., Lin, C., Wang, P., Komura, T., and Wang, W. (2022). SparseNeuS: Fast generalizable neural surface reconstruction from sparse views. In *ECCV*. 177, 181
- [Lorensen and Cline, 1987] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. 24
- [Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110. 111

- [Lugmayr et al., 2022] Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. (2022). Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471. 174
- [Maier et al., 2017a] Maier, R., Kim, K., Cremers, D., Kautz, J., and Nießner, M. (2017a). Intrinsic3d: High-quality 3D reconstruction by joint appearance and geometry optimization with spatially-varying lighting. In *International Conference on Computer Vision (ICCV)*, Venice, Italy. 33
- [Maier et al., 2017b] Maier, R., Schaller, R., and Cremers, D. (2017b). Efficient online surface correction for real-time large-scale 3D reconstruction. In *British Machine Vision Conference (BMVC)*, London, United Kingdom. xvi, 29, 33, 97, 104
- [Mao et al., 2022] Mao, Y., Zhang, Y., Jiang, H., Chang, A. X., and Savva, M. (2022). Multiscan: Scalable rgbd scanning for 3d environments with articulated objects. In *Advances in Neural Information Processing Systems*. 117
- [Marniok and Goldluecke, 2018] Marniok, N. and Goldluecke, B. (2018). Real-time variational range image fusion and visualization for large-scale scenes using GPU hash tables. In *2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018*, pages 912–920. 29
- [Marniok et al., 2017] Marniok, N., Johannsen, O., and Goldluecke, B. (2017). An efficient octree design for local variational range image fusion. pages 401–412. 29
- [McCormac et al., 2018] McCormac, J., Clark, R., Bloesch, M., Davison, A. J., and Leutenegger, S. (2018). Fusion++: Volumetric Object-Level SLAM. In *International Conference on 3D Vision (3DV)*. 120
- [McCormac et al., 2017] McCormac, J., Handa, A., Davison, A. J., and Leutenegger, S. (2017). Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and*

- Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 4628–4635. IEEE. [120](#), [145](#), [159](#)
- [Mescheder et al., 2019] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*. [26](#), [47](#), [48](#), [49](#), [67](#), [75](#), [76](#), [177](#), [218](#), [219](#), [220](#), [221](#), [222](#), [223](#)
- [Michalkiewicz et al., 2019] Michalkiewicz, M., Pontes, J. K., Jack, D., Baktashmotlagh, M., and Eriksson, A. (2019). Implicit surface representations as layers in neural networks. In *ICCV*. [26](#)
- [Microsoft, 2020] Microsoft (2020). Azure kinect. <https://azure.microsoft.com/en-us/products/kinect-dk>. Accessed: 2023-10-17. [22](#)
- [Mihajlovic et al., 2021] Mihajlovic, M., Weder, S., Pollefeys, M., and Oswald, M. R. (2021). Deepsurfels: Learning online appearance fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14524–14535. [11](#)
- [Mildenhall et al., 2019] Mildenhall, B., Srinivasan, P. P., Ortiz-Cayon, R., Kalantari, N. K., Ramamoorthi, R., Ng, R., and Kar, A. (2019). Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. [27](#)
- [Mildenhall et al., 2020a] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020a). NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*. [27](#), [85](#), [86](#), [87](#), [97](#), [98](#), [100](#), [171](#), [177](#), [181](#), [186](#), [195](#)
- [Mildenhall et al., 2020b] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020b). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *CVPR*, volume abs/2003.08934. [119](#)
- [Miller, 1995] Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41. [128](#), [132](#)

- [Mirzaei et al., 2023] Mirzaei, A., Aumentado-Armstrong, T., Derpanis, K. G., Kelly, J., Brubaker, M. A., Gilitschenski, I., and Levinshstein, A. (2023). SPIn-NeRF: Multiview segmentation and perceptual inpainting with neural radiance fields. In *CVPR*. 179
- [Moreau et al., 2022] Moreau, A., Piasco, N., Tsishkou, D., Stanculescu, B., and de La Fortelle, A. (2022). LENS: Localization enhanced by NeRF synthesis. In *Conference on Robot Learning*. 183
- [Mori et al., 2020] Mori, S., Erat, O., Broll, W., Saito, H., Schmalstieg, D., and Kalkofen, D. (2020). Inpaintfusion: Incremental RGB-D inpainting for 3d scenes. *IEEE Trans. Vis. Comput. Graph.*, 26(10):2994–3007. 176
- [Mori et al., 2022] Mori, S., Schmalstieg, D., and Kalkofen, D. (2022). Good keyframes to inpaint. *IEEE TVCG*. 176
- [Müller et al., 2022] Müller, T., Evans, A., Schied, C., Foco, M., Bódis-Szomorú, A., Deutsch, I., Shelley, M., and Keller, A. (2022). Instant neural radiance fields. In *ACM SIGGRAPH 2022*. 181, 211
- [Murez et al., 2020] Murez, Z., van As, T., Bartolozzi, J., Sinha, A., Badrinarayanan, V., and Rabinovich, A. (2020). Atlas: End-to-end 3d scene reconstruction from posed images. In *ECCV*. 31, 119
- [Narita et al., 2019] Narita, G., Seno, T., Ishikawa, T., and Kaji, Y. (2019). PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things. In *International Conference on Intelligent Robots and Systems (IROS)*. 120, 145, 155, 159
- [Nathan Silberman and Fergus, 2012] Nathan Silberman, Derek Hoiem, P. K. and Fergus, R. (2012). Indoor segmentation and support inference from rgb-d images. In *ECCV*. 7, 116, 128
- [NavVis, 2021] NavVis (2021). Vlx-2. <https://www.navvis.com/vlx-2>. Accessed: 2023-10-02. 21

- [Nekrasov et al., 2021] Nekrasov, A., Schult, J., Litany, O., Leibe, B., and Engelmann, F. (2021). Mix3D: Out-of-Context Data Augmentation for 3D Scenes. In *International Conference on 3D Vision (3DV)*. 115, 119, 144, 155, 157
- [Newcombe et al., 2011a] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011a). KinectFusion: Real-time dense surface mapping and tracking. 33, 85, 89, 118, 120
- [Newcombe et al., 2011b] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011b). Dtam: Dense tracking and mapping in real-time. In *ICCV*. 83
- [Newson et al., 2014] Newson, A., Almansa, A., Fradet, M., Gousseau, Y., and Pérez, P. (2014). Video inpainting of complex scenes. *SIAM J. Imaging Sci.*, 7(4):1993–2019. 175
- [Nguyen-Phuoc et al., 2019] Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., and Yang, Y.-L. (2019). Hologan: Unsupervised learning of 3D representations from natural images. In *CVPR*. 178
- [Nichol et al., 2021] Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. (2021). Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*. 174
- [Niemeyer et al., 2022] Niemeyer, M., Barron, J. T., Mildenhall, B., Sajjadi, M. S. M., Geiger, A., and Radwan, N. (2022). RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*. 177, 192, 193
- [Niemeyer et al., 2020] Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2020). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*. 26, 27
- [Nießner et al., 2013] Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6):169:1–169:11. 25, 29, 35, 36

- [Oechsle et al., 2019] Oechsle, M., Mescheder, L., Niemeyer, M., Strauss, T., and Geiger, A. (2019). Texture fields: Learning texture representations in function space. In *ICCV*. 27, 85, 94, 95
- [Oechsle et al., 2020] Oechsle, M., Niemeyer, M., Reiser, C., Mescheder, L., Strauss, T., and Geiger, A. (2020). Learning implicit surface light fields. In *International Conference on 3D Vision (3DV)*. 27
- [Ost et al., 2021] Ost, J., Mannan, F., Thurey, N., Knodt, J., and Heide, F. (2021). Neural scene graphs for dynamic scenes. In *CVPR*. 178
- [Ozturkcan, 2021] Ozturkcan, S. (2021). Service innovation: Using augmented reality in the IKEA Place app. *Journal of Information Technology Teaching Cases*, 11(1). 183
- [Park et al., 2019] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*. 26, 47, 48, 49, 67, 75, 76, 177, 218, 219, 220, 221, 222, 223
- [Paschalidou et al., 2018] Paschalidou, D., Ulusoy, A. O., Schmitt, C., Gool, L. V., and Geiger, A. (2018). Raynet: Learning volumetric 3d reconstruction with ray potentials. In *CVPR*, pages 3897–3906. 28
- [Pathak et al., 2016] Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2536–2544. IEEE Computer Society. 174
- [Patwardhan et al., 2005] Patwardhan, K. A., Sapiro, G., and Bertalmío, M. (2005). Video inpainting of occluding and occluded objects. In *Proceedings of the 2005 International Conference on Image Processing, ICIP 2005, Genoa, Italy, September 11-14, 2005*, pages 69–72. IEEE. 175

- [Peng et al., 2023] Peng, S., Genova, K., Jiang, C. M., Tagliasacchi, A., Pollefeys, M., and Funkhouser, T. (2023). OpenScene: 3D Scene Understanding with Open Vocabularies. In *CVPR*. 115, 210
- [Peng et al., 2020] Peng, S., Niemeyer, M., Mescheder, L. M., Pollefeys, M., and Geiger, A. (2020). Convolutional occupancy networks. In *ECCV*. 26
- [Penner and Zhang, 2017] Penner, E. and Zhang, L. (2017). Soft 3d reconstruction for view synthesis. 27
- [Pfister et al., 2000] Pfister, H., Zwicker, M., Van Baar, J., and Gross, M. (2000). Surfels: Surface elements as rendering primitives. 17, 25
- [Pham et al., 2019] Pham, Q., Hua, B., Nguyen, D. T., and Yeung, S. (2019). Real-Time Progressive 3D Semantic Segmentation for Indoor Scenes. 120, 159
- [Philip and Drettakis, 2018] Philip, J. and Drettakis, G. (2018). Plane-based multi-view inpainting for image-based rendering in large scenes. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 179
- [PrimeSense,] PrimeSense. Primesense carmine. <http://web.archive.org/web/20131102094504/http://www.primesense.com/solutions/techno>
Accessed: 2023-10-19. 22
- [Qi et al., 2017a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). 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. 114
- [Qi et al., 2017b] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30. 114
- [Radford et al., 2021] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning Transferable Visual Models From Natural

- Language Supervision. In *International Conference on Machine Learning (ICML)*. 126
- [Rahaman et al., 2019] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. (2019). On the spectral bias of neural networks. In *International Conference on Machine Learning (ICML)*. 26
- [Ranftl et al., 2021] Ranftl, R., Bochkovskiy, A., and Koltun, V. (2021). Vision transformers for dense prediction. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 12159–12168. IEEE. 21
- [Ranftl et al., 2022] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. (2022). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(3):1623–1637. 21
- [Ravi et al., 2020] Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.-Y., Johnson, J., and Gkioxari, G. (2020). Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*. 154
- [Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788. 112
- [Rematas and Ferrari, 2020] Rematas, K. and Ferrari, V. (2020). Neural voxel renderer: Learning an accurate and controllable rendering tool. In *CVPR*. 27
- [Rematas et al., 2022] Rematas, K., Liu, A., Srinivasan, P. P., Barron, J. T., Tagliasacchi, A., Funkhouser, T., and Ferrari, V. (2022). Urban radiance fields. In *CVPR*. 178
- [Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28. 112

- [Richard et al., 2019] Richard, A., Cherabier, I., Oswald, M. R., Tsiminaki, V., Pollefeys, M., and Schindler, K. (2019). Learned multi-view texture super-resolution. In *International Conference on 3D Vision (3DV)*. 33
- [Riegler and Koltun, 2020] Riegler, G. and Koltun, V. (2020). Free view synthesis. In *ECCV*. 27
- [Riegler et al., 2017a] Riegler, G., Ulusoy, A. O., Bischof, H., and Geiger, A. (2017a). Octnetfusion: Learning depth fusion from data. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 57–66. 28, 67, 68
- [Riegler et al., 2017b] Riegler, G., Ulusoy, A. O., and Geiger, A. (2017b). Octnet: Learning deep 3d representations at high resolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6620–6629. 28
- [Rockwell et al., 2021] Rockwell, C., Fouhey, D. F., and Johnson, J. (2021). PixelSynth: Generating a 3D-consistent experience from a single image. In *ICCV*. 178, 179, 194, 196, 197, 200, 201
- [Roessle et al., 2022] Roessle, B., Barron, J. T., Mildenhall, B., Srinivasan, P. P., and Nießner, M. (2022). Dense depth priors for neural radiance fields from sparse input views. In *CVPR*. 178, 187
- [Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing. 42, 113, 149
- [Rother et al., 2004] Rother, C., Kolmogorov, V., and Blake, A. (2004). "grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314. 113
- [Rothermel et al., 2016] Rothermel, M., Haala, N., and Fritsch, D. (2016). A median-based depthmap fusion strategy for the generation of oriented points. In

ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, volume III-3. 29

- [Rozenberszki et al., 2022] Rozenberszki, D., Litany, O., and Dai, A. (2022). Language-Grounded Indoor 3D Semantic Segmentation in the Wild. In *ECCV*. 115, 127
- [Rünz et al., 2018] Rünz, M., Buffier, M., and Agapito, L. (2018). MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects. 120
- [Runz et al., 2020] Runz, M., Li, K., Tang, M., Ma, L., Kong, C., Schmidt, T., Reid, I., Agapito, L., Straub, J., Lovegrove, S., et al. (2020). FroDO: From detections to 3D objects. In *CVPR*. 177
- [Russell et al., 2008] Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77:157–173. 115
- [Saito et al., 2019] Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., and Li, H. (2019). PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*. 26, 27, 177
- [Saito et al., 2020] Saito, S., Simon, T., Saragih, J., and Joo, H. (2020). PIFuHD: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *CVPR*. 26, 27, 177
- [Sandström et al., 2022] Sandström, E., Oswald, M. R., Kumar, S., Weder, S., Yu, F., Sminchisescu, C., and Van Gool, L. (2022). Learning online multi-sensor depth fusion. In *European Conference on Computer Vision*, pages 87–105. Springer. 12
- [Sara Fridovich-Keil and Alex Yu et al., 2021] Sara Fridovich-Keil and Alex Yu, Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. (2021). Plenoxels: Radiance fields without neural networks. In *CVPR*. 177

- [Savinov et al., 2016] Savinov, N., Häne, C., Ladicky, L., and Pollefeys, M. (2016). Semantic 3d reconstruction with continuous regularization and ray potentials using a visibility consistency constraint. In *CVPR*, pages 5460–5469. [28](#), [35](#), [51](#), [52](#)
- [Savinov et al., 2015] Savinov, N., Ladicky, L., Hane, C., and Pollefeys, M. (2015). Discrete optimization of ray potentials for semantic 3d reconstruction. In *CVPR*, pages 5511–5518. [28](#), [35](#)
- [Savva et al., 2019] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., and Batra, D. (2019). Habitat: A Platform for Embodied AI Research. In *ICCV*. [95](#), [96](#), [117](#)
- [Sayed et al., 2022] Sayed, M., Gibson, J., Watson, J., Prisacariu, V., Firman, M., and Godard, C. (2022). SimpleRecon: 3D reconstruction without 3D convolutions. In *ECCV*. [20](#), [31](#), [194](#), [197](#)
- [Schneider et al., 2016] Schneider, L., Cordts, M., Rehfeld, T., Pfeiffer, D., Enzweiler, M., Franke, U., Pollefeys, M., and Roth, S. (2016). Semantic Stixels: Depth is not enough. [119](#)
- [Schönberger and Frahm, 2016] Schönberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. [20](#), [23](#), [52](#), [71](#), [178](#), [184](#)
- [Schönberger et al., 2016] Schönberger, J. L., Zheng, E., Pollefeys, M., and Frahm, J.-M. (2016). Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*. [52](#), [71](#), [178](#), [184](#)
- [Schöps et al., 2017a] Schöps, T., Oswald, M. R., Speciale, P., Yang, S., and Pollefeys, M. (2017a). Real-time view correction for mobile devices. *IEEE TVCG*. [83](#)
- [Schöps et al., 2017b] Schöps, T., Sattler, T., Häne, C., and Pollefeys, M. (2017b). Large-scale outdoor 3d reconstruction on a mobile device. *Computer Vision and Image Understanding (CVIU)*. [85](#)

- [Schuhmann et al., 2021] Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J., and Komatsuzaki, A. (2021). Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*. [126](#)
- [Schult et al., 2023] Schult, J., Engelmann, F., Hermans, A., Litany, O., Tang, S., and Leibe, B. (2023). Mask3D for 3D Semantic Instance Segmentation. In *International Conference on Robotics and Automation (ICRA)*. [126](#), [139](#)
- [Schwarz et al., 2020] Schwarz, K., Liao, Y., Niemeyer, M., and Geiger, A. (2020). Graf: Generative radiance fields for 3D-aware image synthesis. *NeurIPS*. [178](#)
- [Schöps et al., 2019] Schöps, T., Sattler, T., and Pollefeys, M. (2019). SurfelMeshing: Online surfel-based mesh reconstruction. [25](#), [30](#), [33](#), [94](#), [95](#), [103](#)
- [Seitz and Dyer, 1999] Seitz, S. M. and Dyer, C. R. (1999). Photorealistic scene reconstruction by voxel coloring. *IJCV*. [33](#), [94](#)
- [Sengupta et al., 2013] Sengupta, S., Greveson, E., Shahrokni, A., and Torr, P. H. (2013). Urban 3d semantic modelling using stereo vision. In *2013 IEEE International Conference on robotics and Automation*, pages 580–585. IEEE. [114](#)
- [Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905. [113](#)
- [Shih et al., 2020] Shih, M.-L., Su, S.-Y., Kopf, J., and Huang, J.-B. (2020). 3D photography using context-aware layered depth inpainting. In *CVPR*. [179](#)
- [Shotton et al., 2013] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*. IEEE. [xv](#), [36](#), [52](#), [53](#)

- [Siddiqui et al., 2023] Siddiqui, Y., Porzi, L., Bulò, S. R., Müller, N., Nießner, M., Dai, A., and Kotschieder, P. (2023). Panoptic lifting for 3d scene understanding with neural fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9043–9052. [124](#), [131](#)
- [Silberman et al., 2012] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor Segmentation and Support Inference from RGBD Images. pages 746–760. Springer Berlin Heidelberg. [116](#), [124](#), [135](#)
- [Sitzmann et al., 2019a] Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., and Zollhöfer, M. (2019a). Deepvoxels: Learning persistent 3d feature embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2437–2446. [xvi](#), [27](#), [60](#), [85](#), [86](#), [87](#), [95](#), [96](#), [102](#), [103](#)
- [Sitzmann et al., 2019b] Sitzmann, V., Zollhöfer, M., and Wetzstein, G. (2019b). Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 1119–1130. [xvi](#), [27](#), [85](#), [86](#), [87](#), [96](#), [102](#)
- [Smith and Topin, 2017] Smith, L. N. and Topin, N. (2017). Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120. [154](#)
- [Song et al., 2015] Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576. [116](#), [128](#)
- [Song et al., 2017] Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T. A. (2017). Semantic scene completion from a single depth image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 190–198. IEEE Computer Society. [176](#)

- [Srinivasan et al., 2019] Srinivasan, P. P., Tucker, R., Barron, J. T., Ramamoorthi, R., Ng, R., and Snavely, N. (2019). Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*. 27
- [Stathopoulou and Remondino, 2023] Stathopoulou, E. K. and Remondino, F. (2023). A survey on conventional and learning-based methods for multi-view stereo. *The Photogrammetric Record*. 20
- [Steinbrücker et al., 2013] Steinbrücker, F., Kerl, C., and Cremers, D. (2013). Large-scale multi-resolution surface reconstruction from RGB-D sequences. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 3264–3271. 29, 36
- [Stereolabs, 2019] Stereolabs (2019). Zed 2. <https://www.stereolabs.com/zed-2/>. Accessed: 2023-10-02. 21
- [Stier et al., 2021] Stier, N., Rich, A., Sen, P., and Höllerer, T. (2021). Vortex: Volumetric 3d reconstruction with transformers for voxelwise view selection and fusion. In *2021 International Conference on 3D Vision (3DV)*, pages 320–330. IEEE. 31
- [Straub et al., 2019] Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J. J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H. M., Nardi, R. D., Goesele, M., Lovegrove, S., and Newcombe, R. (2019). The Replica dataset: A digital replica of indoor spaces. *arXiv*. xvi, xix, 95, 96, 100, 117, 133, 138
- [Stückler and Behnke, 2014] Stückler, J. and Behnke, S. (2014). Multi-resolution surfel maps for efficient dense 3d modeling and tracking. *J. Visual Communication and Image Representation*, 25(1):137–147. 30
- [Stutz and Geiger, 2018] Stutz, D. and Geiger, A. (2018). Learning 3d shape completion from laser scan data with weak supervision. In *IEEE Conference*

- on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society. 67
- [Sucar et al., 2021] Sucar, E., Liu, S., Ortiz, J., and Davison, A. (2021). iMAP: Implicit mapping and positioning in real-time. In *ICCV*. 31, 178
- [Sulaiman et al., 2020] Sulaiman, M. Z., Aziz, M. N. A., Bakar, M. H. A., Halili, N. A., and Azuddin, M. A. (2020). Matterport: virtual tour as a new marketing approach in real estate business during pandemic COVID-19. In *International Conference of Innovation in Media and Visual Design (IMDES)*. 183
- [Sun et al., 2021] Sun, J., Xie, Y., Chen, L., Zhou, X., and Bao, H. (2021). Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15598–15607. 31
- [Suvorov et al., 2022] Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., and Lempitsky, V. (2022). Resolution-robust large mask inpainting with fourier convolutions. In *WACV*. 174, 183, 185, 191, 192, 193, 194, 196, 197, 198
- [Szeliski and Golland, 1998] Szeliski, R. and Golland, P. (1998). Stereo matching with transparency and matting. In *ICCV*. 33
- [Takai et al., 2010] Takai, T., Hilton, A., and Mastuyama, T. (2010). Harmonised texture mapping. In *International Conference on 3D Vision (3DV)*. 32
- [Tarini et al., 2017] Tarini, M., Yuksel, C., and Lefebvre, S. (2017). Rethinking texture mapping. In *ACM SIGGRAPH 2017 Courses*. 33
- [Tewari et al., 2020] Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J., Nießner, M., Pandey, R., Fanello, S., Wetzstein, G., Zhu, J.-Y., Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D. B., and Zollhöfer, M. (2020). State of the art on neural rendering. *Comput. Graph. Forum*. 27

- [Theobalt et al., 2007] Theobalt, C., Ahmed, N., Lensch, H. P. A., Magnor, M. A., and Seidel, H.-P. (2007). Seeing people in different light-joint shape, motion, and reflectance capture. *IEEE TVCG*. 32
- [Thies et al., 2019] Thies, J., Zollhöfer, M., and Nießner, M. (2019). Deferred neural rendering: Image synthesis using neural textures. 27
- [Thies et al., 2020] Thies, J., Zollhöfer, M., Theobalt, C., Stamminger, M., and Nießner, M. (2020). Image-guided neural object rendering. In *ICLR*. 27
- [Thomee et al., 2016] Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. (2016). Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73. 126
- [Thonat et al., 2016] Thonat, T., Shechtman, E., Paris, S., and Drettakis, G. (2016). Multi-view inpainting for image-based scene editing and rendering. In *3DV*. 176
- [Tou and Gonzalez, 1974] Tou, J. T. and Gonzalez, R. C. (1974). Pattern recognition principles. 112
- [Tremblay et al., 2022] Tremblay, J., Meshry, M., Evans, A., Kautz, J., Keller, A., Khamis, S., Loop, C., Morrical, N., Nagano, K., Takikawa, T., and Birchfield, S. (2022). RTMV: A ray-traced multi-view synthetic dataset for novel view synthesis. *ECCVW*. 177
- [Tsiminaki et al., 2019] Tsiminaki, V., Dong, W., Oswald, M. R., and Pollefeys, M. (2019). Joint multi-view texture super-resolution and intrinsic decomposition. In *BMVC*, page 15. BMVA Press. 32
- [Tsiminaki et al., 2014] Tsiminaki, V., Franco, J.-S., and Boyer, E. (2014). High resolution 3d shape texture from multiple videos. In *CVPR*. 32
- [Ulusoy et al., 2016] Ulusoy, A. O., Black, M. J., and Geiger, A. (2016). Patches, planes and probabilities: A non-local prior for volumetric 3d reconstruction. In *CVPR*, pages 3280–3289. 31

- [Ulusoy et al., 2015] Ulusoy, A. O., Geiger, A., and Black, M. J. (2015). Towards probabilistic volumetric reconstruction using ray potentials. In *2015 International Conference on 3D Vision, 3DV 2015, Lyon, France, October 19-22, 2015*, pages 10–18. 31
- [Ummenhofer and Brox, 2013] Ummenhofer, B. and Brox, T. (2013). Point-based 3d reconstruction of thin objects. In *ICCV*, pages 969–976. xv, 51, 52
- [Valentin et al., 2013] Valentin, J. P., Sengupta, S., Warrell, J., Shahrokni, A., and Torr, P. H. (2013). Mesh based semantic modelling for indoor and outdoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2067–2074. 114
- [Velodyne, 2019] Velodyne (2019). Alpha prime. <https://velodynelidar.com/products/alpha-prime/>. Accessed: 2023-10-02. 21
- [Vineet et al., 2015] Vineet, V., Miksik, O., Lidegaard, M., Nießner, M., Golodetz, S., Prisacariu, V. A., Köhler, O., Murray, D. W., Izadi, S., Pérez, P., et al. (2015). Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 75–82. IEEE. 114, 120, 145
- [Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee. 112
- [Wachter et al., 2014] Wachter, M., Moehrl, N., and Goesele, M. (2014). Let there be color! large-scale texturing of 3d reconstructions. In *ECCV*. 32, 83, 85, 86, 94, 95, 103
- [Wang et al., 2019a] Wang, C., Huang, H., Han, X., and Wang, J. (2019a). Video inpainting by jointly learning temporal structure and spatial details. In *The*

- Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 5232–5239. AAAI Press. [175](#)
- [Wang et al., 2016] Wang, J., Wang, Z., Tao, D., See, S., and Wang, G. (2016). Learning common and specific features for RGB-D semantic segmentation with deconvolutional networks. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V*, volume 9909 of *Lecture Notes in Computer Science*, pages 664–679. Springer. [147](#)
- [Wang et al., 2019b] Wang, K., Gao, F., and Shen, S. (2019b). Real-time scalable dense surfel mapping. In *International Conference on Robotics and Automation (ICRA)*. [25](#), [33](#)
- [Wang et al., 2021] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2021). Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*. [133](#)
- [Wang et al., 2022a] Wang, Q., Li, Z., Salesin, D., Snavely, N., Curless, B., and Kontkanen, J. (2022a). 3D moments from near-duplicate photos. In *CVPR*. [179](#)
- [Wang et al., 2022b] Wang, W., Dai, J., Chen, Z., Huang, Z., Li, Z., Zhu, X., Hu, X., Lu, T., Lu, L., Li, H., et al. (2022b). Internimage: Exploring large-scale vision foundation models with deformable convolutions. *arXiv preprint arXiv:2211.05778*. [114](#), [123](#), [124](#), [126](#)
- [Wang et al., 2018] Wang, Y., Tao, X., Qi, X., Shen, X., and Jia, J. (2018). Image inpainting via generative multi-column convolutional neural networks. *Advances in neural information processing systems*, 31. [174](#)

- [Wang et al., 2004] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE TIP*. 95, 195
- [Weder et al., 2023a] Weder, S., Blum, H., Engelmann, F., and Pollefeys, M. (2023a). Labelmaker: Automatic semantic label generation from rgb-d trajectories. *Preprint*. 11
- [Weder et al., 2023b] Weder, S., Engelmann, F., Schönberger, J. L., Seki, A., Pollefeys, M., and Oswald, M. R. (2023b). Alster: A local spatio-temporal expert for online 3d semantic reconstruction. *Preprint*. 11
- [Weder et al., 2023c] Weder, S., Garcia-Hernando, G., Monzpart, A., Pollefeys, M., Brostow, G. J., Firman, M., and Vicente, S. (2023c). Removing objects from neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16528–16538. 11
- [Weder et al., 2020] Weder, S., Schönberger, J. L., Pollefeys, M., and Oswald, M. R. (2020). RoutedFusion: Learning real-time depth map fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11, 60, 75, 76
- [Weder et al., 2021] Weder, S., Schönberger, J. L., Pollefeys, M., and Oswald, M. R. (2021). NeuralFusion: Online Depth Fusion in Latent Space. In *CVPR*. 11
- [Wen et al., 2019] Wen, C., Zhang, Y., Li, Z., and Fu, Y. (2019). Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *ICCV*. 29
- [Wexler et al., 2007] Wexler, Y., Shechtman, E., and Irani, M. (2007). Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3):463–476. 175
- [Whelan et al., 2015] Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J. J., and McDonald, J. (2015). Real-time large-scale dense rgb-d slam with volumetric fusion. *The Int. J. Robotics Research*. 83

- [Whelan et al., 2016] Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). Elasticfusion: Real-time dense SLAM and light source estimation. *The Int. J. Robotics Research*. [25](#), [29](#), [30](#), [33](#), [35](#)
- [Wood et al., 2000] Wood, D. N., Azuma, D. I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D. H., and Stuetzle, W. (2000). Surface light fields for 3d photography. [32](#)
- [Woodford and Vogiatzis, 2012] Woodford, O. J. and Vogiatzis, G. (2012). A generative model for online depth fusion. In *ECCV*, pages 144–157. [30](#)
- [Worrall et al., 2017] Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. (2017). Interpretable transformations with encoder-decoder networks. In *ICCV*. [27](#)
- [Wu et al., 2022] Wu, Q., Liu, X., Chen, Y., Li, K., Zheng, C., Cai, J., and Zheng, J. (2022). Object-compositional neural implicit surfaces. In *ECCV*. [178](#), [183](#)
- [Wu and Leahy, 1993] Wu, Z. and Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113. [113](#)
- [Wu et al., 2015] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1912–1920. IEEE Computer Society. [xvii](#), [67](#), [68](#), [70](#), [71](#), [78](#), [225](#)
- [Xia et al., 2018] Xia, F., R. Zamir, A., He, Z.-Y., Sax, A., Malik, J., and Savarese, S. (2018). Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE. [117](#)
- [Xiao et al., 2013] Xiao, J., Owens, A., and Torralba, A. (2013). Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE international conference on computer vision*, pages 1625–1632. [116](#)

- [Xie et al., 2023] Xie, S., Zhang, Z., Lin, Z., Hinz, T., and Zhang, K. (2023). Smartbrush: Text and shape guided object inpainting with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22428–22437. [174](#)
- [Xu et al., 2019a] Xu, Q., Wang, W., Ceylan, D., Mech, R., and Neumann, U. (2019a). DISN: deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 490–500. [26](#)
- [Xu et al., 2019b] Xu, R., Li, X., Zhou, B., and Loy, C. C. (2019b). Deep flow-guided video inpainting. In *CVPR*. [175](#), [191](#)
- [Xu et al., 2019c] Xu, X., Chen, Y.-C., and Jia, J. (2019c). View independent generative adversarial network for novel view synthesis. In *ICCV*. [177](#)
- [Yang et al., 2021] Yang, B., Zhang, Y., Xu, Y., Li, Y., Zhou, H., Bao, H., Zhang, G., and Cui, Z. (2021). Learning object-compositional neural radiance field for editable scene rendering. In *ICCV*. [178](#), [183](#), [194](#), [196](#), [197](#), [200](#), [201](#)
- [Ylimäki et al., 2018] Ylimäki, M., Kannala, J., and Heikkilä, J. (2018). Accurate 3-d reconstruction with rgb-d cameras using depth map fusion and pose refinement. *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1977–1982. [31](#)
- [Yu et al., 2021] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., and Kanazawa, A. (2021). PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*. [177](#)
- [Yu et al., 2020] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645. [124](#)
- [Yu et al., 2018] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. S. (2018). Generative image inpainting with contextual attention. In *Proceedings*

- of the *IEEE conference on computer vision and pattern recognition*, pages 5505–5514. [174](#)
- [Yu et al., 2002] Yu, S. X., Gross, R., and Shi, J. (2002). Concurrent object recognition and segmentation by graph partitioning. *Advances in neural information processing systems*, 15. [113](#)
- [Yu and Shi, 2004] Yu, S. X. and Shi, J. (2004). Segmentation given partial grouping constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):173–183. [113](#)
- [Yu et al., 2022] Yu, Z., Chen, A., Antic, B., Peng, S. P., Bhattacharyya, A., Niemeyer, M., Tang, S., Sattler, T., and Geiger, A. (2022). Sdfstudio: A unified framework for surface reconstruction. [131](#), [133](#)
- [Yuan et al., 2022] Yuan, Y.-J., Sun, Y.-T., Lai, Y.-K., Ma, Y., Jia, R., and Gao, L. (2022). NeRF-editing: geometry editing of neural radiance fields. In *CVPR*. [181](#)
- [Yuksel et al., 2010] Yuksel, C., Keyser, J., and House, D. H. (2010). Mesh colors. [33](#)
- [Yuksel et al., 2019] Yuksel, C., Lefebvre, S., and Tarini, M. (2019). Rethinking texture mapping. In *Comput. Graph. Forum*. [33](#)
- [Zach, 2008] Zach, C. (2008). Fast and high quality fusion of depth maps. In *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*. [35](#), [51](#), [52](#)
- [Zach et al., 2007] Zach, C., Pock, T., and Bischof, H. (2007). A globally optimal algorithm for robust tv-l1 range image integration. In *ICCV*, pages 1–8. [28](#), [35](#)
- [Zeng et al., 2020] Zeng, Y., Fu, J., and Chao, H. (2020). Learning joint spatial-temporal transformations for video inpainting. In *ECCV*. [175](#)
- [Zeng et al., 2019] Zeng, Y., Fu, J., Chao, H., and Guo, B. (2019). Learning pyramid-context encoder network for high-quality image inpainting. In *CVPR*. [174](#)

- [Zhang et al., 2023a] Zhang, G., Ji, J., Zhang, Y., Yu, M., Jaakkola, T. S., and Chang, S. (2023a). Towards coherent image inpainting using denoising diffusion implicit models. 174
- [Zhang et al., 2023b] Zhang, J., Liu, H., Yang, K., Hu, X., Liu, R., and Stiefelhaugen, R. (2023b). Cmx: Cross-modal fusion for rgb-x semantic segmentation with transformers. *IEEE Transactions on Intelligent Transportation Systems*. 126
- [Zhang et al., 2020] Zhang, J., Zhu, C., Zheng, L., and Xu, K. (2020). Fusion-Aware Point Convolution for Online Semantic 3D Scene Segmentation. In *CVPR*. 121, 145, 154, 155, 157, 159
- [Zhang et al., 2018] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*. 195
- [Zhang et al., 2021] Zhang, X., Fanello, S., Tsai, Y.-T., Sun, T., Xue, T., Pandey, R., Orts-Escolano, S., Davidson, P., Rhemann, C., Debevec, P., et al. (2021). Neural light transport for relighting and view synthesis. 27
- [Zhi et al., 2019] Zhi, S., Bloesch, M., Leutenegger, S., and Davison, A. J. (2019). Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *CVPR*, pages 11776–11785. 31, 120
- [Zhi et al., 2021a] Zhi, S., Laidlow, T., Leutenegger, S., and Davison, A. (2021a). In-place scene labelling and understanding with implicit scene representation. In *ICCV*. 178
- [Zhi et al., 2021b] Zhi, S., Laidlow, T., Leutenegger, S., and Davison, A. J. (2021b). In-place scene labelling and understanding with implicit scene representation. In *ICCV*. 117, 119, 124, 133, 134, 135, 138, 141
- [Zhirong Wu et al., 2015] Zhirong Wu, Song, S., Khosla, A., Fisher Yu, Linguang Zhang, Xiaoou Tang, and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920. xv, 46, 47, 56

- [Zhou et al., 2017] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2017). Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641. [114](#), [116](#), [124](#), [126](#)
- [Zhou et al., 2020] Zhou, H., Ummenhofer, B., and Brox, T. (2020). Deeptam: Deep tracking and mapping with convolutional neural networks. *Int. J. Comput. Vis.*, 128(3):756–769. [31](#)
- [Zhou and Koltun, 2013] Zhou, Q. and Koltun, V. (2013). Dense scene reconstruction with points of interest. *ACM Trans. Graph.*, 32(4):112:1–112:8. [xv](#), [xvi](#), [xvii](#), [xix](#), [50](#), [52](#), [54](#), [71](#), [81](#), [226](#)
- [Zhu et al., 2020] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*. [112](#)
- [Zhu et al., 2022] Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M. R., and Pollefeys, M. (2022). Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796. [32](#)
- [Zienkiewicz et al., 2016] Zienkiewicz, J., Tsiotsios, A., Davison, A. J., and Leutenegger, S. (2016). Monocular, real-time surface reconstruction using dynamic level of detail. In *Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016*, pages 37–46. [30](#)
- [Zollhöfer et al., 2015a] Zollhöfer, M., Dai, A., Innmann, M., Wu, C., Stamminger, M., Theobalt, C., and Nießner, M. (2015a). Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics (TOG)*, 34(4). [28](#), [33](#)
- [Zollhöfer et al., 2015b] Zollhöfer, M., Dai, A., Innmann, M., Wu, C., Stamminger, M., Theobalt, C., and Nießner, M. (2015b). Shading-based Refinement on Volumetric Signed Distance Functions. [33](#)

- [Zollhöfer et al., 2018] Zollhöfer, M., Stotko, P., Görlitz, A., Theobalt, C., Nießner, M., Klein, R., and Kolb, A. (2018). State of the Art on 3D Reconstruction with RGB-D Cameras. *Computer Graphics Forum (Eurographics State of the Art Reports 2018)*, 37(2). [33](#)