THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire Jean Kuntzmann

Capture volumétrique à partir d'images RVB basée sur le rendu différentiel

Volumetric capture from RGB images using differential rendering

Présentée par :

Briac TOUSSAINT

Direction de thèse :

Jean-Sébastien FRANCO DIRECTEUR DE RECHERCHE, CENTRE INRIA UNIVERSITE GRENOBLE ALPES	Directeur de thèse
Edmond BOYER SENIOR SCIENTIST, Meta Reality Labs Zurich	Co-directeur de thèse

Rapporteurs :

ADRIAN HILTON
FULL PROFESSOR, UNIVERSITY OF SURREY
GEORGE DRETTAKIS
DIRECTEUR DE RECHERCHE, CENTRE INRIA D'UNIVERSITE COTE D'AZUR

Thèse soutenue publiquement le **16 septembre 2025**, devant le jury composé de :

GEORGES-PIERRE BONNEAU, PROFESSEUR DES UNIVERSITES, UNIVERSITE GRENOBLE ALPES	Président
JEAN-SEBASTIEN FRANCO, DIRECTEUR DE RECHERCHE, CENTRE INRIA UNIVERSITE GRENOBLE ALPES	Directeur de thèse
ADRIAN HILTON, FULL PROFESSOR, UNIVERSITY OF SURREY	Rapporteur
GEORGE DRETTAKIS, DIRECTEUR DE RECHERCHE, CENTRE INRIA D'UNIVERSITE COTE D'AZUR	Rapporteur
CHRISTIAN THEOBALT, FULL PROFESSOR, MAX PLANCK INSTITUTE	Examinateur
TORSTEN SATTLER, SCIENTIST, CIIRC	Examinateur

Invités :

DIEGO THOMAS
ASSOCIATE PROFESSOR, KYUSHU UNIVERSITY
EDMOND BOYER
SENIOR SCIENTIST, META Reality Labs

Abstract

Volumetric capture aims to recover the shape and appearance of a person from multi-view inputs. Here, we assume the case of a passive capture setup built from synchronized color cameras alone. Based on this input modality, this thesis investigates the applicability of differential rendering for high-quality yet efficient 3D reconstruction.

We define two competing goals that a reconstruction algorithm should meet to be effective in this setting. The most important one is to achieve high accuracy, both in terms of geometry and photo-realism of newly generated viewpoints. The second goal is to do so parsimoniously, under limited computational resources. The latter requirement is difficult to fulfill along with the first one, but is especially important once temporal sequences and high-resolution input images come into play.

Differential rendering has gained traction in the research community for 3D reconstruction, outperforming traditional multi-view stereo in challenging cases, while also being capable of very realistic novel view synthesis. Thus, differential rendering appears as a promising tool for volumetric capture, and could similarly improve over classic approaches in this context. However, most differential rendering approaches are penalized by disproportionate computational requirements, incompatible with our targeted use case, and limited scalability.

Consequently, this thesis will investigate ways of addressing the efficiency and scalability constraints. To this end, we build on recent volumetric differential rendering approaches and propose a disentangled representation of shape and color, efficiently balancing responsibilities between the geometry and the appearance model. We achieve scalability thanks to a sparse grid, whose topology adapts dynamically during training. We also contribute a minimalist embedding of appearance, by explicitly modeling the intrinsic and extrinsic appearance components of the surface. This allows us to reconstruct shiny materials with few parameters. In addition, we propose a novel dataset of clothed mannequins to benchmark reconstruction algorithms on real images taken in a capture platform, allowing for an unbiased evaluation in this setting. We evaluate our work against a panel of classic multi-view stereo approaches, implicit surface differential renderers and 3D Gaussian splatting approaches, demonstrating clear improvements in terms of accuracy and efficiency over the state-of-the-art.

This investigation proves successful as we achieve millimeter accuracy on said benchmark and real-time novel view synthesis, with about a minute of computations on a single GPU. These results generalize not only to human scenes but also to generic objects, as we prove by evaluating our approach on representative human and object reconstruction benchmarks.

Résumé

La capture volumétrique vise à récupérer la forme et l'apparence d'une personne. Nous prenons ici le cas d'une plateforme de capture passive, contenant uniquement des caméras couleur synchronisées. À partir de cette modalité d'entrée, cette thèse étudie l'applicabilité du rendu différentiel à cette tâche.

Nous définissons deux objectifs concurrents qu'un algorithme de reconstruction doit atteindre pour être efficace dans ce contexte. Le plus important est d'atteindre une grande précision, tant en termes de géométrie que de photoréalisme. Le second objectif est d'y parvenir avec parcimonie, avec des ressources de calcul limitées. Cette dernière exigence devient particulièrement importante si l'on considère des séquences temporelles et des images d'entrée haute résolution.

Le rendu différentiel permet de faire le rendu de nouveau points de vue autour d'une scène de façon très convaincante, et des pistes prometteuses ont été proposées pour améliorer la qualité géométrique des formes récupérées. Cependant, ces travaux sont généralement validés sur des bases de données d'objets génériques plutôt que sur des humains réels dans un contexte de capture volumétrique, avec des budgets de calcul disproportionnés ou une extensibilité restreinte.

S'inspirant de ces approches prometteuses, nos travaux exploitent le rendu différentiel volumétrique où la forme est paramétrée soit par un champ d'opacité, ou bien par une surface implicite. Nous développons un nouveau pipeline pour répondre aux contraintes d'efficacité et d'extensibilité, avec des changements majeurs par rapport à une implémentation naïve. En particulier, nous basons notre méthodologie sur une grille de voxels creuse avec une stratégie de réallocation dynamique. Nous développons un nouveau modèle d'apparence, à la fois expressif et efficace, permettant de reconstruire des matériaux brillants à faible coût. Nous incluons également une initialisation basée sur les silhouettes et une supervision à partir d'images d'arrière-plan, applicables spécifiquement dans une plateforme de capture.

De plus, nous proposons un nouveau jeu de données constitué de mannequins habillés, pour comparer les algorithmes de reconstruction sur des images réelles prises dans une plateforme de capture, et permettant une évaluation juste et impartiale dans ce contexte. Nous évaluons nos travaux par rapport à un panel d'approches stéréo multi-vues classiques, d'approches de rendu différentiel de surface implicite et d'approches de splatting de gaussiennes 3D, et démontrons des améliorations en termes de précision et d'efficacité par rapport à l'état de l'art.

Cette étude s'avère concluante : nous obtenons une précision millimétrique sur ce benchmark et un rendu en temps réel, avec environ une minute de calcul sur un seul GPU. Ces résultats sont généralisables non seulement aux scènes humaines, mais aussi aux objets génériques, comme nous le démontrons en évaluant notre approche sur des benchmarks représentatifs de reconstruction d'objets et d'humains.

Acknowledgements

I would like to express my gratitude to my advisor, Jean-Sébastien, for his tireless and deep investment in my PhD, both at a technical and academic level, as well as for giving me a lot of freedom in my work organization, research topics and methodology. I would also like to thank Edmond and Diego for their expertise and guidance, as well as other researchers and administrative assistants: Stefanie, Sergi, Julien, Fabrice, Nathalie. In particular, I am grateful for Diego's warm welcome during my visit to Kyushu University.

I thank the various interns, engineers and PhD students whom I've shared an office with — Boyao, Maxime, Hector, David, Youssef, Laura, Mattias — for their interest in my work and the long discussions that invariably ensued. More broadly, I thank all my colleagues for the good times we shared at Inria and elsewhere, and for organizing outings: skiing, cycling, swimming, hiking, eating out, dancing, going to the cinema or just ungluing me from the computer screen for a coffee break. In no specific order: Nicolas, Mathieu, Pierre, Anil, Rim, Aymen, Samara, Abdel, Vaibhav, Leo, Antoine, Mohammed, Itzel, Bashwar, Leon, Felix, Gagan, Vicente, Gabriel and most certainly a few others I forgot to include here.

Lastly, I thank my family for their care, support and genuine interest in my work during these four years of PhD.

Contents

Contents **7**

List of Figures **10**

List of Tables **16**

1 Introduction **19**
1.1 Context and Motivations 19
1.2 Problem statement 20
1.3 Challenges 21
1.3.1 Accounting for the domain gap 21
1.3.2 Reconciling geometric and photometric accuracy 23
1.3.3 Achieving scalability 24
1.4 Outline and Contributions 25
1.4.1 A parsimonious and scalable framework for hybrid volumetric / surface reconstruction 25
1.4.2 A disentangled representation of geometry and appearance 25
1.4.3 A local spatio-angular factorization of appearance 25
1.4.4 A novel dataset for unbiased reconstruction benchmarking 26
1.5 Publications 26

2 Background and Related Works **27**
2.1 Volumetric capture 27
2.2 Differential rendering 30
2.2.1 Differential volume rendering 32
2.2.2 Gaussian Splatting 33
2.3 Differential rendering for implicit surfaces 34
2.3.1 Differential surface rendering 34
2.3.2 Hybrid rendering 35
2.4 Parameterizations 36
2.4.1 Deep multi-layer perceptrons 37
2.4.2 Dense grids 38
2.4.3 Factorized grids 39
2.4.4 Sparse representations 40
2.4.5 Unstructured representations 42
2.5 Appearance models 43
2.5.1 Spherical harmonics 43
2.5.2 Neural networks 44

2.6 Takeaways 45

3 Fast Gradient Descent for Surface Capture Via Differentiable Rendering 47
3.1 Introduction 47
3.2 Methodology 49
3.2.1 Rendering model 49
3.2.2 Losses and regularization 50
3.2.3 Gradient descent 52
3.2.4 Surface extraction 53
3.3 Sparsity and coarse-to-fine strategies 54
3.3.1 Ray culling 54
3.3.2 Coarse-to-fine strategy 54
3.3.3 Implementation details 55
3.4 Evaluations 55
3.4.1 Validation with general-purpose benchmarks 55
3.4.2 Qualitative results on DTU and NerF datasets 56
3.4.3 Surface capture evaluations 59
3.4.4 Performance analysis 61
3.5 Conclusion 62
3.5.1 Discussion 63

4 Millimetric Human Surface Capture in Minutes 64
4.1 Introduction 64
4.2 Related work 66
4.3 Method 68
4.3.1 Preliminary Notions 68
4.3.2 Key Technical Contributions 69
4.3.3 Sparse Scene Representation 70
4.3.4 Allocation bounds 71
4.3.5 Adaptive Ray-Marching with Deferred SDF Decoding . . 72
4.3.6 Pre-Computed Color Decoding 73
4.3.7 Regularizations and Losses 74
4.3.8 Neural Architecture 75
4.3.9 Output Model size 75
4.4 Experiments 75
4.4.1 Dataset Acquisition Protocol 75
4.4.2 Baselines 76
4.5 Results 78
4.5.1 Quantitative results 78
4.5.2 Detailed quantitative results 79
4.5.3 Volume rendering comparisons 79
4.5.4 Error visualization 79
4.5.5 Ablations 83
4.5.6 Qualitative results on moving subjects 84
4.5.7 Exposure variations and sensor noise 84
4.5.8 Additional experimental details 86
4.6 Conclusion 89

4.7 Acknowledgements 90

5 Light Field Probes for Neural Surface Reconstruction 91
5.1 Introduction 91
5.2 Related Works 93
5.3 Method 94
5.3.1 Decoupled Parametrization of Radiance 94
5.3.2 Light Field Probes on a Coarse Grid 96
5.3.3 SDF Estimation Approach 98
5.3.4 Training 99
5.4 Implementation Details 99
5.4.1 Scheduling 99
5.4.2 Neural Architecture comparison 99
5.5 Experiments 100
5.5.1 Protocol 100
5.5.2 MVMannequins 100
5.5.3 ActorsHQ 101
5.5.4 DTU 106
5.5.5 BlendedMVS 109
5.5.6 Ablations 112
5.5.7 Performance Analysis 114
5.6 Limitations 114
5.7 Conclusion 115
5.8 Acknowledgments 115

6 Conclusion 116
6.1 Summary 116
6.2 Limitations 117
6.3 Future works 118

A Appendix of Chapter 3 119
A.1 Detailed definitions 119
A.1.1 Robust loss 119
A.1.2 Parametrizations 120
A.1.3 Discrete integrals 120
A.1.4 Losses 120
A.2 Analytical gradient derivations 120
A.2.1 Gradients of the photometric loss 120
A.2.2 Gradients of the ray color consistency loss 121
A.2.3 Gradients of the spatial regularization loss 121
A.2.4 Gradients of the spherical harmonics parsimony loss . . . 122
A.2.5 Gradients of the ballooning loss 122
A.2.6 Gradients of the exposure regularization 122
A.3 Optimization 123

B Appendix of Chapter 4 124
B.1 Optimization and visualization loops 124
B.2 Data structures comparison 126
B.3 Volume renderings 126

C Appendix of Chapter 5 **133**
C.1 Detailed hyper-parameters scheduling 133
C.2 Detailed tables 133

Bibliography **133**

List of Figures

1.1 The Kinovis capture platform [1], with 68 4Mpx cameras. 20
1.2 Typical images from the Kinovis capture platform. Note the wide field of views and small projected silhouette. 21
1.3 Motion blur can be seen around the legs during a fast cartwheel motion. 23

2.1 Capture setup including 53 RGB and 53 IR cameras at 4Mpx and unstructured IR laser light sources. Image reproduced from Collet et al. [2]. 28
2.2 Capture setup (left) and corresponding novel view synthesis (right). Image excerpted from Işık et al. [3]'s HumanRF work. The capture platform contains 160 12Mpx RGB cameras. 28
2.3 Real-time reconstructions with 3 RGBD sensors. Image excerpted from Yu et al. [4]. 29
2.4 Common rendering algorithms used in differential rendering. . . 31
2.5 Popular shape representations. 31
2.6 Left: plots of equations 2.10 and 2.11 as a function of the signed distance, for a ray perpendicular to the surface. Right: corresponding volume integration weights $w_i = (1 - \exp(-\sigma_i \delta_i)) \exp(-\sum_{j=0}^{i-1} \sigma_j \delta_j)$. 36
2.7 Deep MLP parametrization 37
2.8 Dense grid parametrization 38
2.9 Left: Factorization of a matrix. Right: Tri-planar factorization. 39
2.10 InstantNGP [5] decomposes the scene as a hierarchy of hashmaps. Collisions become more frequent in deeper levels which results in high-frequency noise. The feature $\mathbf{m}(x)$ is defined as the concatenation of all the levels. 41
2.11 Block-sparse storage with 4^3 blocks and one indirection. 42
2.12 Left: PointNeRF [6]-style Knn interpolation. Right: Interpolation via the Delaunay triangulation of the point cloud as in [7]. . 43
2.13 The first 16 Y_l^m polynomials visualised on the unit sphere. Red (resp. blue) colors denotes positive (resp. negative) values. . . 44
2.14 Generalized neural appearance parametrization. 44

3.1 Typical results of our method: input image (left), colored mesh (center), geometry (right) 48
3.2 Close-up on another mesh: input image (left), colored mesh (center), geometry (right) 48
3.3 Volumetric integral example 50

3.4 Gradient interpolation. Samples (red dots) are spaced uniformly along two adjacent rays. Left: interpolation of the nearest voxels in the forward pass. Right: interpolation of the nearest gradients in the backward pass. 53
3.5 Block-sparse structure with one indirection. 54
3.6 Qualitative results on the DTU dataset. 57
3.7 Qualitative results on the nerf synthetic dataset. 58
3.8 Chamfer heatmaps error comparison on synthetic human benchmark with 3 models, [8] (left columns) vs ours (right columns). . 59
3.9 Ours (columns 2 & 4) vs [8] (columns 1 & 3) 60
3.10 Top: $B = 0$ (right), $B = 1$ (center), $B = 2$ (left) Center: No ballooning term (right), with ballooning (left) Bottom: Without SH parsimony term (top row) vs with (bottom row). From left to right: geometry, mesh colors, $\beta_{0,0}$ only, specularity only . . . 61
3.11 Left: Sectional view of the opacity field. Right: Extracted isosurface. Ours (left), [5] (right) 62

4.1 Left: Reconstructions for 3 examples of dynamic human capture [9], including body and long hair motion. Right: illustration of results on the proposed MVMannequin scanned mannequin database: input view, hand-scanned reference, our reconstruction. Note in both cases, the fine captured surface detail such as hair strands, facial features, cloth folds and creases, and the sewing micro-geometry. 64
4.2 Learnable features and SDF are stored as a sparse hierarchy, with factor decomposition of features at the intermediate level. All leaf voxel tiles are processed for inner SDF smoothness and normal computation. Color tiles are predicted using a tiny MLP with feature product, normal, normal and viewing ray vector dot product, and spherical harmonic projection of the reflected view vector, to preserve its high frequencies. An efficient ray marcher converts the SDF to opacities [10] and integrates the volume to produce pixel-wise color predictions. All orange boxes correspond to one GPU compute kernel in the forward and backward passes. The figure centers on a sample point query on a ray, and materializes corresponding features and voxels in red. . 69
4.3 Our method continuously prunes empty space as optimization progresses. Coarse occupancy tiles are shown in dark gray and fine occupancy tiles are in light gray. Signed-distance values are mapped to a hue to emphasize the level set regularization. The mesh extracted from the level set is shown in the bottom row [11]. 70
4.4 Hand-scanned mannequins. Female (top), male (bottom). 76
4.5 The Kinovis multi-camera platform used to capture the dataset. 77
4.6 Reference scans (top rows) versus our reconstructions (bottom rows) for our 8 female (left) and 6 male (right) mannequins dataset 78
4.7 Completeness and accuracy error curves 79

4.8 Reconstructions on a subset of our proposed dataset with (a) input image, (b) reference scan, (c) our method, (d) Voxurf, (e) NeuS2, (f) Colmap, (g) 2DGS, (h) GOF and their associated accuracy heatmaps. 80
4.9 Qualitative rendering comparisons on the female "cos" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. The Gaussian splatting methods achieve a good image quality overall but tend to fit highly anisotropic Gaussians in textured regions, which impacts the extracted geometry in 2DGS and GOF's cases. Neus2 suffers from noise while Voxurf creates sharp boundaries at depth discontinuities. Our method reconstructs both smooth edges that accurately blend with the background image and detailed textured patterns. 82
4.10 Qualitative rendering comparisons on the female "naked" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 82
4.11 Completeness and accuracy error curves for the ablations. 83
4.12 Our reconstruction without the color smoothness regularizer (left) and with (right). The region around the eyes exhibits a noticeable pixelizing. 83
4.13 Results of our reconstructions on some sequences of the 4DHumanOutfit dataset [9] 84
4.14 Field of view differences imply exposure variations between the cameras. 85
4.15 Crop of the region around the head of the mannequin in figure 4.14 (right). The camera sensors display some level of noise (left) that can be alleviated by temporal averaging (right) but color fringing and staircase artifacts remain due to the Bayer filter. 85
4.16 Convergence failures for Gaussian Opacity Fields (left) due to Gaussians near the cameras occluding the scene, and for 2D Gaussian splatting (right) due to Gaussians above the ground occluding the feet. 87
4.17 Top: Female "jea" scene, bottom: Male "sho" scene. From left to right: Reference scan, Our reconstruction, 2DGS, GOF and their associated accuracy heatmaps. Top: Highly textured areas are not well reconstructed both for 2DGS and GOF, leading to noise and holes. Bottom: We achieve better or equivalent geometric quality than 2DGS and GOF even in regions with less texture. 88

5.1 We design a new appearance model for neural surface approaches, which combines high resolution spatial features and lower resolution angular features for improved reconstruction quality, training and inference speed. We plot the Chamfer distance as a function of training speed for several baselines on MVMannequins (top) and DTU (bottom). 92

5.2 Overview of the color prediction for a single voxel inside a 16^3 tile. The angular features F_a are computed by interpolating and evaluating the spherical harmonics from the 8 corner probes with the reflected vector $\mathbf{r}$. The spatial features F_s are computed as the outer product of three orthogonal planes of resolution 16×16, specific to the tile. A small neural network decodes these inputs into a color. 95
5.3 The reflectivity cannot be encoded in $\mathbf{F}_a$ alone because different incident vectors can map to the same reflected vector (in red). In that case, similar angular features will be obtained for the two viewpoints due to the spatial and angular proximity of the lookups. The angle of incidence disambiguates the two situations. 96
5.4 Left and center: The lighting direction changes at a lower rate on the surface for distant light sources. Right: Self-shadows impose variations in lighting even with directional illumination. 97
5.5 Imprecise segmentation example on ActorsHQ. 101
5.6 High-resolution reconstructions for ActorsHQ. 102
5.7 Qualitative comparison on ActorsHQ. Our method can handle the full resolution images, which lets us reconstruct textured patterns at a sub-millimetric scale. 103
5.8 Reconstruction results on ActorsHQ. 104
5.9 Reconstruction results on ActorsHQ. 105
5.10 Qualitative comparison on DTU. Top: scan 24, middle: scan 37, bottom: scan 63. Left to right: ground truth, Ours, Voxurf, NeuS2, 2DGS. 106
5.11 Reconstructions on challenging examples of DTU. We find that 2DGS excels at reconstructing flat surfaces (doll house roof) but tends to under-perform on reflective materials. 2DGS fails to extract geometry on some parts of the objects (scans 97 and 63). In contrast, our method recovers smooth surfaces even under strong specularities (metal scissors, tuna can and apples). Voxurf struggles on the most shiny materials despite its considerably larger MLP. NeuS2's reconstruction suffers from grid-aligned artifacts, possibly due to discontinuities in its hash-grid interpolation scheme (shoulder of the bunny in scan 110). 107
5.12 Qualitative comparison on DTU. Top: scan 69, middle: scan 97, bottom: scan 110. Left to right: ground truth, Ours, Voxurf, NeuS2, 2DGS. 108
5.13 Reconstructions examples on BlendedMVS. 109
5.14 Accuracy heatmaps on BlendedMVS. The pink color denotes points too far away from the ground truth, which are ignored in the computation of the metrics. Left to right: Ours, Voxurf, NeuS2. Voxurf fails to carve inside the two hemispheres in the stone example and the corners of the base are missing. However, Voxurf is able to carve under the dog statue whereas both NeuS2 and our method fail on this example. NeuS2 is noticably more noisy on all examples. 110
5.15 Qualitative comparison on BlendedMVS. 111

5.16 Our model can be intuitively explained. We disable some components of a pre-trained example from the MVMannequins dataset [12] (left). From left to right, we remove the spatial features (set $\mathbf{F}_s = 0$), only keep the constant coefficient of the SH basis ($l = 1$), disable the incidence angle embedding (set $\mathbf{n} \cdot \mathbf{v} = 1$ in eq. 5.4). The learned Fresnel term encodes grazing reflections (red circles), the angular features provide the shading (green squares) and the spatial features contain the high frequency details. 112
5.17 Effect of the learned Fresnel term on three examples from MVMannequins, DTU and ActorsHQ. Left to right: input image, volume rendering (A), volume rendering w/o the Fresnel term (B) (we set $\mathbf{n} \cdot \mathbf{v} = 1$ in eq. 5.4), difference image ($4 \times |A - B|$). . 113
5.18 We train the scene 63 from DTU with $l = 4$, that we then visualize with l varying from 1 (constant term) to 4, left to right. The corresponding angular features are on the top row, and the predicted color is on the bottom row. We observe that the specularities can be removed simply by disabling the high order spherical harmonics coefficients. 113

B.1 Qualitative rendering comparisons on the female "jea" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 127
B.2 Qualitative rendering comparisons on the female "opt1" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 127
B.3 Qualitative rendering comparisons on the female "opt2" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. The splatting based methods are particularly good at reconstructing high frequency specularities compared to the neural methods, with Voxurf having a noticeable advantage in that regard thanks to its larger MLP. 128
B.5 Qualitative rendering comparisons on the female "opt3" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 128
B.6 Qualitative rendering comparisons on the female "sho" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF 129
B.7 Qualitative rendering comparisons on the female "tig" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 129
B.8 Qualitative rendering comparisons on the male "cos" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 130
B.9 Qualitative rendering comparisons on the male "naked" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 130
B.10 Qualitative rendering comparisons on the male "jea" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 131

B.11 Qualitative rendering comparisons on the male "opt" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. We observe that NeuS2 is sometimes unable to explain the inter-camera color variations from some viewpoints whereas the other baselines do not seem impacted. 131
B.12 Qualitative rendering comparisons on the male "sho" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 132
B.13 Qualitative rendering comparisons on the male "tig" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. 132

List of Tables

3.1 Quantitative results on NeRF synthetic dataset. 56
3.2 Quantitative evaluation on the DTU dataset. 57
3.3 Chamfer evaluation (lower is better). 59
3.4 Ablations of regularization terms on the mic model 61
3.5 PSNR on real data 62

4.1 Indicative baselines characteristics 77
4.2 Top: Comparisons, Bottom: Ablations. Completeness , accuracy and PSNR averaged over all scans. $r/2$, $r/4$ and $r/8$ column denote training with half, quarter and one eighth of the grid and images resolutions respectively. 79
4.3 Per-scene average completeness (mm) 81
4.4 Per-scene average accuracy (mm) 81
4.5 Per-scene average PSNR 81

5.1 Neural architectures in the literature. 100
5.2 Evaluation on MVMannequins with two configurations for (n_s, n_a, l). Per-camera bias vectors are not used (✗-symbol). . . 100
5.3 Evaluation for ActorsHQ. Per-camera bias vectors are not used (✗-symbol). Full, half and quarter-resolution input images are denoted as $r/1$, $r/2$, $r/4$ respectively. Renderings are up-scaled before comparison with the full resolution images (4088×2990). 101
5.4 DTU evaluation with two configurations for (n_s, n_a, l). Per-camera bias vectors are used (✓-symbol). 106
5.5 BMVS evaluation with two configurations for (n_s, n_a, l). Per-camera bias vectors are used (✓-symbol). 109
5.6 Ablation on MVMannequins 112
5.7 We ablate the number of spatial features n_s, angular features n_a and spherical harmonics order l on BlendedMVS. Each triplet denotes a configuration (n_s, n_a, l), the ✓-symbol indicates training with per-camera bias vectors. 112
5.8 Timings in ms 114

C.1 Schedule for DTU 134
C.2 Schedule for BlendedMVS 134
C.3 Schedule for MVMannequins 134
C.4 Schedule for ActorsHQ 134
C.5 MVMannequins per-scene Chamfer (mm) 135
C.6 MVMannequins per-scene PSNR 135

C.7 Detailed Ablation Table. MVMannequins per-scene Chamfer (mm) . . . 135
C.8 Detailed Ablation Table. MVMannequins per-scene PSNR . . . 135
C.9 ActorsHQ per-scene PSNR . . . 135
C.10 DTU per-scene Chamfer (mm) . . . 136
C.11 DTU per-scene PSNR . . . 136
C.12 Detailed Ablation Table. DTU per-scene Chamfer (mm) . . . 136
C.13 Detailed Ablation Table. DTU per-scene PSNR . . . 136
C.14 BlendedMVS per-scene Chamfer . . . 137
C.15 BlendedMVS per-scene PSNR . . . 137
C.16 Detailed Ablation Table. BlendedMVS per-scene Chamfer . . . 137
C.17 Detailed Ablation Table. BlendedMVS per-scene PSNR . . . 137

1 Introduction

1.1 Context and Motivations

Photogrammetry is a process by which the shape and appearance of real-world elements is reproduced in a digital format, from a set of pictures [13]. The entertainment and 3D production industry [14] are showing a growing interest in this technology, as it allows the users to experience an enhanced realism while also reducing labor costs in assets creation. Here, we look at a more specific application of photogrammetry: the reconstruction of clothed humans in motion.

The entertainment industry is again a large application field with several companies already providing such reconstructions [15–18]. Typical purposes range from video game characters or movie doubles [17, 18], music videos or immersive training simulations [15], advertisement or fashion [15, 16].

Medical applications are possible: developmental illnesses such as scoliosis could be diagnosed and monitored semi-automatically from a non-invasive 3D reconstruction. When it comes to research, we envision that shape and motion spaces [19–21] could benefit from training data of higher quality and diversity. The same applies to generative AIs [22] and foundational models [23] for 3D understanding, limited by the lack of 3D content.

Recovering the shape of an object or scene has been largely explored by the computer vision community over the previous decades, using techniques such as multi-view stereo [24, 25]. Multi-view stereo is based on the principle of photo-consistency, which is that a small region on the surface looks similar regardless of the relative orientation of the observer. Importantly, neighboring surface regions should also be dissimilar from each other, *i.e.* they can be discriminated from one another. If these two hypotheses are verified, the distance of the surface points to the cameras can be determined with great accuracy. On the contrary, robustness tends to deteriorate on reflective materials that break the first assumption, as well as on weakly-textured surfaces that are ambiguous. In comparison, differential rendering has shown potential to address

the aforementioned limitations of multi-view stereo. It is a powerful idea that became prominent in the reconstruction literature in recent years [26, 27]. In particular, differential rendering no longer assumes that objects have a mostly Lambertian reflectance, and can compensate the lack of texture through shading cues.



Figure 1.1: The Kinovis capture platform [1], with 68 4Mpx cameras.

In this work, we look at the 3D reconstruction problem applied to the case of a moving person imaged in a synchronized capture platform, as in fig. 1.1. We would like to obtain a very precise estimate of the person's shape and appearance, thus we target the metrological and photometric aspects of the reconstruction problem, rather than its semantic aspect. As will be analysed in §1.3.1, capture platforms present unique characteristics that make the reconstruction problem difficult for stereo-based algorithms [8, 25], and differential rendering appears as a reasonable solution. However, our use case additionally demands sufficient efficiency to process long temporal sequences. Roughly speaking, while some works achieve outstanding results at a great computational expense [28, 29], others typically trade accuracy for speed [5, 30, 31], and may still be too slow regardless. We are looking for a better solution, which leads us to the following question:

- How can differential rendering be adapted for resource-constrained scenarios, taking volumetric capture as our main example?

To answer it, we will explore how the goals of accuracy and efficiency can be achieved together in this context. We will also take interest in the validation of our reconstruction algorithms, in a way that closely relates to the capture of real humans, as no ground truth is available in general. Finally, we will expand our ideas to a wider setting by considering the reconstruction of generic objects.

1.2 Problem statement

A capture platform records a dynamic scene using many cameras taking pictures *concurrently* (*i.e.* all at once). This is opposed to the the reconstruction

of inanimate objects, for which the pictures can be obtained *sequentially* using a single camera. Here, each camera records a video where the frames are precisely timed so that all the cameras stay synchronized. Applying a reconstruction algorithm at each time step gives a sequence of 3D reconstructions that can be played back to form a 4D reconstruction. Afterwards, the scene can be observed under any viewpoint, hence the denominations volumetric video, 4D video, light-field video [32] or volumetric capture that are commonly used (see [15] for online interactive examples). We consider that each time step is independent from the others, and leave temporally-consistent modeling as future work.

Thus, for each time step, we take as input a set of RGB images along with the calibration of the cameras (focal length, position and orientation among other parameters). We also rely on background images or binary masks to identify the pixels which contain useful information. From these inputs alone, we aim to recover a detailed triangle mesh, preferably water-tight. As a by-product of the reconstruction, we obtain the view-dependent color of every surface point in the form of a radiance field. The radiance field can be visualized as-is or it can be converted to vertex colors if needed.

We base our approach on differential rendering, which has lot of potential to improve over multi-view stereo for volumetric capture. There are however many challenges that must be addressed before it becomes an effective tool in our context.

1.3 Challenges

We now analyse the main difficulties that stand in the way of previous approaches, in particular when applied on volumetric capture data.

1.3.1 Accounting for the domain gap

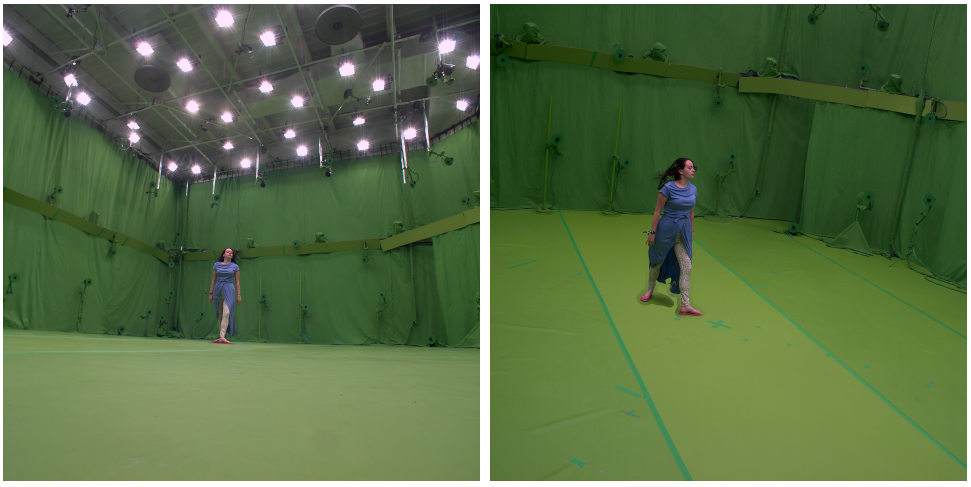

Figure 1.2: Typical images from the Kinovis capture platform. Note the wide field of views and small projected silhouette.

In theory, reconstructing a moving subject from many pictures taken at the same time is equivalent to reconstructing a static object from sequential pictures. This means that general-purpose photogrammetry software can be used but may not give the best results due to the domain gap. For instance, ActorsHQ [3] provides meshes reconstructed with RealityCapture [14], which works well on this instance but tends to fail on lower resolution input images (cf chapter 4). We identify several differences in the setting of our problem compared to mainstream 3D reconstruction:

- **Distance** and **Coverage**. In a capture platform, the cameras are spread on a hemisphere surrounding the stage so that the surface of interest is imaged with a uniform sampling rate, assuming cameras with similar focal lengths. In the angular domain, the surface is also observed from uniformly spread directions due to the arrangement of the cameras on a dome. Only a fraction of the pixels may be occupied, as shown in fig. 1.2. In comparison, multiview stereo generally expects images taken with a smaller baseline [33, 34], with a greater proportion of the pixels being useful and possibly many more images [35]. The viewpoints are spread more irregularly however, with a mix of close-up and far away pictures and no specific angular distribution.

- **Motion**. Reconstructing a moving subject can be challenging due to motion blur, as illustrated in fig. 1.3. The blur is anisotropic, stronger along the projected direction of motion. Consumer cameras may additionally display rolling shutter artifacts, which can be avoided using professional cameras [36] equipped with a global shutter. Consumer digital cameras record pixel values one scan-line after the other. This means that every row of pixels of a given frame is actually captured at a slightly different time, which introduces discrepancies in the observed position across viewpoints. Rolling shutter artifacts are thus highly undesirable. Unfortunately, quantitatively validating reconstruction algorithms under such motion artifacts is difficult since the ground truth shape is unknown.

- **Materials**. Typical surfaces encountered in outdoor scenes [37, 38] have fractal-like textures: stone or concrete facades, roads, tree foliage or bark, or mirror-like behavior: windows or car paint. Indoor scenes [39] often include reflective varnished furniture or metallic utensils, and even translucent glassware. In comparison, human scenes appear much easier to reconstruct due to milder reflections and simpler texture. Skin, hair and clothes are typically not thought of as being reflective in the usual sense, *e.g.* like a mirror, and yet they are far from being Lambertian materials either, which is especially the case at grazing angles. The general lack of texture (consider the blue dress or the arms in figure 1.3) and the low resolution is actually a major problem for stereo-based approaches. Planar assumptions [40] that would otherwise help handling featureless regions in indoor and outdoor scenes do not apply here. As we will show later in this document, multiview stereo algorithms [8, 25] tend to struggle with this type of content in spite of the favorable lighting conditions, camera placement and mild specularities. To a small extent,



Figure 1.3: Motion blur can be seen around the legs during a fast cartwheel motion.

translucent and shiny materials can be encountered in human scenes, in the form of glasses or thin strands of hair, and specular jewelry.

- **Scene contents**. Volumetric capture scenes are highly sparse compared to their indoor and outdoor counterparts. Sparsity can be found in image space, where only 10% to 20% of the pixels pertain to the silhouette (as seen in figure 1.2), and in 3D space, where the acquisition room is empty besides the person occupying it. Significant efficiency gains can be achieved by taking advantage of this sparsity, as will be demonstrated. We develop our algorithms assuming the shape of interest is located near the center of scene, *i.e.* at approximately the same distance to all the cameras, and that the surroundings can be discarded. The second assumption prevents us from reconstructing complete scenes, which require dedicated handling otherwise [38,41,42], and that would be wasteful here.

All these differences justify the investigation of differential rendering for volumetric capture, where previous algorithms assuming textured or flat surface patches usually struggle.

1.3.2 Reconciling geometric and photometric accuracy

Geometric and photometric quality are often considered separately in differential rendering works. Some exclusively focus on novel view synthesis [3, 5, 26–28, 43] while some predominantly study the geometric reconstruction problem [10, 29, 44, 45]. The first kind of approaches are capable of generating very convincing images but the recovered scene lacks geometric consistency: disorganized pieces of matter known as "floaters" can appear in mid-air to explain a single viewpoint, as long as it does not affect the others. Reflections can also be faked by mirroring part of the scene behind the surface or layering several pseudo-surfaces. The second kind of approaches introduces geometric

regularizations [10,46–50] to improve the quality of the recovered surfaces, but this can degrade the photometric quality [10,47].

We postulate that both geometric and photometric accuracy can be achieved at once by properly disentangling the geometric and photometric responsibilities in the scene representation: shading variations caused by a local change in surface normal should be fully explained by the geometry, rather than faked by the appearance model, which would naturally improve the surface quality. We also conjecture that encoding texture variations at a scale matching the pixel resolution would let the appearance model focus on the view-dependent effects, therefore improving photometric fidelity. Both of these ideas depend on a high-resolution modeling of the scene, which comes in opposition with typical grid-based approaches [30,51] advocating for low-resolution feature grids. Following this intuition, we will investigate how to model the scene at a millimetric scale and below.

1.3.3 Achieving scalability

We mainly focus on the reconstruction problem but also design our approach for scalability and parsimony, which are needed anyway to target higher resolutions than prior works. Three aspects are particularly relevant:

- **Processing time**. Multiview human datasets [3,9,52] can reach millions of frames for some, quickly leading to unrealistic computation times to process the data in its entirety. This manuscript investigates ideas with a cost / quality trade-off compatible with these datasets, considering that up to a few minutes of computation per frame and per GPU remains acceptable. Many approaches largely exceed this bound, some by several orders of magnitude, which makes them inadequate on long sequences.

- **Rendering speed**. Being able to visualize the reconstructed scene in real-time, as a 4D video, is one of the purposes of volumetric capture. We aim for real-time rendering out-of-the-box (chap. 4 & 5), since post-processing or distillation could heavily affect processing time [41,53–58]. Further optimizing the representation for mobile devices and streaming remains relevant [56–58]. Doing so efficiently for temporal sequences is left as future work.

- **Memory footprint**. The VRAM consumption at training time can be a bottleneck when targeting high-resolution datasets [3], even with high-end professional hardware [59]. The storage space requirements for the reconstructions can also be substantial [30,51] and while some approaches target a small model size [60–62], the training can be expensive (both in time and in VRAM). In contrast, we target a reasonable model size by default along with a contained VRAM usage, which consequently lets us reconstruct scenes at a finer resolution, sub-millimetric in fact (chap. 5).

1.4 Outline and Contributions

This thesis contributes an accurate and practical reconstruction pipeline for performance capture from many RGB cameras. We show that differential rendering can deliver accurate geometry extraction and photo-realistic novel view synthesis in this context, at a reasonable computational cost. Chapter 2 reviews related works regarding the topics of volumetric capture and differential rendering, where we present a fine-grained analysis of concurrent works and highlight efficiency and accuracy bottlenecks. Based on this study, we make the following contributions:

1.4.1 A parsimonious and scalable framework for hybrid volumetric / surface reconstruction

We show that incorporating sparsity and parsimony in the representation is a key component toward scalability. We pursue sparse representations of shape and efficient encodings of appearance in this manuscript, leading to unprecedented cost / quality trade-offs in the context of volumetric capture. Chapters 3 and 4 explore two sparse representations of shape, through an opacity field and a implicit surface parameterization respectively, while chapter 5 is focused on a minimal appearance model.

1.4.2 A disentangled representation of geometry and appearance

We demonstrate that a good photometric and geometric quality can be achieved at the same time by disentangling these two aspects in the scene representation. We propose a neural appearance model that is fully agnostic to position coordinates in chapter 4, such that the responsibilities pertaining to space (geometry and texture) are delegated to our high-resolution sparse grid whereas all the angular responsibilities are handled by a small neural network. This clear separation of tasks allows us to target high geometric and photometric precision while maintaining efficiency.

1.4.3 A local spatio-angular factorization of appearance

Chapter 5 contributes a novel appearance model, generalizing our approach to shiny materials. We decompose radiance into a set of low-resolution angular features and high-resolution spatial features, with a low dimensionality. The neural component becomes agnostic to both position and absolute orientation, only considering the angle of incidence and the aforementioned features. This decomposition is physically-inspired and thus provides a better supervision of the shape compared to typical neural appearance models.

1.4.4 A novel dataset for unbiased reconstruction benchmarking

We propose a dataset of clothed mannequins in chapter 4, imaged inside a capture platform and subsequently scanned at a high accuracy. This dataset joining real multi-view images and corresponding reference geometry allows for fair evaluation of reconstruction algorithms in the context of a multi-view capture platform. This is in contrast to popular benchmarks relying on either synthetic images or generic objects, far from the conditions of a capture platform.

Finally, chapter 6 concludes this thesis by offering a retrospective of our contributions and proposes future directions of research.

1.5 Publications

This thesis is based on the following publications:

- Chapter 3:

  Briac Toussaint, Maxime Genisson, Jean-Sébastien Franco. Fast Gradient Descent for Surface Capture Via Differentiable Rendering. *3DV 2022 - International Conference on 3D Vision*, Sep 2022, Prague, Czech Republic.

- Chapter 4:

  Briac Toussaint, Laurence Boissieux, Diego Thomas, Edmond Boyer, Jean-Sébastien Franco. Millimetric Human Surface Capture in Minutes. *SIGGRAPH Asia 2024 - 17th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia*, Dec 2024, Tokyo, Japan.

- Chapter 5:

  Briac Toussaint, Diego Thomas, Jean-Sébastien Franco. ProbeSDF: Light Field Probes For Neural Surface Reconstruction. *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), June 2025, Nashville, USA.

2

Background and Related Works

Capturing the shape and appearance of a person is a long-standing problem in computer vision and a wide range of methodologies have been developed for that purpose [2,63–65], drawing from the 3D reconstruction literature. Despite the use of complex and expensive capture setups, the results still sometimes lack in photo-realism or geometric accuracy. Interestingly, a new class of reconstruction algorithms based on the principle of differential rendering has recently emerged. Both appearance and shape are optimized jointly, while also taking into account the specularity of the surface. Thus, we seek to apply differential rendering to the volumetric capture task, with the simultaneous goals of outperforming previous approaches while maintaining practicality.

In this chapter, we first discuss previous works dealing with volumetric capture along with their limitations, before examining the concept of differential rendering. Next, we focus on a subclass of differential rendering approaches targeting high geometric quality using implicit surfaces, that we base our approach on. We will subsequently review a set of popular options for shape and appearance parametrization, to select the most adequate in later chapters.

2.1 Volumetric capture

Volumetric capture (or performance capture) aims to build a representation of a moving person from a set of pictures. Two groups of approaches can be distinguished based on the number of cameras. On one hand, multi-view performance capture [2, 4, 63–68] considers up to hundreds of images taken from distinct cameras at the same time, in a controlled environment. On the other hand, monocular performance capture [69–73] makes use of a single viewpoint per time step. Monocular approaches are much more affordable by construction since only one camera is needed but require strong priors such as a template mesh deformed over time [70, 71], whose motion is possibly driven by an articulated skeleton [20], or learned priors [73]. These priors reduce

the space of solutions and make the problem better posed but can also limit the accuracy by disallowing topological changes or complex motions. In this thesis, we target high-quality reconstruction from multiple viewpoints so our work pertains to the first group.



Figure 2.1: Capture setup including 53 RGB and 53 IR cameras at 4Mpx and unstructured IR laser light sources. Image reproduced from Collet et al. [2].



Figure 2.2: Capture setup (left) and corresponding novel view synthesis (right). Image excerpted from Işık et al. [3]'s HumanRF work. The capture platform contains 160 12Mpx RGB cameras.

A further distinction can be made between active and passive acquisition setups. Active setups may rely on infrared speckle projectors and infrared cameras [2,68], controllable (visible) lighting [64] or RGB-D cameras [4,65,67] such as the Kinect [74]. The underlying principle behind shape estimation from observations under known lighting directions is photometric stereo [75,76]. Binocular stereo [77] is the working principle for RGB-D cameras, with more recent devices using time-of-flight technology instead. In contrast, passive setups only rely on RGB cameras. Incorporating active devices helps removing ambiguities inherent to the reconstruction problem, that arise due to insufficient information in what RGB images can provide. There are however significant downsides to active approaches too.

Photometric stereo can be impractical since both the light source(s) position(s) and the material's response to light must be known in advance. A single light source is insufficient in general [75] so multiple light sources are needed.

Interference between the light sources can be avoided with a wavelength-based multiplexing [76] or time-slicing [64]. Neither options are desirable since the lighting is either directionally colored (interfering with the appearance) or flashing (likely to be uncomfortable for the actors). Lastly, the assumption of a known reflectance map makes it difficult to reconstruct several materials at once or specular materials for which the Lambertian assumption does not apply.



Figure 2.3: Real-time reconstructions with 3 RGBD sensors. Image excerpted from Yu et al. [4].

RGBD cameras seem better suited for the task. They can provide depth information along with color information, which allows for impressive real-time volumetric capture for telepresence applications [4, 65, 67]. The number of viewpoints can also be reduced since the depth is strongly related to the 3D shape [4]. Nonetheless, depth cameras suffer from a lower pixel resolution, limited range and higher noise level than RGB cameras. As a consequence, the recovered geometry may lack in fine details due to temporal and spatial averaging of the depth frames [78–80]. Zollhofer et al. [81] leverage the comparatively higher resolution of the rgb images and propose a shape-from-shading refinement to recover high-frequency geometric details. Their methodology is comparable in principle to the differentiable rendering approaches that have been developed later, but with major simplifications: the surface normal is optimized under the assumptions of a Lambertian material and of a low-frequency and far away illumination.

High-end capture setups leveraging infrared projectors [2, 68] achieve high-quality results but are costly since both infrared and color cameras are needed. The quality gap compared to RGB-only reconstruction is reducing [66] and this thesis pushes further in this direction. Entirely passive acquisition setups have potential to match or outperform active setups in some situations: mid-scale volumes allowing large motions over several meters, imaged by many cameras. In comparison, active setups or monocular approaches are highly relevant for applications constrained by a lower budget or strict real-time requirements such as telepresence. Here, we focus on high-quality offline reconstruction from many rgb images, with a different quality / processing speed trade-off.

The standard pipeline for 3D reconstruction from rgb images requires to calibrate the cameras [24, 82, 83] so that 3D points can be related to the 2D observations. Two options are common. In the case of in-the-wild reconstruction, structure-from-motion [25] produces a sparse point cloud along with the camera parameters, that can be used as an initialization of the shape [6, 27]. In the case of volumetric capture, the cameras are calibrated ahead of time [84]

so this point cloud is not directly available. However, silhouettes can easily be extracted from the images in this scenario, using chroma keying or background images. This allows to compute the visual hull [85], a decent initialization of the shape that becomes more accurate with more cameras [2,63,66]. The exterior of the visual hull is guaranteed not to contain the true shape, which drastically reduces the search space. Multiview stereo algorithms [25,66] can subsequently be used to refine the geometry based on a photoconsistency criterion, and a final meshing step is applied on the resulting dense pointcloud [86,87]. The appearance is usually obtained once the geometry is known with sufficient precision, using traditional texture mapping or more advanced image-based rendering techniques [39,88,89]. Photoconsistency is robust and some approaches based on differentiable rendering incorporate it as a regularization [46,50] or even use it in place of the photometric loss [44]. Although costly, it is also possible to use a dense pointcloud reconstructed by multiview stereo as an initialization [90].

This general pipeline is effective for most cases but becomes limiting in some situations. In particular, multiview stereo approaches struggle with highly specular materials, textureless regions in the images or fine structures [91]. In such cases, the shape may not be reconstructed in these locations which creates holes or missing parts in the pointcloud. Inaccuracies in the geometry also negatively impact the subsequent steps recovering the appearance. In contrast, differential rendering unifies shape reconstruction and appearance reconstruction in a single task so that both objectives participate. This becomes particularly useful when dealing with reflective materials. Weak shading cues in textureless regions can also be exploited with differential rendering, whereas photoconsistency often explicitly ignores changes in shading. Lastly, differential rendering models the scene at the pixel level, which makes it more likely to handle fine structures.

2.2 Differential rendering

Current bottlenecks hindering volumetric capture applications are complex and expensive acquisition setups, long processing times and imperfect realism. Differential rendering has potential to improve in all these aspects compared to traditional multiview stereo, or at least to complement it. The key idea of differential rendering is to combine a scene representation and a rendering algorithm that are both differentiable. The scene representation can be trained with gradient descent so that the renderings of the scene explain the ground truth images, for every available viewpoint. If the supervision is sufficiently dense, unseen viewpoints can be convincingly interpolated. This approach is two-in-one: both the shape and view-dependent appearance are recovered at once, simplifying the traditional reconstruction pipeline outlined in the previous section. However, a meshing step is still required in most cases if an explicit mesh is needed. The scene representation is composed of a shape representation and of an appearance model that are generally independent from each other, although some combinations are typically favored. Differential rendering methods can be categorized within the space generated by the Cartesian

product of these three sets:

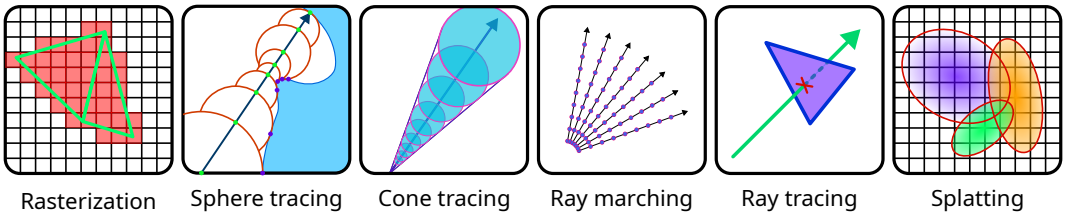

Figure 2.4: Common rendering algorithms used in differential rendering.

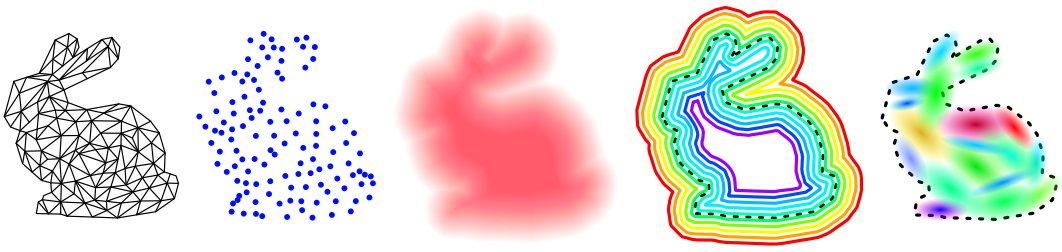

Figure 2.5: Popular shape representations.

- The set of rendering algorithms: mesh rasterization [92, 93], sphere tracing [45, 94], cone tracing [95], ray marching [10, 26, 96], ray tracing [97], splatting [27].
- The set of shape representations: triangle meshes [92, 93], point clouds [6, 90], opacity volumes [26, 96], implicit surfaces [10], Gaussians [27].
- The set of appearance models: spherical harmonics [27, 96], spherical Gaussians [98, 99], neural [10, 26], physically-based [93, 97, 98, 100].

These enumerations are not exhaustive and the number of possibilities is further amplified by hybrid approaches [101]. In this thesis, we opt for ray marching into an opacity volume (chapter 3) later upgraded to an implicit surface (chapter 4). Many other options are obviously possible but these were already shown to be viable choices early on [10, 26], with newer works exploring other combinations [27, 95, 97]. As of the writing of this document, the debate is not settled and new representations are still being proposed [42, 102]. In terms of the appearance model, we select spherical harmonic coefficients in chapter 3, replaced by neural features in chapter 4 for parameter parsimony. We additionally contribute a hybrid appearance model in chapter 5 for combined expressiveness and efficiency. A parametrization of the scene representation must also be chosen, as will be discussed in sec 2.4. For now, we give an overview of volume rendering in the following subsection, which is the basic tool used for the differential rendering of opacity volumes and implicit surfaces.

2.2.1 Differential volume rendering

Among the many rendering algorithms, volume rendering is particularly suitable for gradient-based optimization because of its smoothness. Other rendering algorithms can contain discontinuities that require special handling [92, 103, 104]. Representing the scene as a continuous volume avoids this issue entirely and also allows modeling both opaque surfaces (cloth, skin) and translucent media (hair) in a unified framework.

Light transport is typically approximated since the reconstruction problem becomes much more difficult if light sources and their interaction with the scene have to be modeled explicitly [100]. Instead, the opacity field is paired with a directionally-dependent color field to make a radiance field [26]. Every point of space is modeled as an emitter of radiance in every direction. A similar concept had been proposed before, known as light fields [105], lumigraphs [106] or the plenoptic function [107] for image-based rendering but more recent works popularized this formalism for 3D reconstruction and novel view synthesis [26, 43].

The optical model described by Max [108], now widespread for radiance fields rendering [26], is a formulation of volume rendering that takes into account absorption and emission but ignores scattering in its simplified form. This additional simplification is helpful because scattering is a complex phenomena that requires careful handling [109, 110]. We now briefly summarize the mathematical formulation of [108]. The extinction or attenuation coefficient σ describes how the intensity I of a light beam is reduced as it travels through a medium along a ray $\mathbf{x}(t) = \mathbf{o} + t\mathbf{d}$ with origin $\mathbf{o}$ and direction $\mathbf{d}$:

$$\frac{dI}{dt} = -\sigma I \iff I(t) = I_0 \exp\left(-\int_0^t \sigma(\mathbf{x}(s))ds\right) \tag{2.1}$$

In the general case, σ depends on the wavelength (σ_{rgb}) but is taken to be a scalar field here, relabeled as the opacity or density field in the differential rendering literature: $\sigma\colon \mathbb{R}^3 \to \mathbb{R}^+$. Large values of σ correspond to opaque areas whereas σ should be zero for transparent regions. The transmittance $T(t) \in [0, 1]$ describes the residual proportion of the light beam remaining after travelling a distance t:

$$T(t) = \exp\left(-\int_0^t \sigma(\mathbf{x}(s))ds\right) \tag{2.2}$$

This opacity field is combined with the radiance field $\mathbf{c}\colon \mathbb{R}^3 \times \mathbb{S}^2 \to [0, 1]^3$ that maps spatial and angular coordinates to RGB radiance values. Since the input images are generally encoded in the sRGB color space, the radiance field is clamped to the $[0, 1]$ range instead of proper radiance values defined on $\mathbb{R}+$ in a linear color space. The color of a particular camera ray is defined by the integral:

$$\mathbf{C}(\mathbf{o}, \mathbf{d}) = \int_0^\infty \mathbf{c}(\mathbf{x}(t), \mathbf{d})\sigma(\mathbf{x}(t))T(t)dt \tag{2.3}$$

The color is integrated in front-to-back order, starting at the position of the viewpoint $\mathbf{o}$ at $t = 0$ and stopping when the transmittance $T(t)$ becomes small

enough. This continuous integral is discretized by a simple quadrature rule:

$$T_i = \exp(-\sum_{j=0}^{i-1} \sigma_j \delta_j) \tag{2.4}$$

$$\mathbf{C}(\mathbf{o}, \mathbf{d}) = \sum_{i=0}^{N-1} T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i \tag{2.5}$$

With $\mathbf{c}_i = \mathbf{c}(\mathbf{x}(t_i), \mathbf{d})$, $\sigma_i = \sigma(\mathbf{x}(t_i))$, $\delta_i = t_{i+1} - t_i$ and t_i the discrete sample distances on the ray that are monotonously increasing. The spacing of the samples may be uniform, as in chapter 3, or follow an importance sampling scheme for improved accuracy, as in chapter 4. The rendered color $\mathbf{C}$ of the ray $\mathbf{r}$ emerging through a particular pixel of an input image can be compared to the ground truth color $\mathbf{C}_{\text{gt}}$ of that pixel. The photometric loss measures the error for a set of pixels and corresponding rays $\mathcal{R}$:

$$\mathcal{L}_{\text{photometric}} = \sum_{\mathbf{r} \in \mathcal{R}} ||\mathbf{C} - \mathbf{C}_{\text{gt}}|| \tag{2.6}$$

Several types of photometric losses can be used such as the simple L2 [26] or L1-like [111] losses, or more complex perceptual losses [27]. Equation 2.5 is clearly differentiable with respect to σ_i and $\mathbf{c}_i$ so that partial derivatives of the photometric loss can be propagated to the radiance and opacity fields with the chain rule. These derivatives are further propagated to the actual parametrization for the scene representation.

2.2.2 Gaussian Splatting

Gaussian splatting [27] is a reformulation of volume rendering where the continuous distribution of densities is replaced by a discrete set of semi-transparent elements, generalizing sphere splatting [112] with anisotropic Gaussians [113]. This representation yields high-performance and high-quality rendering at the same time and has largely superseded neural radiance fields for novel view synthesis, in part due to the convenience of a reduced training time.

A stream of recent works leverage this representation for the volumetric capture of human faces [99, 114–117], achieving real-time rendering, relighting and animatable expressions with photo-realistic quality. Geometric accuracy is not targeted by these works whereas novel view synthesis is the primary goal. Furthermore, the discrete nature of the 3D Gaussians makes it difficult to obtain a mesh with conventional isosurface extraction algorithms. In contrast, we want to capture both the shape and appearance of complete humans which is why we turn our attention to another branch of Gaussian splatting approaches aiming for accurate shape reconstruction [47–50]. These approaches are designed for general 3D reconstruction which naturally encompasses volumetric capture and makes them compatible with our goals.

Guédon et al. [47] (SuGaR) propose to render depth maps as the weighted average of the Gaussians' depth and derive a regularization based on it. A mesh is obtained by sampling points on the depth maps and applying Poisson surface reconstruction [86]. Huang et al. [48] (2DGS) employ 2D Gaussians

instead of 3D Gaussians to describe small surface patches. This enables to explicitly compute ray / splat intersections and improves the accuracy of the depth estimation away from the center point of the splats. In contrast, Yu et al. [49] (GOF) define an opacity field everywhere in space from the discrete set of 3D Gaussians. A mesh is obtained with the marching tetrahedra algorithm [118] with a binary search on the edges of the tetrahedra to accurately locate the level set. Chen et al. [50] (PGSR) adopt MVS priors with the normalized cross-correlation computed on pixel patches, similarly to [46], along with an unbiased depth formulation. These recent works are particularly strong contenders in terms of geometric quality on unbounded scenes and object reconstruction. PGSR [50] even matches the quality of the best neural implicit surface reconstruction methods [29, 46] and is faster. We evaluate these methods in chapters 3 and 4 on human reconstruction and identify some of their shortcomings in this particular context. Importantly, their training time still ranges from 10 minutes to one hour or more which is penalizing for temporal sequences. Other approaches specifically designed for 4D novel view synthesis [119–121] currently omit surface reconstruction.

2.3 Differential rendering for implicit surfaces

Approaches based on opacity fields achieve outstanding photometric quality [28, 38] but often do not produce high-quality surfaces at the time of this writing. Regions with uniform colors are ambiguous since they can be explained by a cloud-like distribution of opacities, without a sharp interior and exterior boundary. Extracting a mesh by thresholding the density field and applying marching cubes [11] leads to holes and foam-like structures inside the opaque regions since the interior is unsupervised [5]. Geometric accuracy is important for many downstream applications of volumetric capture and thus a surface-aware strategy is necessary. In parallel to the novel view synthesis research, another branch of works have applied differential rendering on implicit surfaces which has proven to be promising for our problem. We now discuss these works and the challenges they pose to make them compatible with surface capture.

2.3.1 Differential surface rendering

Volume rendering combines contributions from a potentially large number of points along the ray which implies a substantial cost. In comparison, surface rendering only considers a single point taken as the first intersection of the ray with the shape. Niemeyer at al. [122] (DVR) define the surface as $\{\mathbf{x}, f_\theta(\mathbf{x}) = \tau\}$, with $f_\theta \colon \mathbb{R}^3 \to [0, 1]$ an occupancy network [123]. The intersection point is obtained with a root finding algorithm which still requires sampling many times along the ray, but gradients only need to be propagated at the intersection point rather that at all the query points. Liu et al. [94] replace the occupancy network with a DeepSDF network [124] and use the sphere tracing algorithm [125] to quickly find the first intersection point. Yariv et al. [45] (IDR) also use the sphere tracing algorithm but the signed distance property has to be learned,

instead of being pre-trained on an object database. The Eikonal constraint is typical to penalize the deviations of a level set f from a true signed distance function [126]:

$$\mathcal{L}_{\text{Eik}} = (||\nabla f|| - 1)^2 \tag{2.7}$$

This regularization can be applied densely in the scene or at a sparse set of randomly sampled points, as is common for neural implicit surfaces. Mesh-based rasterization [92] and ray-tracing [97] are also viable alternatives to sphere-tracing, notwithstanding an underlying neural implicit parametrization of the mesh in the case of [97].

Surface rendering offers an advantage compared to volume rendering in two aspects. First, in terms of performance as only a single point contributes to the color, and second, in terms of surface quality. Since there is a well-defined surface, it can be regularized for smoothness and normal vectors can link the local orientation to the appearance [45]. The normal vectors and curvature information are not readily available in the case of an opacity field. A mesh can also be consistently extracted from a level-set whereas a single opacity threshold may give unsatisfactory results. However, naive surface rendering lacks differentiability at occlusion boundaries which penalizes convergence and requires anti-aliasing [92], edge sampling [104] or supervision from silhouette masks [45]. Furthermore, carving holes is more difficult since only one depth hypothesis is considered at a time. In contrast, volume rendering can take advantage of point samples located on both sides of the real surface to guide the optimization. It becomes apparent that surface and volume rendering are complementary and in fact can be unified in hybrid frameworks that we now discuss.

2.3.2 Hybrid rendering

A number of methods combine volume rendering and surface rendering in order to benefit from the strengths of both, with the main goal being accurate surface reconstruction. Oechsle et al. [127] (UNISURF) distribute a few samples in an interval centered at the first intersection with the surface. This idea successfully removes the reliance on masks for reconstruction but is still not fully differentiable at occlusion boundaries since the depth is discontinuous. In contrast, Yariv et al. [128] (VolSDF) and Wang et al. [10] (NeuS) propose to link the density field σ to an underlying signed distance field f, evaluated along the ray $\mathbf{x}(t) = \mathbf{o} + t\mathbf{d}$.

Let $\Psi_\beta(x)$ and $\Phi_s(x)$ be two sigmoid functions such that:

$$\Psi_\beta(x) = \begin{cases} \frac{1}{2}\exp(\frac{x}{\beta}) & \text{if } x \leq 0 \\ 1 - \frac{1}{2}\exp(-\frac{x}{\beta}) & \text{if } x > 0 \end{cases} \tag{2.8}$$

$$\Phi_s(x) = \frac{1}{1+\exp(-sx)} \tag{2.9}$$

$\Psi_\beta(x)$ is the cumulative distribution function (CDF) of the Laplace distribution and $\Phi_s(x)$ is the CDF of the logistic distribution. The conversion from

the SDF f to an opacity σ is defined as follows:

$$\sigma_{\text{volsdf}}(\mathbf{x}) = \alpha \Psi_\beta(-f(\mathbf{x})) \tag{2.10}$$

$$\sigma_{\text{neus}}(\mathbf{x}) = \max\left(\frac{-\frac{d}{dt}\Phi_s(f(\mathbf{x}(t)))}{\Phi_s(f(\mathbf{x}(t)))}, 0\right) \tag{2.11}$$

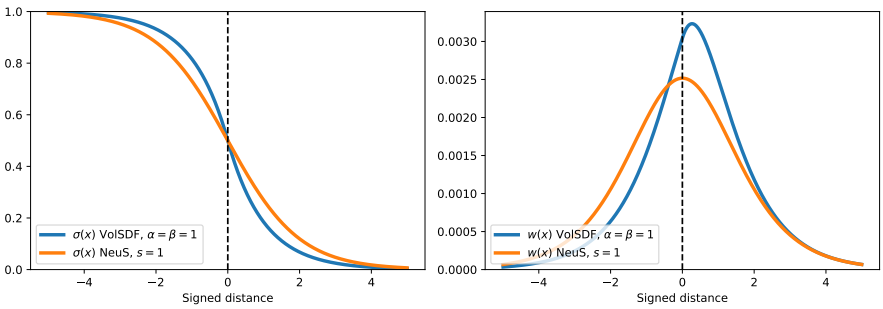

Figure 2.6: Left: plots of equations 2.10 and 2.11 as a function of the signed distance, for a ray perpendicular to the surface. Right: corresponding volume integration weights $w_i = (1 - \exp(-\sigma_i \delta_i)) \exp(-\sum_{j=0}^{i-1} \sigma_j \delta_j)$.

The sharpness of the transition from transparent to opaque is controlled by α and β for VolSDF and by s alone for NeuS. These parameters can be either scheduled or optimized during training to start with volume rendering and finish with surface rendering in the limit. As shown in fig. 2.6, σ_{neus} has the property that the zero level-set of the SDF coincides with the maximum volume integration weight whereas σ_{volsdf} does not. Nevertheless, VolSDF also achieves good results as the offset of the peak vanishes in the limit of $\beta \to 0$.

This methodology has achieved unprecedented surface reconstruction quality on multi-view object benchmarks [29, 31, 44, 46] but is often penalized by long training times and limited scalability to high-resolution input images. We demonstrate how to remold it for volumetric capture in chapters 4 and 5, lifting these two limitations and achieving unprecedented levels of detail in this context. We emphasise that obtaining high-quality results while simultaneously accelerating training, inference, and reducing memory usage is highly challenging, which calls for innovations going beyond mere software optimizations. The next sections take a deeper look in the inner-workings of popular differential rendering methods in order to identify their bottlenecks, first considering how they are parameterized (§2.4) and then how view-dependant appearance is modeled (§2.5).

2.4 Parameterizations

The parameterization of the radiance field plays an essential role in the performance and scalability of the differential rendering algorithms proposed in the literature. Typical implementations rely on large neural networks for both the geometry and appearance [10, 26, 29, 38, 43, 44, 46, 111, 128]. This is convenient from the programmer's perspective but is impractical in the context

of volumetric capture, since each reconstruction would take several hours at best. Many types of parameterizations have been proposed to circumvent this limitation and we review the most promising ones in this section. We take particular interest in the sparsity and efficiency so that good geometric quality and real-time rendering become possible.

2.4.1 Deep multi-layer perceptrons

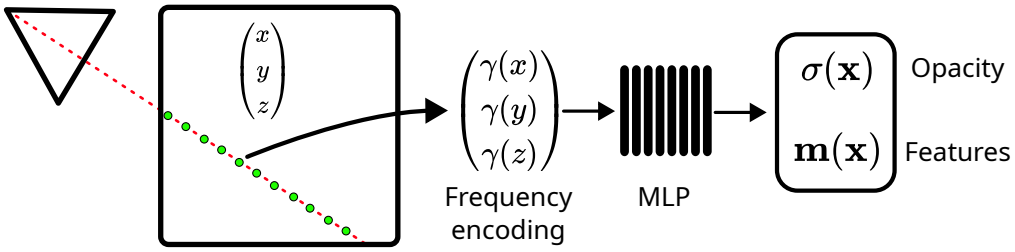

Figure 2.7: Deep MLP parametrization

Encoding the scene in the weights of a deep MLP is appealing due to deep learning libraries providing ready-made building blocks and automatic differentiation. Coordinate MLPs take as input scene coordinates and angular coordinates (hence the naming), pre-processed by a frequency encoding $\gamma(x)$. This encoding is a mapping from $\mathbb{R}$ to a high-dimensional vector and splits the input into components of increasing frequencies. Explicitly decomposing the inputs into frequency components has been shown to enhance the network's ability to learn high frequency signals [26, 129], in a simple manner. Several choices are possible such as trigonometric functions [26] in eq. 2.12, saw-tooth functions [130] or real spherical harmonic polynomials that are sometimes used for the angular inputs [5, 131].

$$\gamma(x) = (\sin(2^0\pi x), \cos(2^0\pi x), \ldots, \ldots \sin(2^{L-1}\pi x), \cos(2^{L-1}\pi x)) \tag{2.12}$$

NeRF [26] uses 8 fully-connected layers with 256 neurons for the density MLP, that outputs a density value σ and a feature vector $\mathbf{m} \in \mathbb{R}^{256}$, based on the positional encoding $\gamma(\mathbf{x}) \in \mathbb{R}^{60}$ of the sample point $\mathbf{x}$. The feature vector $\mathbf{m}$ is concatenated with $\gamma(\mathbf{d}) \in \mathbb{R}^{24}$ and then passed to the color prediction MLP consisting of a single layer of 128 neurons. This architecture has been reused in many following works [10, 45, 111, 128] with some adjustments on the specifics such as the number of layers, neurons, activation functions or types of positional encoding.

Besides the implementation convenience, MLPs are also very lightweight in terms of parameter count compared to other parametrizations. For instance, a model size of 5MB is often reported for [26]. There are however significant disadvantages such as a slow training and a large memory footprint. In particular, the large memory consumption imposes small batch sizes which worsens the training time, taking up to several days in some cases [132]. Rendering a single image at a low resolution takes several seconds, two to three orders of magnitude away from real-time rendering.

Most of this cost can be attributed to the large number of floating-point operations inherent to dense matrix multiplication. The memory bandwidth costs associated to the reads and writes of the weights and activations are also a concern, although they are not predominant for large matrices compared to the arithmetic costs. Note that the opposite becomes true for smaller neural networks. For this reason, we avoid transfers of intermediary results to global memory by fusing as many operations as possible in the same GPU kernel, as will be discussed in detail in chapter 4. This fully-fused strategy is gaining traction for real-time neural inference [5, 133–136] but is also useful for accelerating training.

The other parameterizations that we discuss next have a more advantageous trade-off between model size and arithmetic complexity than a monolithic MLP but are more complex to implement.

2.4.2 Dense grids

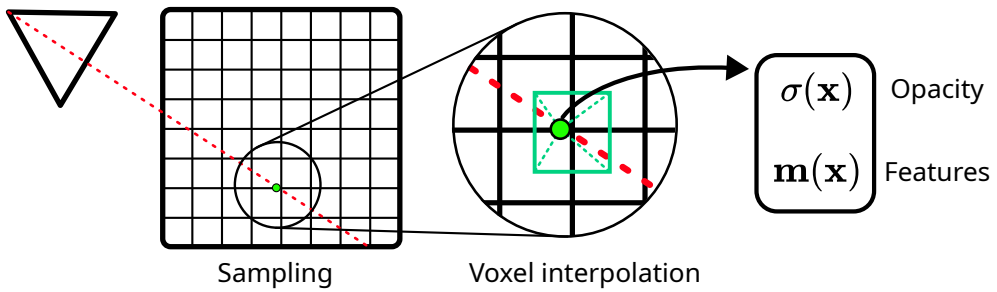

Figure 2.8: Dense grid parametrization

Dense grids offer cheap $O(1)$ random-access lookups which makes then popular for volume rendering applications [108]. However, directly tabulating the radiance field into a 5D array would be very memory intensive so a shallower neural network is often retained to produce view-dependent colors from localized features embedded in a 3D grid (hence a 4D tensor).

Sun et al. [51] (DVGO) encode density values $\sigma \in \mathbb{R}$ and features $\mathbf{f} \in \mathbb{R}^{12}$ in a 160^3 grid for novel view synthesis. The density is remapped by a softplus activation after interpolation at the sample points, to allow for sharp opacity variations within a voxel. An equivalent idea is explored by Karnewar et al. [137] with a ReLU activation instead. This non-linear remapping allows to encode geometric details at a higher resolution than that of the grid.

Wu et al. [30] (Voxurf) apply this strategy for implicit surface reconstruction. The density is replaced by a signed distance, converted to opacities by the rendering model of NeuS [10]. The resolution of the grid is slightly higher at 256^3 but with lower dimensional features $\mathbf{f} \in \mathbb{R}^6$ than for DVGO. Both approaches improve the training speed to 20 minutes and the rendering speed to $\sim$ 1Hz compared to NeRF and NeuS, with equivalent or better quality. However, the number of parameters is much greater, exceeding 100 million parameters for Voxurf, compared to less than 1 million for NeRF.

Memory is clearly an issue for these approaches which restricts the grid resolution. A smaller but still costly MLP is needed to predict higher fre-

quencies than what the discretization of the grid allows, based on interpolated features. This methodology imposes to predict the color at each sample rather than at each voxel. We show in chapter 4 that increasing the resolution of the discretization and exchanging the order of interpolation and color prediction unlocks new opportunities and leads to close to an order of magnitude faster training and more than two orders of magnitude faster rendering.

Unlike deep MLPs, grid-based approaches require smoothness regularizations for convergence. Thanks to the frequency embedding γ, deep MLPs can learn all the frequencies at the same time whereas grids only expose a single one. Here, regularizations are a way to bias the training toward lower spatial frequencies. Coarse-to-fine training is also a widely-known principle to discover lower frequencies first and has been used across-the-board in computer vision [138–141]. Closer to our work, both DVGO and Voxurf start at a lower resolution before upscaling the grid in a second stage.

2.4.3 Factorized grids

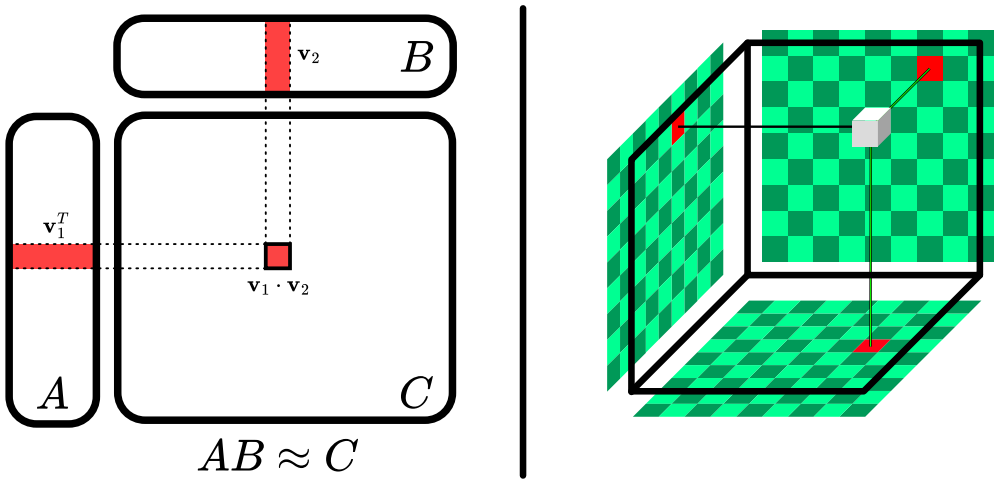

Figure 2.9: Left: Factorization of a matrix. Right: Tri-planar factorization.

Dense grids are penalized by a cubic memory footprint which makes it difficult to scale to high resolutions. For instance, there would be 8 billion voxels in a 2m cube discretized with millimetric voxels. This issue is conveniently resolved by a tensor factorization into lower rank components, with several options available. Much like a matrix that can be approximated by a sum of vector outer products, a 3D grid can be approximated by the outer product of vectors and / or matrices combinations. The values of the voxels are not known in advance so the factorization should be understood as a re-parametrization of a dense volume rather than a post-processing step.

Notably, Chen and Xu et al. [60] (TensoRF) propose two decompositions based on vector / matrix outer products or just vector outer products. The space complexity is reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$ respectively. In particular, the vector / matrix decomposition outperforms DVGO in quality with 70% less parameters, with a similar training time. A related parametrization is the tri-planes factorization, generalized by the k-planes factorization for

4D grids and beyond [142, 143] (K-planes, Hexplane). The tri-planes factorization is very similar to the vector / matrix decomposition of TensoRF but simpler. The 3D point $\mathbf{x} = (x, y, z)$ is projected onto 3 orthogonal planes of features $\mathbf{P}_{xy}, \mathbf{P}_{xz}, \mathbf{P}_{yz}$. The planes are bi-linearly interpolated at the projected coordinate, followed by a component-wise multiplication in eq. 2.16. This operation can be repeated at multiple scales for improved quality [142, 144].

$$\mathbf{f}_z = lerp(\mathbf{P}_{xy}, xy) \tag{2.13}$$
$$\mathbf{f}_y = lerp(\mathbf{P}_{xz}, xz) \tag{2.14}$$
$$\mathbf{f}_x = lerp(\mathbf{P}_{yz}, yz) \tag{2.15}$$
$$\mathbf{f} = \mathbf{f}_x \odot \mathbf{f}_y \odot \mathbf{f}_z \tag{2.16}$$

Importantly, each of the components (vectors, matrices) are themselves dense grids of lower dimensionality. This means that the methodology still fits in standard deep learning libraries, without requiring custom GPU code, and is thus simple to implement. However, several important downsides of these factorization techniques should be mentioned. First, the quality is dependent on the alignment of the scene with the direction of the planes. This can be addressed by learning rotation matrices for the planes along with the rest of the parameters, as shown by [145]. Second, the space complexity reduction from $O(n^3)$ to $O(n^2)$ is somewhat misleading since the constants are hidden in the big O notation: the sparsity gains are in part counteracted by a larger feature dimensionality and thus also by a costlier interpolation and color decoding.

A better strategy is to combine the factorization with a sparse representation of the scene. Gao and Xu [146] (Strivec) apply the vector outer product decomposition proposed by TensoRF on cubic cells centered on each point of a sparse point cloud. Only the cells near the query point contribute their features, thus the decomposition is local rather than global. This allows to further reduce the number of parameters but still keep low dimensional features. A sparse factorization is also less sensitive to the orientation of the scene [146]. In chapters 4 and 5, we apply the tri-plane decomposition on tiles of 16^3 voxels, where the tiles are sparsely allocated in a coarse grid. We show that as little as 4 coefficients per feature are enough in chapter 5.

2.4.4 Sparse representations

The approaches mentioned previously favor convenience of implementation over speed in most cases, by representing the scene with deep neural networks or contiguous tensors of features. However, most scenes are highly sparse and could be encoded with similarly sparse representations, with possible gains in quality and performance. Implementing such data structures requires GPU code and significant debugging efforts. The recent emergence of open-source libraries handling some of these functionalities [147, 148] could be helpful for future works but the current and previous strategies heavily rely on custom hand-crafted solutions.

Hashmaps. Mueller et al. [5] (InstantNGP) proposed to represent the scene with a hierarchy of hashmaps. Instead of storing voxels directly and handling hash collisions as in [149], Mueller et al. [5] store low-dimensional

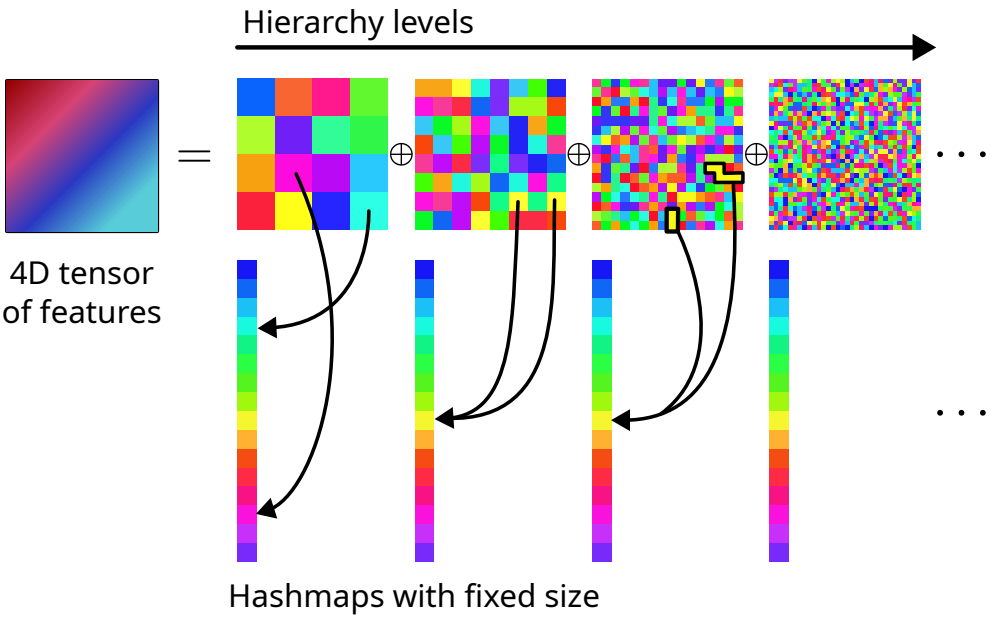

Figure 2.10: InstantNGP [5] decomposes the scene as a hierarchy of hashmaps. Collisions become more frequent in deeper levels which results in high-frequency noise. The feature $\mathbf{m}(x)$ *is defined as the concatenation of all the levels.*

features and explicitly map colliding hashes to the same entry. This is similar to the factorization techniques discussed in the previous section, which map all voxels along one axis to the same texel on a plane with an orthogonal projection. In this case however, the mapping from a 3D location to a particular hashmap entry is governed by the hash function that has no geometric structure. This work demonstrated exceptionally fast convergence with 15 seconds being sufficient to outperform NeRF. Part of this efficiency can be attributed to the fully-fused MLP implementation [133,134] and to the hashing strategy itself, both implemented in CUDA. This approach has been extended to implicit surface reconstruction by Wang et al. [31] (NeuS2). We identify two main limitations for this parametrization: the first one is the collisions in the hashmaps that create high-frequency noise in the resulting images and extracted geometry. The second one is the rendering performance that is less than 10Hz, not enough to be qualified as real-time. We evaluate against NeuS2 in chapters 4 and 5 with clear improvements in both aspects. Overall, hashgrid encodings inherit the same weaknesses as the factorization methods: fetching and interpolating many features becomes costly when combined with volume rendering.

Sparse grids. Sparse grids encode the same information as a dense grid but only near the surface. Typical implementations rely on coarse-to-fine learning with periodic pruning to discover the structure of the scene during training [96, 150], while some require a pre-trained scene that is then distilled into a more efficient representation [41, 53, 54]. If only pruning is performed, some geometric details may be missed in the coarse resolutions and cannot be recovered afterward. We avoid this issue in chapter 4 by alternating pruning and growing after every training iteration, guided by the signed distance val-

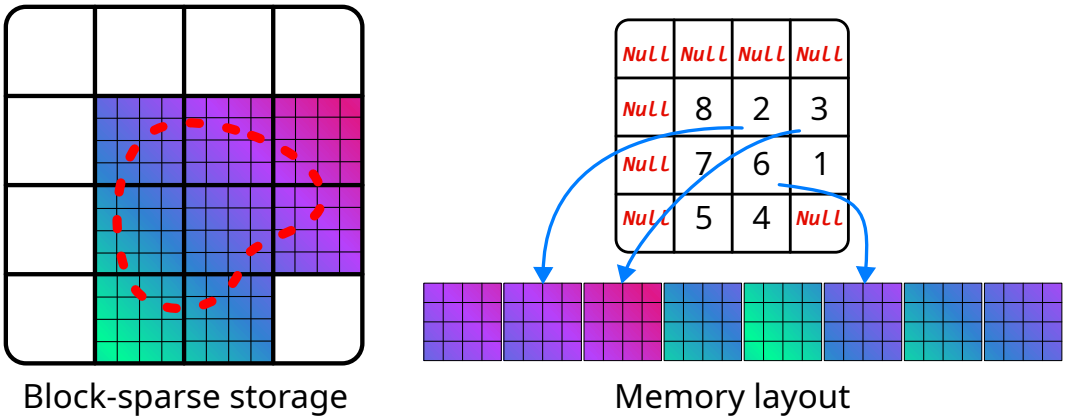

Figure 2.11: Block-sparse storage with 4^3 blocks and one indirection.

ues. Grid cells sufficiently far away are discarded so that the space complexity approaches $\mathcal{O}(n^2)$.

Yu et al. [96] (Plenoxels) use a dense pointer grid where only the voxels with valid pointers need to be allocated. The complexity is still $\mathcal{O}(n^3)$ but the constant is lowered since the size of a pointer is much smaller than that of a voxel's parametrization. At the other end of the spectrum, Liu et al. [150] (NSVF) obtain a better asymptotic space complexity with a sparse voxel octree [151] but the training and rendering speeds are affected by an MLP comparable in size to NeRF's. Hedman et al. [54] (SNeRG) employ a block-sparse representation where each pointer indexes a block of 32^3 voxels. There is a trade-off between the size of the pointer grid and the size of the blocks, where bigger blocks reduce the size of the pointer grid but also waste some space as the blocks themselves will contain empty regions. We propose to use a similar data structure (chapters 3, 4, 5) but with 4^3 voxel blocks to better exploit the sparsity of volumetric capture scenes compared to larger outdoor scenes. This concept is further generalized by N^3-trees [152, 153], where each block may have a different size depending on its level in the hierarchy.

2.4.5 Unstructured representations

Another branch of works explored unstructured representations, which are more malleable than rigid voxel discretizations, and thus potentially faster and lighter. In particular, Kerbl et al. [27]'s 3D Gaussians have arbitrary positions, orientations and scales, making them ideal for representing thin structures encountered in natural scenes such as blades of grass and leaves. Other works leverage point cloud representations to store neural features such as Xu et al. [6] (Point-NeRF) for novel-view synthesis and Chen et al. [90] (Fast Dipole Sums) for 3D reconstruction. Delaunay triangulations [7] and centroidal Voronoi tessellations [154] are also promising parametrizations due to the adaptive size of the cells.

Interpolating and regularizing an unstructured representation is not simple and may require acceleration structures such as kD-trees or bounding volume hierarchies for efficient neighbor queries [154]. There may be important costs associated to the recursive lookups and data structure updates, similarly to

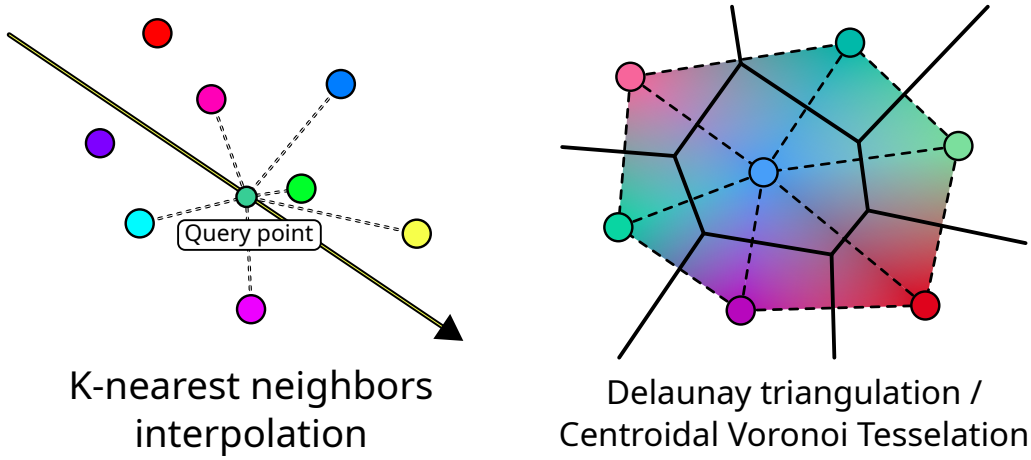

Figure 2.12: Left: PointNeRF [6]-style Knn interpolation. Right: Interpolation via the Delaunay triangulation of the point cloud as in [7].

sparse grids. In the particular case of 3D Gaussian splatting and its followup works [47–50], there in no space partitioning scheme so depth maps and normal maps are leveraged to regularize the shape through its 2D projections. Consequently, the extracted surface is not watertight since unobserved regions cannot be regularized.

Unstructured representations are successful for scene-level reconstructions but sparse voxel grids have interesting characteristics that make them highly suitable for reconstructing smooth and continuous surfaces. We evaluate against several Gaussian splatting baselines on object and human scenes in chapters 4 and 5, and find that our work handles high-resolution images and high-frequency texture patterns better in this context.

2.5 Appearance models

Strong view-dependent effects make the reconstruction problem ill-posed and difficult to solve. How to take these effects into account is a central topic for multi-view stereo and differential rendering approaches. This is especially important for the latter since the choice of appearance model impacts both the geometric and photometric quality.

Multi-view stereo approaches often make the assumption that the surface is Lambertian, or more generally allow for affine variations (NCC) of the observed color [155]. Differential rendering goes beyond by allowing more general changes using an appearance model, a mathematical description of the outgoing radiance. We analyse two popular options in this section.

2.5.1 Spherical harmonics

Spherical harmonics are a versatile mathematical tool, previously used for modeling diffuse shading [156], low frequency lighting [157] or scattering [158] in computer graphics. Kerbl et al. [27] and Yu et al. [96] adopt this representation for the outgoing radiance of every Gaussian or voxel, as written in

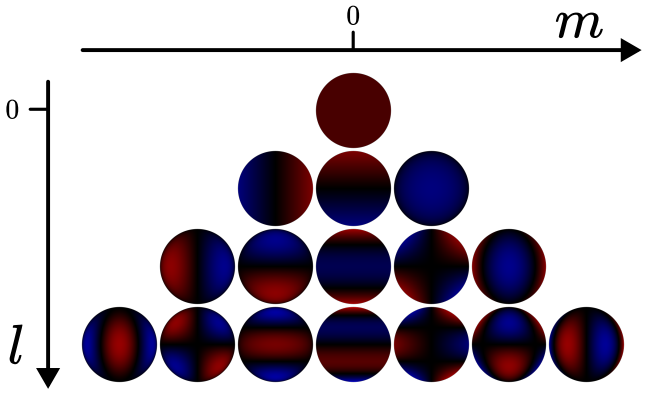

Figure 2.13: The first 16 Y_l^m polynomials visualised on the unit sphere. Red (resp. blue) colors denotes positive (resp. negative) values.

equation 2.17.

$$\mathbf{L}_o(\theta,\phi) = \sum_{l=0}^{L}\sum_{m=-l}^{l} \beta_l^m Y_l^m(\theta,\phi) \tag{2.17}$$

The outgoing radiance at any point $\mathbf{x}$ is given by $\mathbf{L}_o : \mathcal{S}^2 \mapsto [0,1]^3$ that maps view direction angles to RGB colors. The spherical harmonic (SH) functions Y_l^m are polynomials of degree l, we refer to [159] for mathematical details.

The main advantages of the spherical harmonics basis are its smoothness and differentiability as well as its generality: any angular distribution can be encoded as long as it does not contain frequencies beyond the truncation order L. The computational cost is typically lower than with neural networks, which makes it appealing for volumetric capture so we use this appearance model in chapter 3. However, the memory cost becomes penalizing at high resolutions, even with a low order basis, so we explore alternative appearance models in chapters 4 and 5.

2.5.2 Neural networks

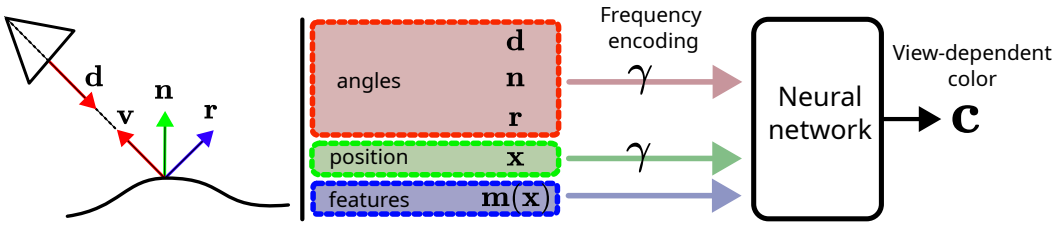

Figure 2.14: Generalized neural appearance parametrization.

Multi-layer perceptron are the standard appearance model for novel view synthesis [26, 38, 131, 132] and implicit surface reconstruction [10, 29, 45, 128]. Many accelerated variants also rely on MLPs but with much smaller architectures for training or inference speedups [5, 28, 30, 31, 41, 51, 54, 55, 60]. There are a lot of variations in how the color is derived from position and angular inputs so we only review the most important ones.

Mildenhall et al. [26] 's color MLP predicts colors based on the frequency-encoded view vector $\gamma(\mathbf{d})$ and a feature vector $\mathbf{m}$ derived from the density MLP. Similarly to the SH basis described above, this appearance model does not depend on the orientation of the observer relative to the surface but only on its absolute orientation.

In comparison, implicit surface reconstruction approaches include the normal vector along with the view vector [10, 29–31, 128], which has been shown to provide an implicit geometric regularization by Yariv et al. [45]. The reasoning is that correctly orienting the normal vector produces more accurate specularities by making the task easier for the color MLP, hence the geometry is improved through back-propagation of the rendering.

Mueller et al. [5], followed by Li et al. [29] and Wang et al. [31], replace the Fourier encoding γ of the view direction with spherical harmonics Y_l^m as it is a more natural frequency encoding in $\mathcal{S}^2$. An interesting property of the SH polynomials is that they are closed under rotations [159]: the embedding of a rotated vector is a linear combination of the embedding of the original vector. This is not the case with the Fourier embedding. Thus, rotating the scene does not affect the expressiveness of the color prediction.

Verbin et al. [131] propose to apply the spherical harmonics embedding on the reflected vector $\mathbf{r} = \mathbf{d} - 2(\mathbf{d} \cdot \mathbf{n})\mathbf{n}$ rather than on the view vector $\mathbf{d}$. The angular variations of the observed radiance depend on the lighting and on the curvature of the surface: conditioning on the reflected vector effectively separates these two effects and simplifies the task of the neural network.

In chapter 4, we use a two-layer MLP with 32 neurons conditioned on the normal, the spherical harmonics embedding of the reflected vector and an 8-dimensional feature vector fetched from our sparse grid. Surprisingly, this very compact parametrization is sufficient for accurate reconstruction of human subjects in a capture platform. The number of parameters is significantly reduced compared to a spherical harmonic basis and the small neural network unlocks real-time rendering for neural parameterizations. This appearance model favors low-frequency lighting due to the compact representation which makes sense for human volumetric capture. We propose a novel appearance model in chapter 5, in order to handle more varied materials and illumination conditions.

2.6 Takeaways

Extensive work has been done to improve various aspects of the differential rendering methodology: improved novel view synthesis quality [27, 28, 38, 111], improved geometric accuracy [10, 29, 44, 46, 50, 128], faster training [5, 27, 30, 31, 51, 96], real-time rendering [27, 41, 54–58], smaller representations [60–62], to name a few. Unfortunately, these improvements often come with significant trade-offs. Striking examples come to mind: Li et al. [29]'s representation requires 16 hours of training while Reiser et al. [58]'s distillation takes 60 hours using 8 GPUs, recovering highly accurate surfaces and achieving high-quality novel view synthesis on mobile devices respectively. Although impressive, most of these approaches may not scale to massive datasets given their compute cost,

orders of magnitude greater than what will be proposed later in this document.

A key takeaway of §2.4 is that inefficiencies can often be attributed to a large neural component [29, 30, 150]. Explicit representations generally fare better during inference [7, 27, 42, 58] but may still require significant training time. We need a well-rounded approach, since maximizing one characteristic at the expense of the others would be impractical. Our strategy is thus to reconcile these conflicting characteristics, first tackling the issue of a long training time in chapter 3, then addressing concerns about the quality of the recovered geometry in chapter 4 and finally expanding our approach to a more general setting in chapter 5, where we show that it is possible to reconstruct shinier materials and intricate objects without compromising on previous goals.

3

Fast Gradient Descent for Surface Capture Via Differentiable Rendering

3.1 Introduction

As detailed in the introductory chapter (see chapter 1), we target high-quality geometric and photometric reconstruction for surface capture using differential rendering. Computational efficiency is an additional key objective in this context, necessary to process long temporal sequences or large-scale datasets. We tackle this issue first, since we identify it as the main bottleneck of current works. In particular, this chapter examines how the accuracy and parsimony goals can be addressed within the family of volume-based differential rendering approaches. Most of the ideas developed here also apply to the volume/surface hybrid family considered in subsequent chapters. This study will also remain relevant for the more general setting of object reconstruction, later discussed in chapter 5.

Thus, we examine how differential rendering algorithms can be applied and improved for practical setups such as multi-view surface capture of human subjects, an inherently challenging task due to the complexities of motion, clothing, hair, self-occlusion, and other corrupting factors. As discussed in section 1.1, this task is of broad interest for all applications requiring 3D content that reflects and digitizes the real world, such as 3D broadcasting, entertainment, virtual reality, serious games, virtual-try on, and is gaining new relevance in light of the push toward the metaverse in the industry. Producing this content in controlled environments with many cameras is also vastly useful to build training sets for data-driven methods addressing the same problems with less views [160, 161], or even monocular input [162, 163].

Differential rendering methods have gathered significant attention from the research community, as a new angle to achieve high fidelity multi-view reconstruction. This makes them particularly palatable candidates to address the

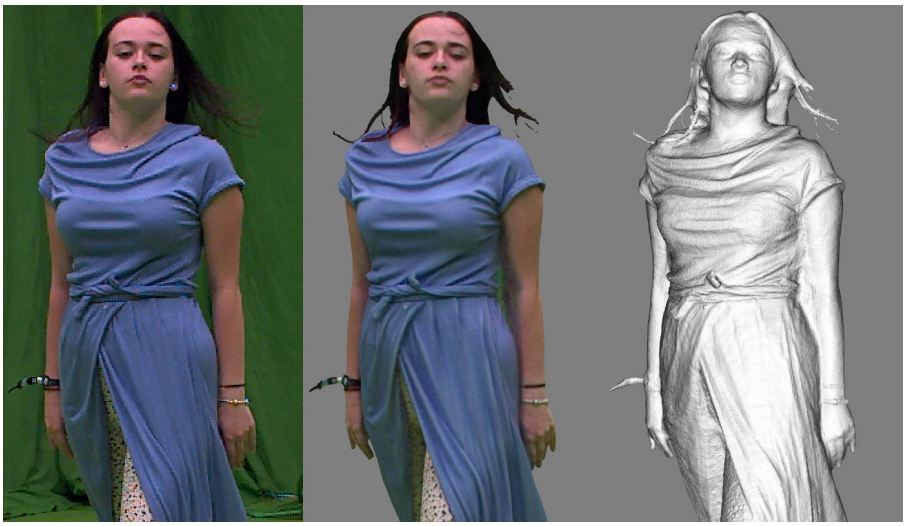

Figure 3.1: Typical results of our method: input image (left), colored mesh (center), geometry (right)



Figure 3.2: Close-up on another mesh: input image (left), colored mesh (center), geometry (right)

aforementioned quality requirement. Among those, volumetric differential rendering approaches such as the popular NeRF [26] have shown that scene estimation can be cast as a light field problem parameterized by a neural network. A rough surface can be extracted by thresholding the inferred opacity field, but better surface reconstructions can be extracted with dedicated differential rendering approaches centered on a surface parameterization [10, 45, 127]. While all of these methods produce amazing results, they typically require several hours per frame, and tens of hours for some. The new volumetric methods [5, 51, 96] addressed that issue and obtain results comparable to [26] in a few minutes and even a few seconds in the case of [5]. However, they do not recover meshes of good quality since the focus is on a radiance field scene encoding.

Contributions. In this chapter, we show that a volumetric framework can be modified to output better meshes thanks to the adjunction of appropriate losses. We also show that, with a simple non-neural parameterization analogous to [96] and an appropriate spatial encoding, the problem entirely fits in GPU memory, and can be cast as a pure optimization solved with classic, non-stochastic gradient descent. This yields a fully explainable implementation of differential rendering offering a new quality/computation time trade-off, whose results approach state of the art surface-based differential methods for mildly reflective models of the popular Nerf and DTU benchmarks, and quite

significantly improved model quality with respect to classic multi-view surface reconstructions in realistic human capture scenarios. Last, we show that these results are achievable in one to two minutes, an order of magnitude faster than a typical state of the art geometric reconstruction method applied in this context [8].

3.2 Methodology

Our goal is to extract a surface given a set of multi-view calibrated images $\mathcal{I}_n$ and background images $\mathcal{B}_n$. The background images are typically available in performance capture setups by recording the acquisition room when empty, giving a strong cue to isolate the subject of interest. We use a differential volume reconstruction as proxy, for which we add appropriate regularization losses (section 3.2.2), such that a surface is easily extractible from the resulting opacity volume.

3.2.1 Rendering model

The reconstruction volume contains two volumetric fields: opacity and color. The actual parameterization of the opacity field varies a lot between methods, ranging from a regular voxel grid [51], a sparse grid [96], hash tables [5], or a fully connected neural network [26]. The color field has special parameterizations to achieve view-dependent effects: a neural network [26], neural features fed to a smaller neural net [5, 51] or spherical harmonics [96]. Independently from the actual parameterization, all volumetric differentiable rendering methods are based on the non-linear optimization of the following color similarity loss:

$$\mathcal{L}_{\text{photo}} = \sum_{r \in \text{rays}} ||\mathbf{C}(r) - \mathbf{C}_{\text{gt}}(r)|| \tag{3.1}$$

where $\mathbf{C}_{\text{gt}}(r)$ is the ground truth color of the pixel from which the ray r originates. $\mathbf{C}(r)$ is the color obtained by sampling the volume at regular intervals along the ray r and accumulating color and opacity information over a finite range. We subsequently drop r from notation for clarity, and introduce the following:

$$\alpha_i \in [0,1] \quad \text{transparency of a sample} \tag{3.2}$$

$$\mathbf{c}_i \in [0,1]^3 \quad \text{color of a sample} \tag{3.3}$$

We opt for a sparse grid representation, where color and transparency are obtained by interpolating from neighboring grid samples similarly to [96]. Since the samples are regularly spaced, they can be indexed by an integer k along

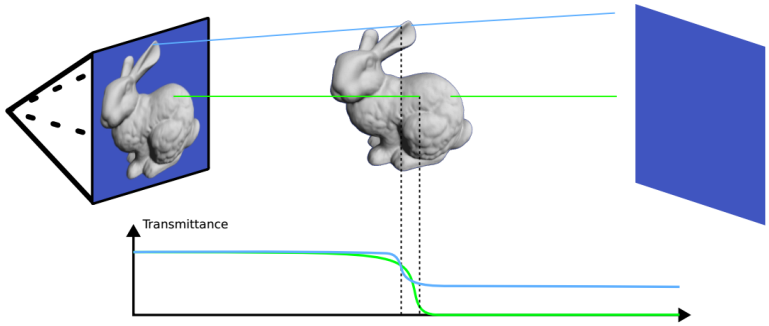

Figure 3.3: Volumetric integral example

the ray.

$$T_k = \prod_{j<k} \alpha_j \quad \text{partial transmittance} \tag{3.4}$$

$$\mathbf{C}_k = \sum_{j\leq k} (1-\alpha_j)\mathbf{c}_j T_j \quad \text{partial color} \tag{3.5}$$

$$\mathbf{C} = \sum_{k<\infty} (1-\alpha_k)\mathbf{c}_k T_k + T_\infty \mathbf{C}_{\text{bg}} \tag{3.6}$$

Note that for our purposes, a simple extension of the integration allows to account for the background $\mathbf{C}_{\text{bg}}$ in the final integrated color $\mathbf{C}$, when all ray opacities sampled before are transparent. We opt for the convention that $\alpha = 0$ for fully opaque media and $\alpha = 1$ for transparent regions. Figure 3.3 shows the transmittance value T_k along two rays. The transmittance starts at 1 but may not be 0 at the exit. In that case, the background is blended additively at the end of the ray integration.

The color and transparency of a particular sample are obtained by linear interpolation of the 8 closest voxels: $\alpha = \text{lerp}(\bar{\alpha})$ and $\mathbf{c} = \text{lerp}(\bar{\mathbf{c}})$, where the bar above a letter denotes voxels' values. We drop the indices for convenience. Following [96], we achieve view dependent effects, *e.g.* diffuse lighting and some specular lighting, by using spherical harmonics (SH):

$$\bar{\mathbf{c}}(\mathbf{v}) = \sum_{l=0}^{B} \sum_{m=-l}^{l} Y_{l,m}(\mathbf{v})\boldsymbol{\beta}_{l,m} \tag{3.7}$$

where $\mathbf{v}$ is the unit direction of a ray, $\boldsymbol{\beta}_{l,m} \in [-1,1]^3$ are the coefficients controlling the color and B is the number of SH bands. We typically use 0, 1 or 2 for B, which translates to 4, 13 and 28 coefficients per voxel respectively (including α). The $Y_{l,m}(x,y,z)$ functions are simple, computation-friendly polynomials.

3.2.2 Losses and regularization

Robust photometric loss. Let $||x||_{\mathcal{H}}$ be the robust Huber norm. If x is a vector, we take the sum of the norm applied component-wise.

$$||x||_{\mathcal{H}} = \begin{cases} \frac{x^2}{2} & \text{if } x < \epsilon \\ \epsilon(|x| - \frac{\epsilon}{2}) & \text{if } x \geq \epsilon \end{cases} \tag{3.8}$$

We opt for a robust differentiable version of the photometric loss, which mitigates spill of outlier colors :

$$\mathcal{L}_{\text{photo}} = \sum_{r \in \text{rays}} ||\mathbf{C}(r) - \mathbf{C}_{\text{gt}}(r)||_{\mathcal{H}} \tag{3.9}$$

Spatial Regularization. We classically use a total variation regularization over all voxels:

$$\mathcal{L}_{\text{smooth }\alpha} = \sum_{\text{voxels}} ||\nabla \bar{\alpha}||_2^2 \tag{3.10}$$

We also use a similar term on the voxels' color coefficients $\boldsymbol{\beta}_{l,m}$. Our coarse-to-fine scheme (3.3.2) allows to place sparse grid elements in the vicinity of the surface, with the assumption that non-grid regions are identified as fully inside or outside the object. Those regions are assigned boundary values $\alpha = \eta$ inside and $\alpha = 1$ outside where η is a small positive value. We make sure α is never zero to avoid a division by zero in the computation of the gradients. Similarly, we use black color boundary values for the inside and outside regions. Enforcing boundary values promotes a smooth and monotonous progression of the opacity which helps remove some volumetric noise and allows an easy extraction of a mesh by using marching cubes on $\bar{\alpha}$.

Ray color consistency. Following [51], we also use a per-sample regularization:

$$\mathcal{L}_{\text{sample}} = \sum_{\text{samples}} ||\mathbf{c}_k - \mathbf{C}||_{\mathcal{H}} \; T_k(1 - \alpha_k) \tag{3.11}$$

The purpose of this term is to favor voxel colors consistent with the input color observed by the ray, and to avoid occasional ray-dependent overfitting seen in NeRF, where input colors can be explained by a sum of view-dependent color contributions dispersed among a wider and non-physically meaningful range of voxels along a ray, with some individual colors drifting away from the observed ray color. We empirically observe that this term also drastically improves surface grazing ray colors, where two voxel color explanations may otherwise compete, background or actual surface color. The term also acts as a direct voxel color supervision term which favors better convergence.

Spherical harmonics parsimony. We penalize non-constant SH coefficients ($l > 0$) to prioritize the constant term, which empirically resolves more details for quasi-Lambertian surfaces in our experiments:

$$\mathcal{L}_{\text{SH_parsimony}} = \sum_{\text{voxels}} ||\boldsymbol{\beta}_{l,m}||_{\mathcal{H}} \tag{3.12}$$

Ballooning occupancy bias. Regions of an object with no texture or texture not visible at lower resolutions of our coarse-to-fine strategy are harder to reconstruct, because the coexistence of multiple photoconsistent surface hypotheses usually results in a wide band of multiple low opacity voxels, difficult

to convert to a mesh. A simple workaround is to artificially promote more opaque voxels, which inflates the surface:

$$\mathcal{L}_{\text{ballooning}} = \sum_{\text{voxels}} \bar{\alpha}^2 \tag{3.13}$$

This strategy works well when the surface is close to the visual hull, which is a reasonable expectation of human capture scenes, but can prevent good convergence for scenes with deep concavities or many fine structures.

Color normalization. Additionally, we estimate a per camera gain and bias value to account for differences in exposures, similarly to [43]:

$$\mathbf{C}_{\text{gt}} = g\mathbf{C}_{\text{uncorrected gt}} + b \tag{3.14}$$
$$\mathbf{C}_{\text{bg}} = g\mathbf{C}_{\text{uncorrected bg}} + b \tag{3.15}$$

3.2.3 Gradient descent

We chose not to rely on auto-differentiation for additional control and fine-tuning, since the losses admit simple analytical gradients. We here give the final expressions for the opacity derivative of the photometric term for illustration. Consider a particular sample of values α_k and $\mathbf{c}_k$ from a particular ray of integral $\mathbf{C}$:

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \alpha_k} = \frac{\partial \mathcal{L}_{\text{photo}}}{\partial \mathbf{C}} \frac{\partial \mathbf{C}}{\partial \alpha_k} \tag{3.16}$$
$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \mathbf{C}} = ||\mathbf{C} - \mathbf{C}_{\text{gt}}||'_{\mathcal{H}} \tag{3.17}$$
$$\frac{\partial \mathbf{C}}{\partial \alpha_k} = -\mathbf{c}_k T_k + \frac{\mathbf{C} - \mathbf{C}_k}{\alpha_k} \tag{3.18}$$

We march once per ray to build T_∞ and $\mathbf{C}$ (eq. 3.6), then march a second time (also in front to back order) to build $\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \alpha_k}$, which we store for each sample of each ray, according to the expression above. We can then compute derivatives with respect to the values $\bar{\alpha}$ of a particular voxel with the chain rule:

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \bar{\alpha}} = \sum_{r \in \text{rays}} \sum_{k \in \mathcal{S}(r)} \frac{\partial \mathcal{L}_{\text{photo}}}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial \bar{\alpha}} \tag{3.19}$$

$\mathcal{S}(r)$ is the set of samples on ray r to which the voxel of interest contributes.

One way of summing these contributions is to use *atomicAdd* instructions where each sample is assigned to one GPU thread. A special instruction is required instead of a regular addition in order to avoid data races that would occur when different threads update the same memory location. Atomic instructions are convenient but can be costly, especially under contention or cache misses, and are not deterministic since floating-point addition is not associative. We propose a different accumulation strategy that does not require special hardware support and is deterministic, although approximate. A sample can trivially determine the 8 nearest voxels required for interpolation but

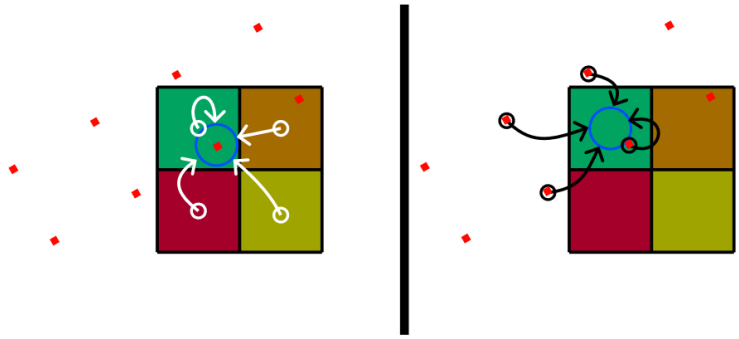

Figure 3.4: Gradient interpolation. Samples (red dots) are spaced uniformly along two adjacent rays. Left: interpolation of the nearest voxels in the forward pass. Right: interpolation of the nearest gradients in the backward pass.

obtaining the reciprocal mapping is difficult: a voxel does not have immediate knowledge of the set of samples using its value. Thus, we approximate eq. 3.19 by reading interpolated values of the per-sample gradients:

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \bar{\alpha}} \approx \sum_{\text{cameras}} lerp(\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \alpha}) \tag{3.20}$$

lerp() denotes tri-linear interpolation. Figure 3.4 illustrates our double interpolation scheme: on the left, each sample linearly interpolates values from the nearest voxels in the forward pass. On the right, each voxel interpolates gradients from the nearest samples in the backward pass. We can do so efficiently since the samples are stored in ray bundles with a pre-defined layout (see §3.3.1), with a uniform spacing along the depth.

The computations are analogous for the partial derivatives with respect to voxel colors, and for the similarly ray-based $\mathcal{L}_{\text{sample}}$ loss. The gradients of the voxel-wise regularizations $\mathcal{L}_{\text{smooth }\alpha}$, $\mathcal{L}_{\text{SH parsimony}}$ and $\mathcal{L}_{\text{ballooning}}$ are much simpler since they only involve a voxel and its immediate neighbors. For instance the gradient of $\mathcal{L}_{\text{smooth }\alpha}$ is:

$$\frac{\partial \mathcal{L}_{\text{smooth }\alpha}}{\partial \bar{\alpha}} = -\Delta\bar{\alpha} \tag{3.21}$$

We provide all gradient derivations along with a summary of the quantities defined in this chapter in appendix A.1.

3.2.4 Surface extraction

We extract meshes from the voxel opacity grid using marching cubes [11] on the transparency values $\bar{\alpha}$. The color is computed per pixel at rasterization time by sampling the volume at the location of the unprojected pixel, taking into account the view direction when evaluating the SH basis. The threshold for the extraction of the isosurface is chosen manually and fixed for all datasets at 0.81. This simple strategy proves effective, provided the optimization converged to an SDF-like distribution of transparency values.

3.3 Sparsity and coarse-to-fine strategies

Most differential rendering methods are prone to two main types of inefficiency: a complex parameterization and the redundant sampling of empty space. Our parameterization is as simple as possible: a sparse voxel grid with a two-level data structure. We also opt for a coarse-to-fine strategy for quick convergence, starting with down-scaled images and a coarse grid that we progressively upscale. We avoid sampling empty space in two ways: first, by computing the visual hull [85] for a rough estimate of the location of the surface, and second, by pre-computing integration ranges for all rays as determined by the estimation in the previous hierarchy level. The combination of the coarse-to-fine and sparse storage allows to focus the method on relevant portions of space while targeting very high resolution, millimetric voxels in our capture experiments, with contained compute times and memory.

3.3.1 Ray culling

To bootstrap convergence, we cull rays that don't initially intersect with the visual hull. Also, we reserve initial voxels using the visual hull's bounding box, in the space regions visible by at least one ray. Our sparse grid is stored as a 3D array of voxel tile pointers (Figure 3.5), where tiles contain 4^3 voxels. Voxels are initialized with a black color and $\alpha = 0.98$. We compute a similar sparse structure for the rays, grouped in 2D bundles of 8×8 pixels. We pre-compute integration ranges for each ray bundle, illustrated by the red frustums, by marching in our structure once. An integration range is a list of continuous sub-ranges, which skip empty space and stop when the transmittance is low. We sort tiles in Morton order for improved cache coherence [164].

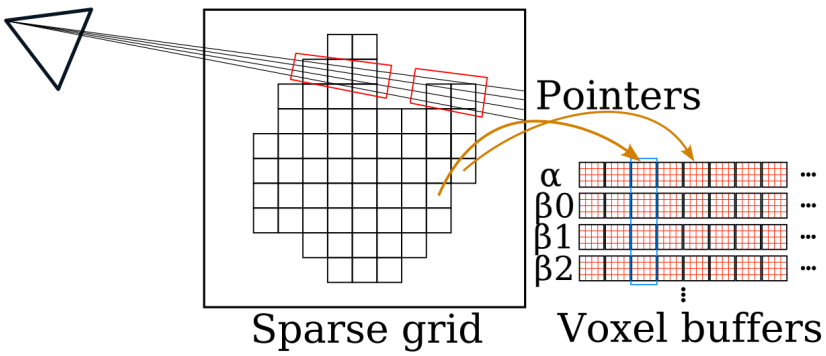

Figure 3.5: Block-sparse structure with one indirection.

3.3.2 Coarse-to-fine strategy

We start the optimization at a coarse resolution for both the grid and the images. After convergence has been reached for the current level, we remove the empty tiles and mark full tiles, then upscale the sparse structure at twice the resolution by linearly interpolating voxels of the coarser level. Using images of the next finer resolution, we recompute integration ranges for the new bundles

of rays. We determine which tiles are empty and which are full by computing the transmittance from each camera to each voxel.

3.3.3 Implementation details

We implement our optimization and visualization framework with C++ and OpenGL compute shaders. We use a step size equal to the diagonal of a voxel when marching in the volume. For datasets without backgrounds, we require segmentation masks and apply an additional loss $\mathcal{L}_{\text{mask}} = \sum_{\text{rays}}(T_\infty - mask)^2$ where *mask* is a binary image. The different losses are weighted by coefficients that we keep constant for all the experiments, see appendix A.3. We use the ADAM optimizer. Voxel coefficients are stored with 8 bit integers. All computations use 32 bit floats but we store gradients in 8 bit, $1, 5, 2$ floating point format. Since gradients are small, we bias the exponent to avoid conversion underflow.

3.4 Evaluations

To broadly characterize the performance of the method, we perform a range of experiments on two real data sets and two synthetic datasets, in both a generic reconstruction setup and surface capture setup. All our experiments were run on a Quadro RTX 5000 GPU with 16GB of video memory. We mark 'x' the table entries for which we were not able to either run the particular experiment (due to video memory limitations) or that the corresponding paper did not provide means of computing it. We list the optimization timings and peak memory used in our experiments. An additional two seconds are required to load the images, compute the visual hull and initialize our data structures. Additional content can be downloaded from `https://hal.inria.fr/hal-03748662`.

3.4.1 Validation with general-purpose benchmarks

Nerf synthetic dataset comparisons. We perform a comparison with [26] and [96] on the Nerf synthetic dataset to characterize general performance. This dataset is composed of 8 scenes imaged by 100 cameras arranged in an hemisphere, with easily extractible background segmentations. The appearance is evaluated with another set of 200 cameras, different from the training set, with all images of resolution 800^2. Note that many objects have highly reflective surfaces and intricate geometry. The quantitative results are summarized in table 3.1. We report the averaged PSNR values given in [96]. Our method performs encouragingly except for strongly reflective surfaces and concavities.

Real data comparisons on DTU. We present results on the classic DTU dataset [34], composed of 49 to 64 close-up pictures of various objects with challenging elements. First, the high resolution images (1600x1200) have inconsistent lighting conditions due to the robot arm casting shadows on the

Scene	Nerf	Plenoxels	Ours $B=1$				Ours $B=2$			
	PSNR	PSNR	PSNR (raster)	PSNR (vol)	Time (s)	vram (GB)	PSNR (raster)	PSNR (vol)	Time (s)	vram (GB)
chair			26.48	26.28	27.1	5.4	27.59	26.72	53.6	6.2
drums			21.85	21.56	25.6	5.6	22.22	21.90	39.9	6.2
ficus			25.06	26.78	24	5.3	26.32	28.03	30.5	5.6
hotdog			28.45	28.12	40.9	5.9	28.77	28.57	68.1	6.8
lego			26.52	26.31	33.4	6.4	26.68	26.40	64.2	7.3
mic			24.22	24.66	17.2	3.8	25.55	25.64	33.0	4.1
materials			22.28	24.11	24.9	5.9	23.30	24.40	40.4	6.5
ship			26.48	26.44	65.8	8.8	27.05	26.91	153.3	10.2
mean	31.01	31.71	25.2	25.5	32.3	5.9	25.9	26.0	60.4	6.6

Table 3.1: Quantitative results on NeRF synthetic dataset.

objects, and second, only the front is visible in most scenes. This makes it hard to compute an accurate visual hull so we manually fit a tight bounding box around the objects. Since no background images are provided, we use the segmentation masks of [45]. We used only one band of spherical harmonics ($B=1$) to keep the problem GPU fitted since the object covers much more rays than for typical human acquisition setups. Table 3.2 shows some quantitative results for a scene subset with the DTU average Chamfer-$L1$ distances metric. We compare against [10, 45] in terms of geometry and appearance. We evaluate both with rendered images obtained by surface rasterization (raster columns), and images obtained by computing the volumetric integral (vol columns). The PSNR attained in the latter is slightly higher than in the former since more voxels contribute to the value of a pixel. Our method works best for highly textured, mildly-specular materials but misses some specular or overly concave parts (see qualitative results below). Our Chamfer distance is consequently worse than [10, 45] but our method achieves higher PSNR, and is 2-3 orders of magnitude faster.

3.4.2 Qualitative results on DTU and NerF datasets

We show qualitative results for tables 3.1 and 3.2. We used a black background during the optimization, which is why the volumetric rendering columns also have a black background. Our method works best for highly textured, non-specular materials (scans 65, 106, 114, 122) and yields bloated / caved-in surfaces otherwise (scans 24, 37, 110). Note that the windows of the house (scan 24) are well reconstructed since they provide a strong texture gradient even at low resolutions.

Scan	IDR		Neus		Ours $B = 1$				
	Chamfer	PSNR	Chamfer	PSNR	Chamfer	PSNR (raster)	PSNR (vol)	Time (s)	vram (GB)
24	1.743	23.30	0.831	x	2.8932	24.43	25.03	98.5	14.3
37	x	x	1.101	x	3.4356	22.026	23.05	97.5	14.1
65	0.892	23.95	0.721	x	1.0942	25.26	26.37	78.0	12.2
106	0.728	22.81	0.532	x	1.4070	28.35	30.59	95.9	12.9
110	1.10	21.26	1.416	x	2.9493	25.36	28.47	93.2	13.5
114	0.450	25.36	0.331	x	0.7607	24.36	25.54	84.6	11.9
122	0.706	27.06	0.552	x	1.0001	31.18	32.71	63.3	10.3
mean	0.936	23.96	0.79	x	2.05	25.7	27.1	88.7	12.9

Table 3.2: Quantitative evaluation on the DTU dataset.

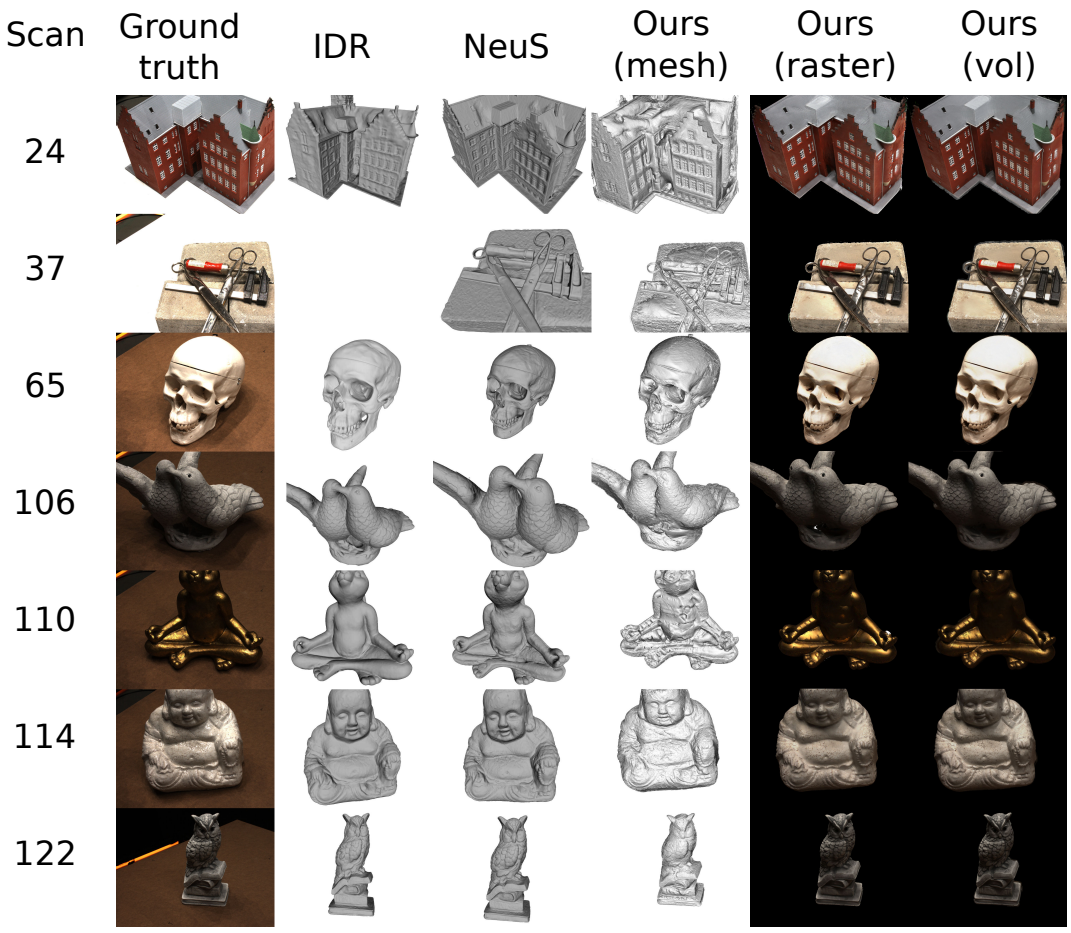

Figure 3.6: Qualitative results on the DTU dataset.



Figure 3.7: Qualitative results on the nerf synthetic dataset.

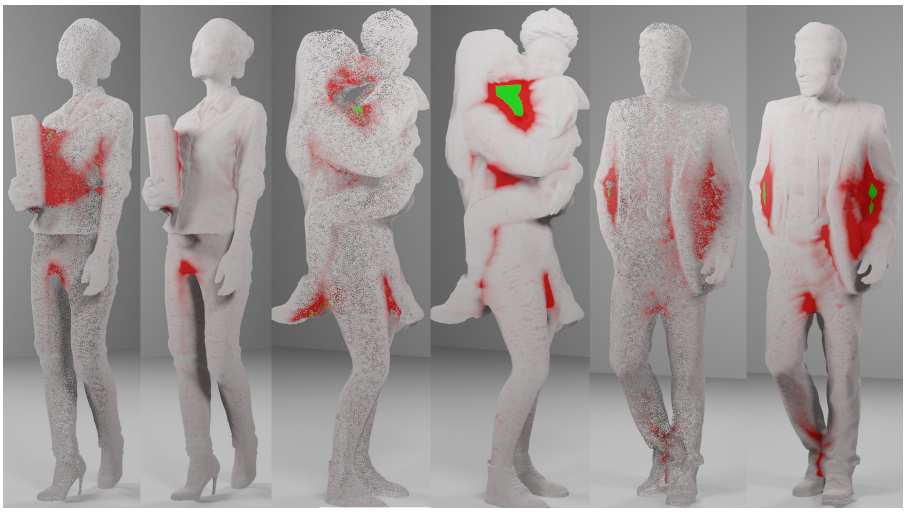

Figure 3.8: Chamfer heatmaps error comparison on synthetic human benchmark with 3 models, [8] (left columns) vs ours (right columns).

3.4.3 Surface capture evaluations

Quantitative synthetic human benchmark. We test our method on 3 free meshes from *renderpeople.com*, with 60 rendered viewpoints in a circle around the objects, rendered on a white background. We report the mean Chamfer distances (centimeters) in table 3.3. Points further than 5 centimeters are ignored similarly to the DTU evaluation protocol (green points in Figure 3.8). Our method performs on par or better than SOTA [8] on this data, while executing around 20 times faster. Note also that, although they are widely used for quantitative reference, such semi-synthetic datasets have intrinsic limitations such as lacking reflective component realism or fine-scale geometric details (*e.g. wrinkles in clothing, hair strands).* As in renderpeople, this often directly results from limitations of the capture method used to obtain the reference geometry itself, which we are attempting to address.

	Mei	Fabienne	Dennis
Our method	0.44	0.45	0.62
Leroy *et al.* [13]	0.44	0.49	0.62

Table 3.3: Chamfer evaluation (lower is better).

Qualitative comparison on real data. We compare with the state of the art surface capture method [8] in Figure 3.9, on data captured by the Kinovis multi-camera platform. It is composed of 68 2048×2048 RGB cameras arranged in an hemisphere with a radius of approximately 4 meters. Our method provides a drastic improvement in the reconstruction of all high frequency details, such as cloth wrinkles, face features, which are mostly smoothed out with [8]. The distinct geometry of fingers and veins are clearly visible in Figure 3.2, despite the noisy input. We want to emphasize that all of these results

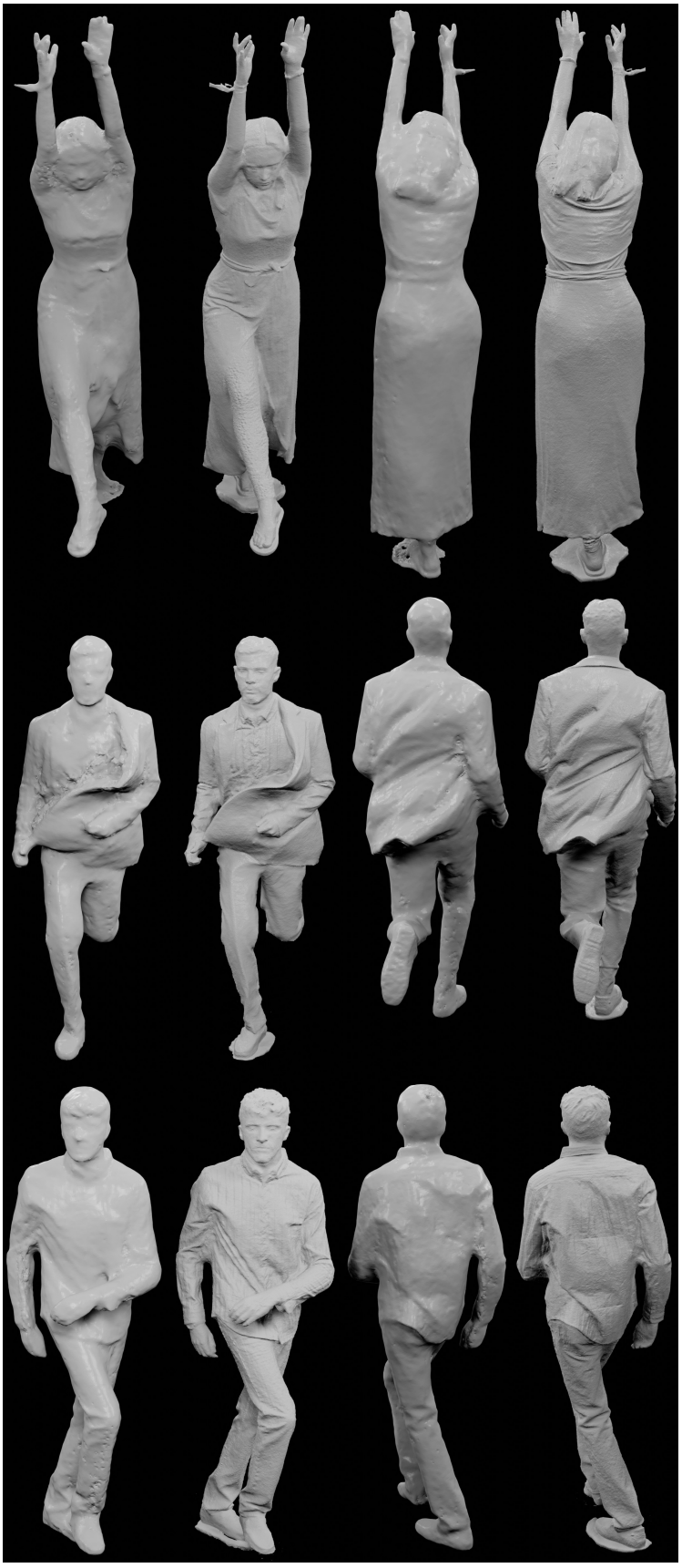

Figure 3.9: Ours (columns 2 & 4) vs [8] (columns 1 & 3)

	B=2	B=1	B=0	no ballooning	no SH parsimony	Coarse (left) to fine (right)			
PSNR raster	25.9	24.3	23.0	25.7	25.7	23.3	24.2	24.8	25.9
PSNR vol	25.7	24.6	23.4	25.7	25.7	23.0	24.4	25.3	25.8

Table 3.4: Ablations of regularization terms on the mic model

were routinely obtained in less than a minute of computation time. The meshes obtained contain 5 million triangles on average, with each triangle projecting on areas smaller than one pixel. We use 4 levels of detail in our coarse-to-fine strategy, with voxels of 1.25 mm at the finest resolution.

3.4.4 Performance analysis

Ablation of regularization terms. We ablate our regularizations and coarse-to-fine scheme in table 3.4 and Figure 3.10 on Nerf's mic model. Varying the number B of SH bands has the highest impact on quality and performance. The surface has a strong tendency to buckle when the reflectance cannot be well approximated by the SH decomposition. We mitigate this effect to some extent with our ballooning regularization but it creates a global bloating of the surface that can be hard to overcome in some scenes. Dropping the spherical harmonic parsimony term results in a faded base color and creates additional surface buckling, even though the overall color is more vibrant.

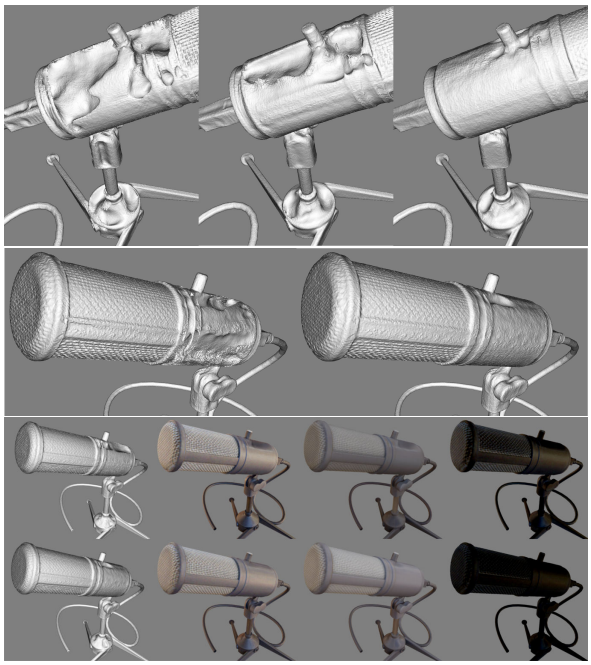

Figure 3.10: Top: $B = 0$ (right), $B = 1$ (center), $B = 2$ (left)
Center: No ballooning term (right), with ballooning (left)
Bottom: Without SH parsimony term (top row) vs with (bottom row). From left to right: geometry, mesh colors, $\beta_{0,0}$ only, specularity only

Ablation of color normalization. The influence of the color normalization term can only be characterized on real data where per-camera exposure discrepancies actually occur, as opposed to synthetic datasets which don't show any. Table 3.5 ablates this optimization term on three frames of our captured data, showing the increase of PSNR resulting from inclusion of the term.

	with color normalization		no color normalization	
	raster	vol	raster	vol
cartwheel t=182	22.6	26.7	21.8	25.3
run t=168	24.6	29.0	23.5	26.9
dance t=367	23.4	26.1	22.5	24.9

Table 3.5: PSNR on real data

Trade-offs of surface extraction strategy. Our regularization and ballooning terms successfully bias the estimation toward opacity fields with well-defined surfaces, free of density artifacts as Figure 3.11 illustrates. Our method then aggressively culls inner volumes deemed far away from the surface (in green in Figure 3.11, left), with sparse cells only in the vicinity of high opacity gradients. While this allows very speedy inference in most cases, it also means the method may miss some concavity information to the resulting volume inflation bias, which explains some of the performance limitations observed in 3.4.1, or *e.g.* seen with over-merging under armpits of subjects in Figure 3.9. The proposed strategy is meant and shown to offer its most compelling tradeoff for the targeted capture cases of mildly concave and mildly reflective subjects.

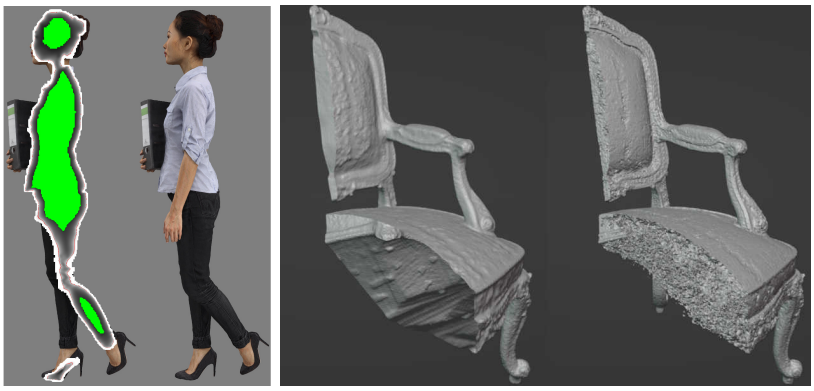

Figure 3.11: Left: Sectional view of the opacity field. Right: Extracted isosurface. Ours (left), [5] (right)

3.5 Conclusion

This chapter investigated a simple and efficient differential rendering implementation, targeted at multi-view surface capture problems. Our choice of

regularization adapts the differential volumetric rendering framework for the recovery of highly detailed meshes, where previous methods had many imperfections such as holes and noise or take several orders of magnitude longer. Our coarse-to-fine strategy and custom gradient descent optimization allow fast convergence on a single GPU, typically in a minute, which was our main objective for this chapter. Additionally, we showed that computing analytical gradients is not overly complex thanks to a simple parameterization, casting differential rendering as a classic optimization.

3.5.1 Discussion

The recovered geometry contains details at a millimetric scale thanks to the proposed high-resolution framework, which suggests that an overall millimetric accuracy is possible. However, a number of limitations are preventing us from reaching this goal yet.

In particular, we identify that our smoothness regularizer with explicit boundary values, as described in section 3.2.2, tends to bias the surface to the middle of the allocated voxel band. Our ballooning strategy in eq. 3.13 applies a generalized inflating term, that also impacts regions that do not need it. We find that these two regularizations are needed to define inside and outside regions, but this small bias also degrades surface quality where the photometric gradients are too weak. This leads us to investigate an implicit surface representation in the following chapter, specifically crafted to be unbiased [10].

The proposed non-stochastic gradient descent batching all the rays at once is another limiting factor, due to its important memory consumption. The per-voxel spherical harmonic basis is also costly in VRAM. The next chapter will propose an alternative training strategy and a different appearance model with much lower memory requirements.

Acknowledgements

This work was supported by French government funding managed by the National Research Agency under the Investments for the Future program (PIA) grant ANR-21-ESRE-0030 (CONTINUUM). The 3-people dataset was sponsored by Naverlabs Europe, and acquired with the Inria Kinovis[1] platform.

[1] https://kinovis.inria.fr

4

Millimetric Human Surface Capture in Minutes

4.1 Introduction



Figure 4.1: Left: Reconstructions for 3 examples of dynamic human capture [9], including body and long hair motion. Right: illustration of results on the proposed MVMannequin scanned mannequin database: input view, hand-scanned reference, our reconstruction. Note in both cases, the fine captured surface detail such as hair strands, facial features, cloth folds and creases, and the sewing micro-geometry.

We have shown so far that differential rendering is an effective tool for surface capture, outperforming typical multiview stereo approaches in this setting, while being much faster. The pixel-wise and continuous formulations are likely the main reasons behind the improvement in quality, as multiview

stereo is limited by larger pixel patches needed for photoconsistency evaluation and discrete depths hypotheses. The first point is particularly relevant here, since few pixels are covered by the person due to the wide field of views required for the capture of large motions in a capture platform.

In this chapter, our goal is to reduce the gap in accuracy with respect to competing differential rendering approaches [10, 26], that we noted in the conclusion of the previous chapter. Specifically, we take inspiration from neural methods with a signed-distance function surface parametrization [10, 45], as they yield state of the art surface reconstruction performance on generic object benchmarks such as [165, 166]. However, these approaches cannot be readily used in our context because of their expensive neural parametrization. We thus propose to store explicit signed-distance values in a sparse voxel grid, which we expect to be more efficient than training a deep neural network to output these at every point of space.

At the same time, neural networks are appealing due to their ability to compress information in a few parameters, and could significantly reduce the model size of our previous parameterization. This leads us to replace the per-voxel spherical harmonic basis of the previous chapter with a shallow neural network, taking as input a smaller feature vector. Most grid-based approaches [30, 31, 51, 60] use this formulation, which comes with a substantially larger inference cost compared to non-neural representations [27]. We estimate that this issue can be avoided if the role of the neural component becomes minimal, following a principle of parsimony. Our idea is to encode features at a higher resolution, again leveraging our sparse grid, but with a lower dimensionality, and therefore decreasing the width of the color prediction network.

We also investigate ways to decrease memory consumption during training, identifying redundant temporary storage as wasteful in that regard. Thus, we develop a fully-fused strategy for ray marching and for the color prediction: ray samples and hidden layer activations are never materialized in memory, not even during training. An interesting side effect is that we can handle an arbitrary number of samples for each ray, self-determined during scene traversal, thanks to an adaptive step size based on an importance sampling scheme.

Actually verifying real-world accuracy in the context of human surface acquisition proves to be a key challenge, as it requires a highly accurate geometric surface reference to compare against: typical real-world datasets provide it for generic objects [165], but are seldom found for multi-view human datasets. We thus introduce the MVMannequin dataset alleviating these drawbacks, comprised of 14 dressed real-size human mannequins in casual pose, providing both sub-millimeter hand-scans of their geometry, and their input images obtained from 68 calibrated viewpoints. We specifically opt for dressed mannequins for their perfect stillness, avoiding any micro-motions such as breathing or muscle tension change, that would yield perturbations beyond this bound. We further demonstrate qualitative achievements of the algorithm on publicly available multi-view human capture sequences of the 4DHumanOutfit dataset [9]. In summary:

- We introduce MVMannequin, a real-world dataset of 14 human-realistic

dressed mannequins both male and female with varied clothing, providing both calibrated multi-view observations and a hand-scanned reference of sub-millimeter accuracy.

- We provide a scalable approach for human surface capture from multiple images, with a novel sparse dynamically updated hierarchical backbone, achieving 1.13mm average accuracy and sub-millimeter accuracy for 77% of surface sampled points on MVMannequin reference data.

- Code and data are available at https://projects-morpheo.gitlabpages.inria.fr/millimetrichumans/.

4.2 Related work

Neural surface reconstruction. Initial volumetric approaches [26, 43] have drawn attention to the powerful capabilities of neural radiance modeling and differential rendering, with recent iterations showing various strategies to make the representation magnitudes more efficient [5, 27]. It is however non-trivial to extract surfaces from the resulting volumetric opacity fields, prone to holes and floaters due to lack of surface regularization. Some promising methods thus reparametrize Gaussian splats for alignment to an underlying surface representation [47–49]. Yet, state of the art surface reconstruction performance is achieved by a concurrent family of neural methods that parametrize the shape implicitly with a signed-distance function (SDF) [10, 128]. Various improvements to boost the accuracy of these methods exist, *e.g.* [44, 46] incorporate a patch-based photo-consistency loss inspired by classic multi-view stereo algorithms, though at a significantly increased computational cost. [29] achieves outstanding accuracy on large scale scenes thanks to a combination of hash grids and numerical gradients, but trades speed for detail. Closest to our objectives, [31] combine hierarchical hash grids and implicit surface parametrization to achieve both accurate reconstruction and fast training, and [30] achieve an impressive combination of high quality rendering and geometric accuracy using a dense voxel grid. All these approaches have yet to be thoroughly validated in the context of human surface capture.

Neural pipeline optimization. Various works are relevant to the scalability and efficiency of the neural surface pipeline. The use of regular grids [30, 51, 96], hierarchical hash tables [5, 31] or lower-dimensional component factorization [60, 142, 143, 167] were all designed to reduce the size of the MLP needed to predict color and opacity/SDF, by parametrizing it with a set of local features instead of using one scene-global network. Hierarchical hashing and factorization, later unified with the works of [146], add a level of sparsification of the neural parameters, while making approaches more compute-friendly and easier to implement. Accelerating the ray-querying of these structures has also been a topic of significant interest, as it usually benefits both training and rendering applications. Again, hierarchical grids [53] or hashtables [5] can be leveraged to this end, along with reparametrization of the angular query component

with spherical harmonic projection [53] or with a small MLP [55]. Rendering-specific optimizations also exist, *e.g.* baking a pre-learned scene into a sparse grid of features, and deferring the MLP color conversion to the pixel domain by performing ray integration on MLP features instead of color [54]. Sparse hierarchical structures may benefit volumetric differential approaches [150], and yield efficient volumetric query implementations on GPU [148]. Our approach unifies these different training, render and representation optimizations, transposes them to neural SDF approaches, and introduces sparse hierarchies that can be updated on-demand at training time.

Human prior models for capture. For completeness, we briefly mention that the geometry estimation problem can be tackled with human prior shape models, *e.g.* by fitting articulated [71], parametric [20], Neural parametric [21, 168] or part-based models [169]. Such model-based approaches are valuable for many applications, *e.g.* to drive and constrain capture of appearance-realistic image-based avatars, where face, hair, or clothing are explicitly modeled as additional rendering layers [170, 171], or added geometric surface detail layers which can be physics-aware [172–174], or both [175], under tight functional constraints *e.g.* real-time [70] or monocular input [176]. Yet generally they pull estimation toward a higher-level prior distribution of shapes and away from measurement of very high frequency detail by construction. We pursue a different direction where such accuracy is the primary objective, notwithstanding the interest in semantically richer shape models.

3D validation datasets. Validating the quality of 3D reconstruction from multi-view images was pioneered with the Middlebury Multi-View Stereo Benchmark [33], which introduced the notions of *accuracy* and *completeness* as Chamfer distances to and from the reconstructed output *w.r.t* a laser-scanned reference. Many datasets have followed, *e.g.* to increase resolution and object variety [165], or for open scenes [37, 166, 177, 178], with the same validation methodology, simultaneously providing registered input views and reference scans of orthogonal modality. While very useful generally, these datasets have in common that they do not match typical full body human acquisition setups, where wider fields of view are used to allow for movement, and humans are imaged onto a small projected sensor area. Concerning 3D human datasets, on one hand many provide parametric human model data [179, 180] or 3D body surface data obtained through photogrammetric or laser scans [16, 181–183], but do not provide matching captured camera images thereof to be used as input for reconstruction. Others do provide multi-view images and 3D reconstructions from these viewpoints, but lack the independently measured comparative 3D reference [3, 52, 184–187]. Finally, some datasets build on the 3D models of the above 3D human datasets as ground truth, providing synthetically rendered viewpoints [188–190] but are intrinsically limited for our purpose due to their domain and use case gaps. More importantly, most come with undesirable caveats built-in: *i.e.* their ground truth lacks millimetric geometric surface detail due to decimated resolution [16], or is itself reconstructed from multi-view photogrammetry with the same measurement bias as the algorithms we need to evaluate. While all mentioned datasets are useful to train human AI models thanks to the

wide subject, clothing and pose variability they provide, they are thus less than ideal for our validation task. This motivates our MVMannequin dataset, providing both high density sub-millimetric 3D hand-scans of human-like bodies in varied clothing, and the corresponding input images for multi-view reconstruction.

4.3 Method

4.3.1 Preliminary Notions

Differential rendering models [10, 26] can be functionally described as follows. For a given *forward pass*, image pixel colors are predicted for a set of rays in four steps: (1) for each ray, originating from the camera center o through an image pixel in the viewing direction v expressed as $\{p(t) = o + t\,v \mid t \geq 0\}$, select a discrete set of M 3D scene point samples $\{p_i = o + t_i\,v | i = 1, ..., M, t_i < t_{i+1}\}$ on the ray according to a *ray sampling model*, (2) for each 3D point sample fetch parameters from a *scene representation*, *e.g.* a feature grid, (3) decode them to a given 3D color and opacity, using *e.g.* shallow MLPs, (4) integrate them in ray marching order according to a *rendering model*. The latter is typically the volumetric blending model of NeRF and NeuS [108]: where the rendered color of a pixel is discretely approximated by the blended sum along its M sampled points on the ray:

$$\mathbf{C} = \sum_{i=1}^{M} (1 - \alpha_i) T_i \mathbf{C}_i, \qquad \text{with } T_i = \prod_{j<i} \alpha_j \tag{4.1}$$

where α_i is the transparency value[1], T_i is the accumulated transmittance. Neus differs from NeRF in that α_i is decoded from an SDF field as a roughly Gaussian mass of width $1/s$ centered on its 0-level set:

$$\alpha_i = 1 - \max\left(\frac{\Phi_s(f(p_{i-1})) - \Phi_s(f(p_i))}{\Phi_s(f(p_{i-1}))}, 0\right), \tag{4.2}$$

where f is the SDF function, $\Phi_s(x) = (1 + e^{-sx})^{-1}$ the Sigmoid function, with the s value learned or scheduled during training.

Let $f_{i-1} = f(p_{i-1})$ and $f_i = f(p_i)$ be two consecutive signed-distance values sampled along a ray:

$$\alpha_i = 1 - \max\left(\frac{\Phi_s(f_{i-1}) - \Phi_s(f_i)}{\Phi_s(f_{i-1})}, 0\right) \tag{4.3}$$

$$= 1 + \min\left(\frac{\Phi_s(f_i) - \Phi_s(f_{i-1})}{\Phi_s(f_{i-1})}, 0\right) \tag{4.4}$$

$$= \min\left(\frac{\Phi_s(f_i)}{\Phi_s(f_{i-1})}, 1\right) \tag{4.5}$$

$$= \min\left(\frac{1 + \exp(-s f_{i-1})}{1 + \exp(-s f_i)}, 1\right) \tag{4.6}$$

[1] We use the convention where α_i denotes the transparency of a sample instead of its opacity: $\alpha_{\text{Ours}} = 1 - \alpha_{\text{Neus}}$.

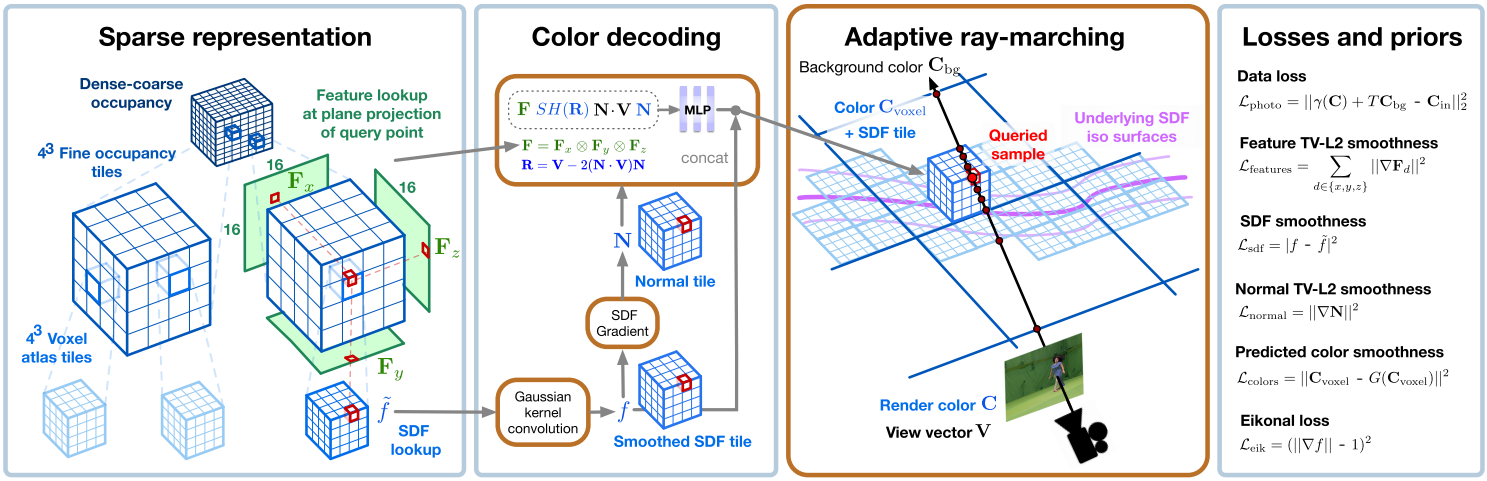

Figure 4.2: Learnable features and SDF are stored as a sparse hierarchy, with factor decomposition of features at the intermediate level. All leaf voxel tiles are processed for inner SDF smoothness and normal computation. Color tiles are predicted using a tiny MLP with feature product, normal, normal and viewing ray vector dot product, and spherical harmonic projection of the reflected view vector, to preserve its high frequencies. An efficient ray marcher converts the SDF to opacities [10] and integrates the volume to produce pixel-wise color predictions. All orange boxes correspond to one GPU compute kernel in the forward and backward passes. The figure centers on a sample point query on a ray, and materializes corresponding features and voxels in red.

3D reconstructions can be obtained from measured images by inverting the differential rendering model under a set of regularizing losses (*e.g.* Eikonal) to obtain scene-specific parameters. An *estimation algorithm* iteratively applies the forward pass to a selected *batch of rays* and minimizes a *rendering loss* between their predicted pixel colors and measured values, by back-propagating the loss gradients to scene parameters using the chain rule (*backward pass*).

4.3.2 Key Technical Contributions

We design a NeuS-inspired neural differential approach to reconstruct human scenes, with all components geared toward drastically maximizing representable resolution, and compute, rendering and GPU memory efficiency. Our forward pass and losses are summarized in fig. 4.2. We additionally leverage background images devoid of the subject of interest, specifically available in this context.

Scene representation. Our approach proposes a novel fixed-depth, constant-overhead sparse voxel-tile hierarchy to store scene parameters (detailed in §4.3.3), leveraging the inherent sparsity of human motion scenes. Each leaf tile voxel contains a scalar corresponding to its signed distance to the underlying shape, and compactly allows to decode a view-dependent color from the outer product of three local feature planes. This provides an inherent memory and compute advantage over dense [51], or hashed grids [31] that store all voxels or scale resolutions on GPU, in turn allowing reconstructions of unprecedented resolutions.

Coarse-to-fine estimation. To avoid resolution limits inherent to fixed grid representations, some approaches [30, 51] refine by subsequently iterating over two grid resolutions. Similar to [150], we lift this restriction with arbitrary coarse-to-fine refinement (fig. 4.3), by periodically restarting with an up-scaled resolution until a desired leaf resolution is reached. This leverages our fixed-overhead storage throughout, with memory scaling linearly in the number of occupied leaf tiles.

Deferring SDF decoding to the ray-marching stage (§4.3.5) and **pre-computing decoded colors** per-voxel tile (§4.3.6) leads to a major remapping of forward pass stages on GPU with only 4 total kernel calls (fig. 4.2), and unlocks a set of desirable properties eluding previous approaches, such as arbitrary size ray-batching for full view sets, per-ray adaptive sampling, with single-kernel MLP-based color decoding and ray-marching. Pseudo-code of CPU/GPU calls is available in appendix B.1.

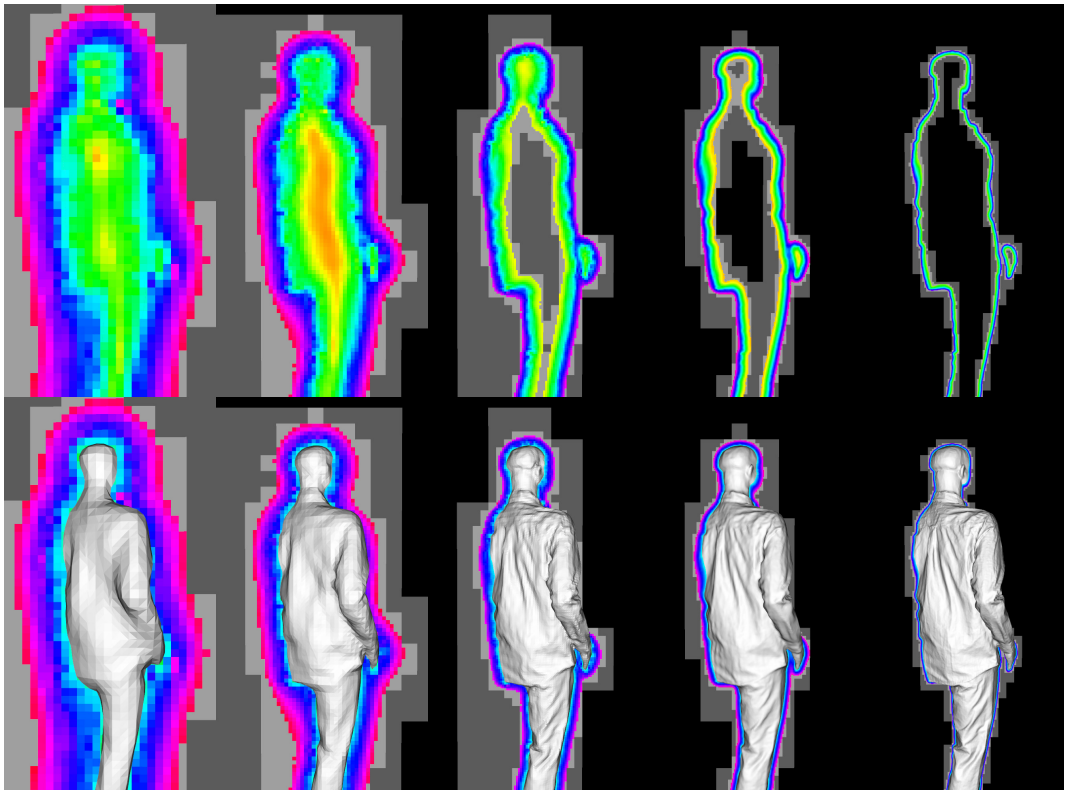

Figure 4.3: Our method continuously prunes empty space as optimization progresses. Coarse occupancy tiles are shown in dark gray and fine occupancy tiles are in light gray. Signed-distance values are mapped to a hue to emphasize the level set regularization. The mesh extracted from the level set is shown in the bottom row [11].

4.3.3 Sparse Scene Representation

Flexibility is key at optimization time to closely adapt to an evolving shape estimate, while retaining sparsity. At rendering time, it is paramount to efficiently skip empty space to sample the volume parsimoniously. The require-

ments of sparsity and adaptability are challenging to fulfill simultaneously, with a range of possible representations analyzed in appendix B.2.

Sparse storage. We opt for an N^3 tree [152, 153] as a sparse backbone structure, with the key originality that our proposal is both entirely managed on GPU and supports dynamic allocation. These trees generalize octrees in that each node has zero or N^3 children, with N the custom branching factor at each level. We empirically settle for a custom root node with larger N, akin to a coarse but dense grid, and two additional tree levels with $N = 4$, as shown in fig. 4.2. This configuration allows for better cache coherency with a constant number of indirections and only a small memory penalty compared to an octree. Each level can be packed in a 3D texture for hardware-accelerated lookups, with voxels grouped into leaf node tiles of size 4^3. These can contain decoded colors and the SDF as a single 4-tuple per voxel. The first two levels only encode occupancy at a coarse and finer level, and essentially store a grid of 3D indices pointing to a 4^3 tile into the next level.

Color feature reduction. We take inspiration from [142, 146], and generate per-voxel color features by taking the outer product of three axis-aligned planes of features: $\mathbf{F} = \mathbf{F}_x \otimes \mathbf{F}_y \otimes \mathbf{F}_z$. Note that $\mathbf{F}$ can be computed on demand and doesn't require specific storage. This empirically reduces leaf-tile storage by 50% to 75% compared to naive per-voxel storage of $\mathbf{F}$. Note that our tri-plane factorization is local to each mid-scale tile encompassing 16^3 voxels, rather than global, for further sparsity.

Dynamic allocations. We maintain a stack that keeps track of unused tiles in order to manage the memory pool from GPU-side operations only, with recursive updates to maintain a coherent hierarchy. This allows dynamic allocation and deletion of voxels or tiles anywhere in the volume after every optimization step, as illustrated fig. 4.3. We delete or introduce voxels based on conservative hysteresis thresholds of their SDF value, such that $|f| > \tau_{\text{max}}$ are deleted whereas voxels with $|f| < \tau_{\text{min}}$ trigger allocation of missing direct neighbors. We derive explicit values for τ_{max} and τ_{min} in the following section, so that rays are guaranteed not to exit too early.

4.3.4 Allocation bounds

The allocation bounds τ_{min} and τ_{max} used to cull or instantiate voxels are derived from the s-parameter of the rendering equation eq. (4.6) such that the transmittance T of a ray is guaranteed to fall below a given threshold η before exiting the band of allocated voxels. The derivation in eq. (4.7) assumes the sdf decreases monotonously along the ray. If not, it means the surface part was too thin and the ray will exit with a transmittance $T > \eta$, and will possibly intersect another part of the surface.

$$T_N = \prod_{i<N} \alpha_i = \prod_{i<N} \frac{1+\exp(-sf_{i-1})}{1+\exp(-sf_i)} = \frac{1+\exp(-s\tau_{\text{max}})}{1+\exp(+s\tau_{\text{max}})} < \eta \tag{4.7}$$

$$\iff \eta \exp(s\tau_{\text{max}}) > 1 \tag{4.8}$$

$$\iff \tau_{\text{max}} > -\log(\eta)/s \tag{4.9}$$

In practice, we use $\tau_{\text{min}} = 0.8\tau_{\text{max}}$ with $\tau_{\text{max}} = -1.5\log(\eta)/s$, slightly larger than the theoretical bound because more voxels are needed for interpolation.

4.3.5 Adaptive Ray-Marching with Deferred SDF Decoding

Virtually all SOTA implementations including [5, 30] bake scene lookups into tensors of dimension proportional to $R \times M \times F$, with R, M, F the number of batch rays, per-ray samples and features respectively. They are handily obtained from vectorized storage lookups and MLP queries on the full sample set of all batch rays. However, all intermediate operations, *e.g.* per MLP layer, generate GPU kernel calls and persistent tensors of similar size needed for back-propagation. This creates a large GPU memory burden that imposes practical limits on batch sizes (often 4096 rays).

In contrast, our approach avoids such overhead by pre-computing MLPs for every non-empty voxel tile beforehand in a temporary tile set, under a revised color decoding model discussed in §4.3.6. Ray-marching can then query colors and SDFs on-demand from temporary tiles, and perform late decoding of the opacity per eq. (4.2) with a single GPU kernel call over all rays and no ray-temporary tensors. Back-propagation can be performed similarly by accumulating gradients in a single set of voxel-tile temporaries instead of multiple memory intensive ray-batch tensors. We can thus process all rays of a given viewpoint at once, and keep iteratively accumulating gradients for arbitrary view sets in the same temporary tile set. To avoid naive uniform sampling, most of which would query empty space, our ray-marching dynamically skips empty tiles in our representation with a hierarchical version of the DDA algorithm [191]. In non-empty regions, we adapt integration step sizes on-the-fly with little overhead thanks to on-demand SDF access. We give a derivation of our adaptive stepping strategy in the following paragraph.

Adaptive stepping. It has been shown that sampling the transmittance uniformly is beneficial for performance [192], and is often achieved by relying on expensive stratified sampling schemes or proposal networks [26]. We can do so more conveniently since the sdf is encoded explicitly in the voxel grid. The basic idea is to compute a step size Δt such that the transmittance at the next sample will decrease by a fixed amount T_{step}. Starting from the known sdf value of the last sample f_{i-1}, we compute an expected value $\hat{f}_i$ in eq. (4.12) that would achieve the desired decrease in transmittance.

$$T_{\text{step}} = T_i - T_{i+1} = T_i - \alpha_i T_i \tag{4.10}$$

$$\implies \alpha_i = 1 - T_{\text{step}}/T_i = \frac{1+\exp(-s f_{i-1})}{1+\exp(-s \hat{f}_i)} \tag{4.11}$$

$$\implies \hat{f}_i = \frac{-1}{s} \log\left(\frac{1+\exp(-s f_{i-1})}{1-T_{\text{step}}/T_i} - 1\right) \tag{4.12}$$

We can then use a first-order approximation of the sdf to compute the step Δt

based on $\hat{f}_i$ in eq. (4.13).

$$f(\mathbf{X}+\Delta t\mathbf{V}) \approx f(\mathbf{X}) + \frac{\partial}{\partial t} f(\mathbf{X}+t\mathbf{V})|_{t=0}\Delta t \tag{4.13}$$

$$= f(\mathbf{X}) + \nabla f(\mathbf{X}) \cdot \mathbf{V}\Delta t \tag{4.14}$$

$$\implies \hat{f}_i = f_{i-1} + \nabla f(\mathbf{X}) \cdot \mathbf{V}\Delta t \tag{4.15}$$

$$= f_{i-1} + (\mathbf{N} \cdot \mathbf{V})\Delta t \tag{4.16}$$

$$\implies \Delta t = \frac{\hat{f}_i - f_{i-1}}{\mathbf{N} \cdot \mathbf{V}} \tag{4.17}$$

We further approximate $\mathbf{N} \cdot \mathbf{V}$ as $(f_{i-1} - f_{i-2})/\Delta t_{i-1}$ in the previous equation. Using this method, each ray will advance independently by a step tailored to the local behavior of the surface at the current sample point. In practice, we restrict $0.1 < \Delta t < 0.5$ in units of a voxel size. In case the ray travels through a back-facing surface, for which $\alpha_i = 1$, we take a step equal to half a voxel's side length.

4.3.6 Pre-Computed Color Decoding

Grid-based implementations [30] of the classic rendering model summarized in §4.3.1 typically compute each sample's color by querying an MLP with a tri-linearly interpolated feature vector:

$$\mathbf{C}_{\text{classic}} = render(MLP(lerp(grid))). \tag{4.18}$$

This allows off-grid queries to produce spatially-varying colors at scales smaller than a voxel [51], but implies that MLP queries are done per sample point during the ray-integration stage. [54] propose a deferred NeRF approach, hinting that the chain of operations can be re-ordered for other purposes:

$$\mathbf{C}_{\text{deferred}} = MLP(render(lerp(grid)). \tag{4.19}$$

Here, the MLP is queried only once per ray over the volume integral of the interpolated grid features, to decode a pixel color. This strategy compresses and accelerates the rendering of already trained NeRF representations, but the latter is what we intend to improve. Instead, we propose to query the MLP once per voxel for a given viewpoint, and render with the result, interpolated at ray-sample locations:

$$\mathbf{C}_{\text{pre-computed}} = render(lerp(MLP(grid))). \tag{4.20}$$

First, this reordering unlocks the ray-marching simplification of §4.3.5. Second, the MLP can now solely focus on the angular distribution of the colors while their spatial distribution becomes fully encoded into the grid, better decoupling the two responsibilities. Consequently, given the restrained BRDF space of human scenes, we can reduce the size of the MLP and its computational cost as long as the voxels are small enough to explain the spatial distribution of the colors, which our sparse representation is designed to achieve (§4.3.3). We empirically demonstrate the validity of this intuition by evaluating our

approach, using a 32-neuron MLP, against [30] and [31] that rely on the *classic* pipeline, in §4.4. More specifically, we parametrize the color prediction MLP as follows:

$$\mathbf{C}_{\text{voxel}} = MLP(concat(\mathbf{F}, \mathbf{N}, \mathbf{N}\cdot\mathbf{V}, SH(\mathbf{R}))) \tag{4.21}$$

with $\mathbf{F} = \mathbf{F}_x \otimes \mathbf{F}_y \otimes \mathbf{F}_z$ the color feature decoded from our sparse structure, $\mathbf{N} \overset{\triangle}{=} \nabla f / ||\nabla f||$ the voxel's normal vector obtained with finite differences, $\mathbf{V}$ the view vector, and $SH(\mathbf{R})$ a frequency encoding of the reflected vector in the form of a spherical harmonic projection. Decomposing $\mathbf{R}$ is more important than other MLP input directions here, as it coincides with specular reflection directions. The entire color prediction is fused in a single GPU kernel to reduce data transfers and memory usage incurred by separate computations (see fig. 4.2). The only temporary storage needed is for the predicted color itself. An equivalent kernel implements the backward pass.

4.3.7 Regularizations and Losses

We detail the losses used for optimization in SDF, feature and color space, to constrain smoothness and explain the multi-view inputs.

Feature smoothness is enforced directly on the feature planes with a TV-L2 loss. This promotes consistency between the neighboring features of different tiles : $\mathcal{L}_{\text{features}} = \sum_{d\in\{x,y,z\}} ||\nabla \mathbf{F}_d||^2$.

Spatial smoothness. Similarly to [30], we convolve the raw sdf $\tilde{f}$ with a 5^3 Gaussian kernel G to obtain a smoother SDF value $f = G(\tilde{f})$. Note that f is used for all subsequent operations. A regularization keeps f and $\tilde{f}$ close to each other: $\mathcal{L}_{\text{sdf}} = |f - \tilde{f}|^2$. The signed-distance property is promoted at each voxel by the Eikonal regularization $\mathcal{L}_{\text{eik}} = (||\nabla f|| - 1)^2$. The normal is also smoothed with $\mathcal{L}_{\text{normal}} = ||\nabla \mathbf{N}||^2$. Finally, we add a regularization on the predicted voxel colors $\mathcal{L}_{\text{colors}} = ||\mathbf{C}_{\text{voxel}} - G(\mathbf{C}_{\text{voxel}})||^2$, to avoid over-fitting to camera sensor noise.

Color correction. A significant practical issue arises with inter-camera exposure variations, which are difficult to globally control through camera hardware. We thus use a per-camera and per-channel color correction curve γ applied on the rendered colors $\mathbf{C}$. We parametrize $\gamma\colon [0,1] \longrightarrow [0,1]$ as a piece-wise linear function that we optimize along the rest of the parameters. We fix the end-points to be 0 and 1 and apply a smoothness regularization to ensure that γ stays monotonous.

Rendering loss. We use a standard photometric loss, additionally accounting for per-camera color correction and background images if available. Since binary masks are never pixel accurate due to labeling error and intrinsic intra-pixel blending at occlusion boundaries, we blend-in the background color $\mathbf{C}_{\text{bg}}$ weighed by the residual ray transmittance, allowing for accurate sub-pixel performance:

$$\mathbf{C}_{\text{predicted}} = \gamma(\mathbf{C}) + T\mathbf{C}_{\text{bg}} \tag{4.22}$$

$$\mathcal{L}_{\text{photo}} = ||\mathbf{C}_{\text{predicted}} - \mathbf{C}_{\text{in}}||_2^2, \tag{4.23}$$

with $\mathbf{C}_{\text{in}}$ the observed color, $\mathbf{C}$ the rendered color without background. Following [193], we apply a per-channel weight on the photometric gradient $\left(\min\left(\mathbf{C}_{\text{predicted}}, \mathbf{C}_{\text{in}}\right) + 0.005\right)^{-1}$ so that bright and dark colors are treated in the same way.

4.3.8 Neural Architecture

Our network architecture is a small MLP with 32 neurons and 2 hidden layers. We use features with a dimensionality of 8 in all experiments and a spherical harmonic embedding of the reflected vector with 16 coefficients. We use ReLU activations with a sigmoid in the last layer. For comparison, NeuS2 uses a 2 hidden layers MLP with 64 neurons and Voxurf uses a 4 hidden layers MLP with 192 neurons, requiring a much greater number of floating-point operations.

In terms of grid resolution, we use voxels with 2mm side length at the highest level of detail, which translates to about 10 million active voxels. In contrast, Voxurf uses 96^3 and 256^3 dense grids at the coarse and fine reconstruction stages respectively, corresponding to voxels of about 8mm at the fine stage assuming a bounding box of 2m. Note that Voxurf uses $256^3 \approx 16.8 \times 10^6$ voxels, slightly more than we do. NeuS2 use 14 hash grid levels spanning grids of size 16^3 up to 2048^3 which gives voxels of 1mm with the same bounding box of 2m. However, each hash grid level only contains $2^{19} \approx 500K$ entries, necessarily leading to a large number of collisions in the last level.

4.3.9 Output Model size

Our trained model size is roughly 100MB in total but about half of the allocated parameters are kept unused. This is the consequence of our sparse structure storing voxels in tiles of 4^3. We compress the model as zip archive and obtain a final file size of about 50MB. Linearizing the datastructure to remove unused voxels and features would be another way of achieving the same effect.

4.4 Experiments

We present the MVMannequin dataset, designed to characterize and compare metric reconstruction performance in human scenes. It includes 14 outfits on full-size mannequins, 6 male and 8 female, each requiring one hour for paired acquisitions using the [1] platform, as described below. We selected a diverse range of clothing outfits, featuring casual or formal clothing and shoes; long dresses, skirts, shorts, and pants; short-sleeved and long-sleeved tops, including jackets; and plain color versus textured (fig. 4.4).

4.4.1 Dataset Acquisition Protocol

To capture sub-millimeter accurate reference data for each of the 14 outfits we provide, we first perform the hand scanning then the multi-camera shot back

to back, to avoid any unwanted pose discrepancies. After this, we acquire background images and proceed with geometric calibrations of the camera set, which require shooting longer sequences of a calibration wand.

Scanning. We scan each instance using an Einscan HX hand scanner[2], in structured light mode with a manufacturer-specified 1mm resolution and 0.05mm accuracy. The scan was recorded in one take without need for sub-part repositioning. No filtering nor hole-filling was used for acquired points clouds. The resulting meshes of all mannequins are shown in fig. 4.4.



Figure 4.4: Hand-scanned mannequins. Female (top), male (bottom).

Multi-camera acquisition. We capture sequences of 10 frames for each outfit using 68 calibrated RGB cameras of 4 megapixel resolution, with focal lengths between 8 and 16mm. The cameras are positioned roughly on a half-ellipsoid with radii 4m and 5m and height 5m looking towards the stage center (fig. 4.5), for an average image resolution of 2.5mm per pixel at the scene center. The total capture surface covers a length of 5.5m and a width of 3.5m, representative of a significant human movement area.

4.4.2 Baselines

We evaluate the quality of our reconstructions against a set of state-of-the-art baselines, two neural Neus2 [31] and Voxurf [30], and two established multi-

[2] https://www.einscan.com/einscan-hx/

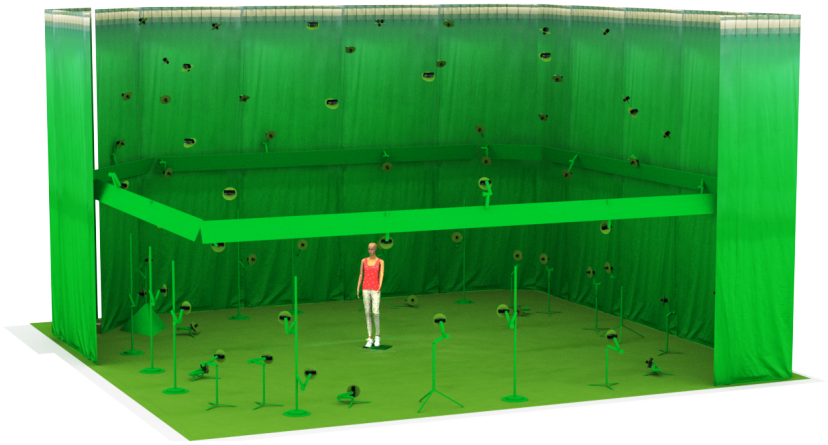

Figure 4.5: The Kinovis multi-camera platform used to capture the dataset.

Baselines	Ours	Neus2	Voxurf	RC	Colmap	3DGS	2DGS	GOF
Masks	both	✓	✓	✗	✗	✗	✗	✗
Full scene	✗	✗	✗	✓	✓	✓	✓	✓
Train	2min	5min	15min	5min	>1h	45min	1h20	3h30
VRAM	5GB	5GB	30GB	-	-	30GB	45GB	20GB
Framerate	∼250	∼15	<1	-	-	>300	>300	>300

Table 4.1: Indicative baselines characteristics

view stereo implementations, Colmap [25] and RealityCapture [14] (RC for short). We also evaluate against three splatting-based methods, the original 3D Gaussian splatting (3DGS) [27], and its recent surface-capable variants 2D Gaussian splatting (2DGS) [48] and Gaussian Opacity Fields (GOF) [49]. A summary of each baseline's characteristics is given in table 4.1.

Dataset preparation. First, we temporally average then undistort the images and the backgrounds. We compute masked images using the backgrounds, as input to all baselines that require them (table 4.1, first row). All baselines use the same calibration except for RealityCapture since there was no simple way to provide one. The sparse point cloud generated by Colmap is given to the splatting-based methods, as required. We disable Colmap's refining of camera intrinsics and extrinsics for fairness with the other baselines.

Geometric evaluation. We clip both the 3D reference and 3D reconstructions at a fixed height to remove the ground. A per-baseline robust ICP [194] accounts for the global registration bias before computing two-way point-to-mesh distances, less sensitive to mesh discretizations than point-to-point distances.

Photometric evaluation. The PSNR of the rendered images is computed within the reprojection of the scan, using the average ICP rigid transform of the best two baselines (Ours and Voxurf). The reprojection is eroded by two pixels since the silhouettes' borders are less accurate for the baselines relying

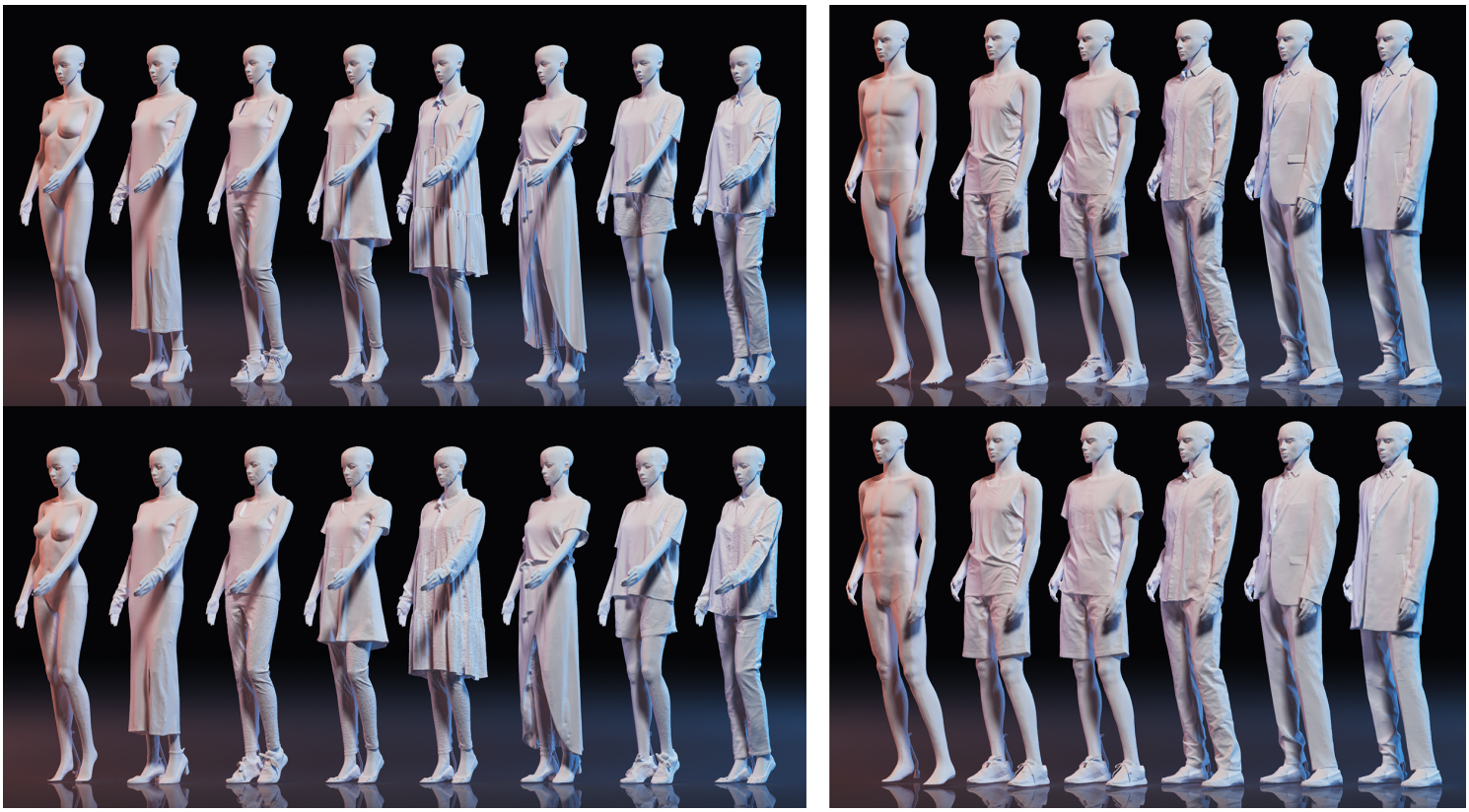

Figure 4.6: Reference scans (top rows) versus our reconstructions (bottom rows) for our 8 female (left) and 6 male (right) mannequins dataset

on masks.

4.5 Results

4.5.1 Quantitative results

We plot accuracy and completeness curves in fig. 4.7, showing the proportion of the vertices having a euclidean distance to the reference that is less than $x$mm. We observe that our proposed method can reconstruct 77% of points with accuracy less than 1mm, while the second best method achieves about 60%, a 28% gain against SOTA. These results show the strong ability of our method to reconstruct geometric details with sub-millimeter accuracy and few outliers, with only 7% of points above 3mm distance from reference scan. RealityCapture and Colmap achieve an acceptable performance in well-textured areas, but tend to fail in uniform or specular regions of the surface, with less detail overall. Aggregate metrics are shown in table 4.2 (top). Both our accuracy and completeness are below 1.2 mm with a PSNR slightly better than Voxurf's. 3DGS obtains a PSNR equivalent to ours thanks to its accurate specularities, but tends to aggregate many anisotropic Gaussians to explain high frequency texture patterns. This severely impacts the geometry of GOF and to a lesser extent 2DGS in those areas, because the Gaussians lack a cohesive orientation.

Metrics	Ours	Voxurf	Neus2	RealityCapture	Colmap	3DGS	2DGS	GOF
Completeness (mm)	**1.16**	1.50	2.51	4.90	3.67	-	2.29	3.27
Accuracy (mm)	**1.13**	1.68	1.97	3.64	3.36	-	4.41	2.81
PSNR (dB)	**36.33**	35.51	30.80	-	-	36.12	34.89	34.05

Metrics	r/2	r/4	r/8	w/ masks	w/o color correct.	w/o color smoothing
Completeness (mm)	1.63	1.84	6.97	1.68	2.56	1.20
Accuracy (mm)	1.44	1.75	5.11	1.54	2.32	1.14
PSNR (dB)	33.20	31.21	26.34	36.14	32.34	36.41

Table 4.2: Top: Comparisons, Bottom: Ablations. Completeness , accuracy and PSNR averaged over all scans. $r/2$, $r/4$ and $r/8$ column denote training with half, quarter and one eighth of the grid and images resolutions respectively.

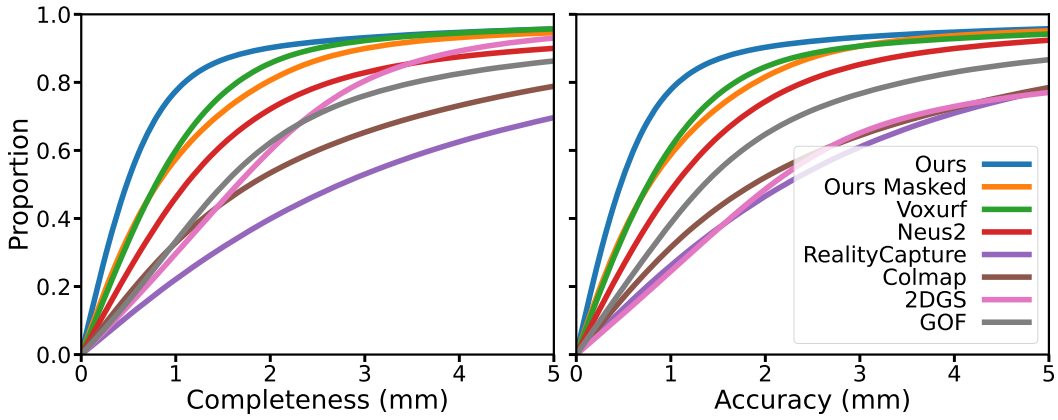

Figure 4.7: Completeness and accuracy error curves

4.5.2 Detailed quantitative results

Tables 4.3, 4.4 and 4.5 provide complete metrics for the chosen baselines on all the scenes of our dataset.

4.5.3 Volume rendering comparisons

Figures 4.9 and 4.10 show the volume rendering results on two scenes, corresponding to the first two columns of the tables 4.3, 4.4 and 4.5. Qualitative comparisons on the rest of the dataset are available in appendix B.3.

4.5.4 Error visualization

Fig. 4.8 provides a visual rendition of the error distribution on the surface, which is globally lower in our case, although Voxurf is able to better carve some of the concave regions. NeuS2 exhibits high frequency noise while Colmap is particularly sensitive to specularities on the arms, legs and head of the mannequins. GOF fails to extract a well-defined surface in the high texture

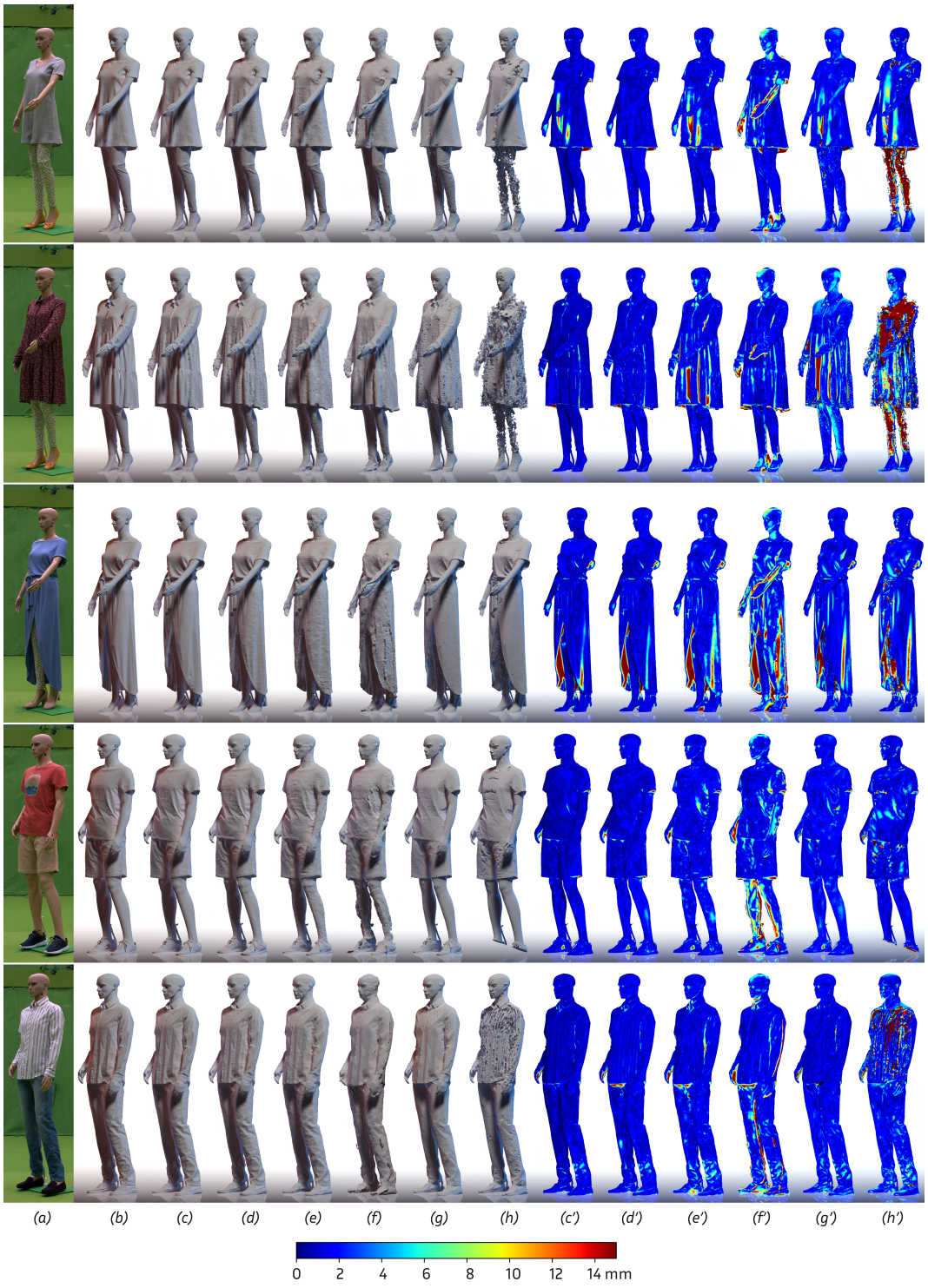

Figure 4.8: Reconstructions on a subset of our proposed dataset with (a) input image, (b) reference scan, (c) our method, (d) Voxurf, (e) NeuS2, (f) Colmap, (g) 2DGS, (h) GOF and their associated accuracy heatmaps.

	female								male					
Completeness (mm)	cos	naked	jea	opt1	opt2	opt3	sho	tig	cos	naked	jea	opt	sho	tig
Ours	1.69	**0.53**	**1.22**	**1.14**	2.16	1.45	**0.98**	**0.61**	**1.99**	**0.50**	**0.82**	**1.76**	**0.73**	**0.70**
Ours Masked	2.05	1.50	1.62	1.33	2.23	1.46	1.41	0.88	3.54	1.52	1.06	2.55	1.14	1.17
Voxurf	**1.44**	1.27	1.63	1.39	**1.89**	**1.35**	1.27	0.89	2.57	1.21	1.25	2.79	1.05	1.00
Neus2	5.18	1.47	2.59	2.47	3.44	2.41	2.22	1.28	4.48	1.48	1.89	3.06	1.66	1.50
RealityCapture	3.74	6.16	6.07	6.47	5.77	3.47	4.68	3.94	7.04	4.89	2.49	5.57	3.83	4.47
Colmap	3.25	3.95	3.34	5.17	4.54	2.97	3.63	2.24	4.68	3.69	2.36	5.54	2.99	3.03
2DGS	3.82	2.12	2.63	2.13	2.55	2.35	2.02	1.56	3.69	2.09	1.38	2.39	1.72	1.64
GOF	5.86	1.45	7.01	2.29	2.88	4.34	2.47	4.14	2.74	1.42	2.24	2.53	4.77	1.61

Table 4.3: Per-scene average completeness (mm)

	female								male					
Accuracy (mm)	cos	naked	jea	opt1	opt2	opt3	sho	tig	cos	naked	jea	opt	sho	tig
Ours	**1.41**	**0.55**	**0.99**	**1.17**	2.08	**1.45**	**1.17**	**0.75**	**1.55**	**0.68**	**0.79**	**1.67**	**0.78**	**0.80**
Ours Masked	1.53	1.47	1.19	1.31	2.11	1.57	1.50	0.97	2.86	1.66	0.97	2.13	1.14	1.21
Voxurf	1.96	1.29	1.61	1.84	2.24	1.87	1.68	1.08	2.62	1.39	1.29	2.29	1.20	1.12
Neus2	2.98	1.44	1.74	2.11	2.68	2.14	1.97	1.23	2.85	1.63	1.49	2.29	1.53	1.44
RealityCapture	1.93	5.03	4.83	4.86	4.07	2.33	3.77	3.31	4.51	4.36	1.98	3.05	3.15	3.72
Colmap	2.12	4.59	2.65	4.01	3.83	2.65	3.78	2.77	3.54	4.72	2.60	3.14	3.18	3.43
2DGS	6.29	2.42	13.98	4.37	4.25	4.90	3.31	5.47	4.67	2.40	2.12	3.21	2.23	2.10
GOF	5.32	1.30	3.10	1.88	**2.05**	4.40	1.86	5.44	1.99	1.47	4.68	2.39	1.67	1.77

Table 4.4: Per-scene average accuracy (mm)

	female								male					
PSNR (db)	cos	naked	jea	opt1	opt2	opt3	sho	tig	cos	naked	jea	opt	sho	tig
Ours	29.24	**40.03**	**30.55**	**39.94**	38.01	**33.82**	37.18	**31.38**	**39.22**	**40.88**	35.25	38.59	37.11	37.49
Ours Masked	**29.31**	39.6	30.54	39.63	37.8	33.69	37.0	31.29	39.03	40.5	35.1	38.3	36.9	37.3
Voxurf	28.4	37.5	30.2	39.3	37.3	32.8	36.7	30.8	38.2	40.1	34.5	37.6	36.7	37.2
Neus2	27.6	34.0	28.6	31.9	33.2	30.3	31.6	28.9	31.3	33.2	29.7	25.8	32.4	32.7
3DGS	28.1	39.91	29.7	39.4	**38.28**	33.2	**37.42**	30.6	38.5	40.62	**36.10**	**38.60**	**37.47**	**37.79**
2DGS	27.3	37.9	28.9	37.6	36.8	32.2	36.2	29.9	38.0	38.8	35.0	37.3	36.1	36.5
GOF	27.1	37.2	28.1	37.1	36.3	31.5	35.3	28.8	37.2	38.5	33.4	35.9	35.3	34.9

Table 4.5: Per-scene average PSNR



Figure 4.9: Qualitative rendering comparisons on the female "cos" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. The Gaussian splatting methods achieve a good image quality overall but tend to fit highly anisotropic Gaussians in textured regions, which impacts the extracted geometry in 2DGS and GOF's cases. Neus2 suffers from noise while Voxurf creates sharp boundaries at depth discontinuities. Our method reconstructs both smooth edges that accurately blend with the background image and detailed textured patterns.

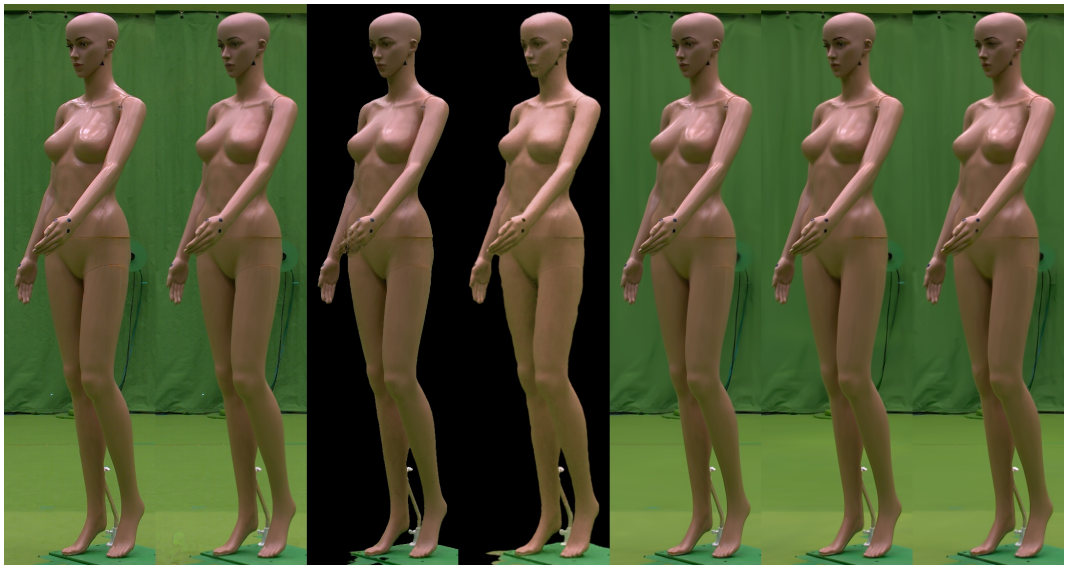

Figure 4.10: Qualitative rendering comparisons on the female "naked" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.

frequency regions but achieves good accuracy otherwise, although noisy. All our reconstructions are shown in fig. 4.6.

4.5.5 Ablations

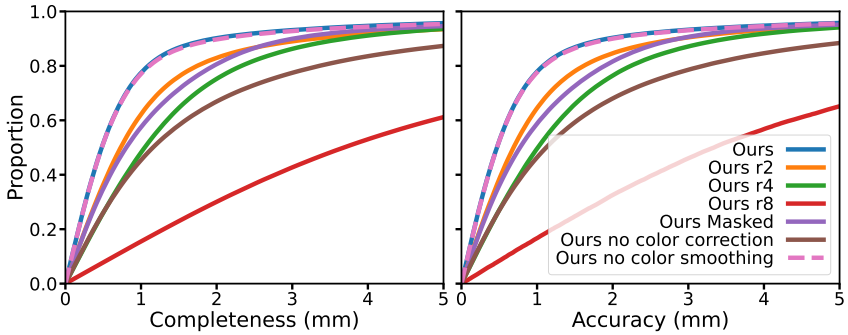

Figure 4.11: Completeness and accuracy error curves for the ablations.



Figure 4.12: Our reconstruction without the color smoothness regularizer (left) and with (right). The region around the eyes exhibits a noticeable pixelizing.

We ablate the quality improvement per coarse-to-fine detail level, training with masked images instead of full background images, exposure correction, and color smoothness in table 4.2 (bottom), with corresponding error curves in fig. 4.11. The color smoothness term avoids over-fitting to the sensor noise with a slightly decreased PSNR but a higher perceived quality (see fig. 4.12 for qualitative comparison). Color correction is important both for convergence of the geometry and rendering quality, explaining the inter-camera color variations. Training with background images improves metric performance by 30% over binary masks, as it allows modeling intra-pixel occlusion blending and avoids inherent inaccuracies of the masks. Our method is on par with Voxurf in terms of geometric quality with masked images, but with a higher PSNR, and outperforms Voxurf when background images are used. Each subsequent level of refinement (denoted as r/8, r/4, r/2 and Ours) is trained with images at twice the resolution and with voxels half the size, showing clear improvements in quality.

4.5.6 Qualitative results on moving subjects



Figure 4.13: Results of our reconstructions on some sequences of the 4DHumanOutfit dataset [9]

We provide results on a representative set of motion sequences from the publicly available 4DHumanOutfit dataset [9]. The selected sequences exhibit running, boxing, crouching, jumping, dancing and a cartwheel motion, all covering several meters and varied poses. Fig. 4.13 and fig. 4.1 illustrate the reconstructions obtained from these sequences. Animated reconstructions are presented in the accompanying video, available on the project webpage: `https://projects-morpheo.gitlabpages.inria.fr/millimetrichumans/`. Note the abundance of surface detail on the faces, hair in motion, and clothing. Folds and creases are accurately reconstructed except in some concavities with uniform and ambiguous color. Notably, the micro-geometry of clothing seams is also captured in our reconstructions, as can be seen on various pants worn by the actors, and the noise levels on the surfaces are low enough for these details to be consistently represented frame after frame, despite the fact that our method does not apply any temporal smoothness, or estimate any temporal surface alignment.

4.5.7 Exposure variations and sensor noise

The acquisition platform used mixes large and medium camera fields of view leading to changes in exposure, as visible in fig. 4.14. Figure 4.15 shows close-ups on the sensor noise and Bayer filter artifacts in the input images.

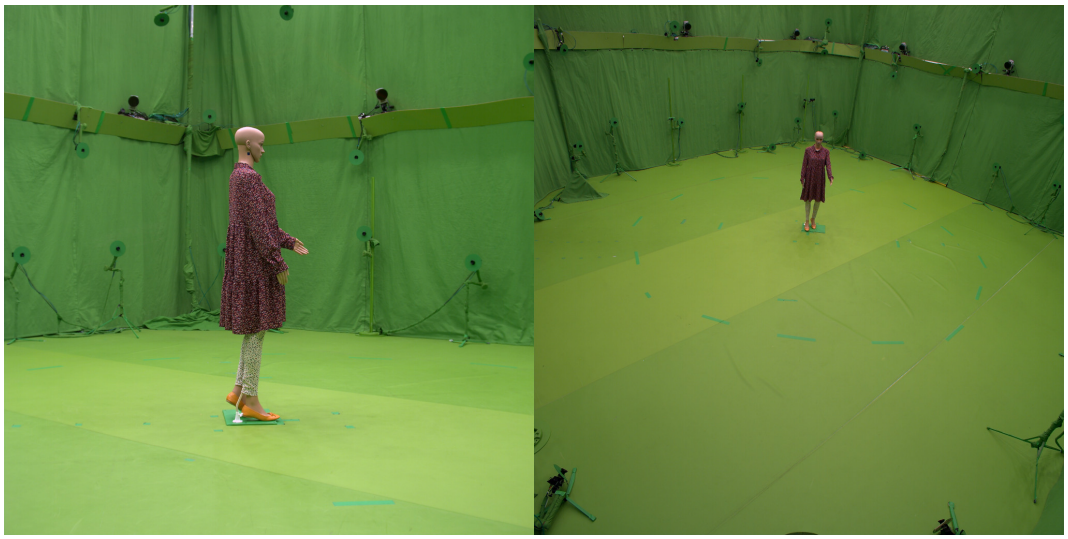

Figure 4.14: Field of view differences imply exposure variations between the cameras.



Figure 4.15: Crop of the region around the head of the mannequin in figure 4.14 (right). The camera sensors display some level of noise (left) that can be alleviated by temporal averaging (right) but color fringing and staircase artifacts remain due to the Bayer filter.

4.5.8 Additional experimental details

We run all experiments on a linux workstation equipped with an Nvidia RTX A6000, unless stated otherwise. We use all images available both at training and test time. In the following, we provide additional details specific to each method used for the comparison in section 4.4. The resulting mesh of the baselines reconstructing a complete scene (Colmap, RealityCapture, 2DGS and GOF) is cropped with a bounding box before evaluation.

Ours We fit a bounding box on the complete acquisition volume and compute a coarse visual hull to find the region of interest and initialize the volume. Visual hull computation and initialization takes less than 1 second.

Neus2 Neus2 provides two means of accounting for varying camera exposure: an extra latent code and an exposure parameter that are trained on a per-camera basis. We enable both in an attempt to match our optimization of a per-camera color curve. We train Neus2 for 7500 iterations using the DTU configuration. We translate and scale the cameras to place the scene to be reconstructed inside the unit ball.

Voxurf We use the DTU configuration files with masks setting. Voxurf starts training with a coarse initialization based on DVGO. They extract a bounding box from this reconstruction before running their coarse and fine optimization schemes. We use the default calibration since Voxurf can automatically find the region of interest.

Colmap We run colmap using the cameras' calibration and disable the optimization of the cameras' parameters. We run patchmatch stereo on full size images using the geometric consistency setting. The depthmaps are fused in a dense point cloud then meshed using the Poisson mesher.

RealityCapture We run RealityCapture on a windows laptop since linux is not supported. We let it estimate the calibration data since we could not find a simple way to provide calibrated inputs.

Gaussian splatting methods The sparse point cloud resulting from the triangulation of the features detected by colmap is given to 3DGS, GOF and 2DGS. We train them on full size 2048^2 images instead of a re-scaled version that is used by default when the images exceed a width of 1600 pixels. We train for the default 30k iterations. We supply the "decoupled appearance" option to GOF since our images contain inter-camera color variations that are not specifically handled for the other two. We train 2DGS with the median depth setting instead of the average depth since it otherwise fails to extract a unique surface with its TSDF fusion scheme. The data loader of 3DGS (and inherited by 2DGS and GOF) silently discards the principal point from the calibration. We changed the code of the three baselines to take into account our non-centered cameras. This only amounts to changing the projection matrix for 3DGS and 2DGS as noted by other contributors but these changes are not part of the main branches yet. After discussion with the authors, the principal point must additionally be forwarded to several CUDA kernels in GOF's case. We opened a pull request so that others may benefit from this improvement. We observe that 3DGS converges reliably whereas 2DGS and GOF have more trouble in some instances (see fig. 4.16) in which a few Gaussians occlude the scene from one or several viewpoints, lowering the average PSNR. In terms of



Figure 4.16: Convergence failures for Gaussian Opacity Fields (left) due to Gaussians near the cameras occluding the scene, and for 2D Gaussian splatting (right) due to Gaussians above the ground occluding the feet.

geometric quality, 2DGS creates noisy polygons attached to the inside of the outer surface in the highly textured regions, as visible in fig. 4.17 (Top). These polygons are not removed when we select the main connected component since they are attached. GOF extracts a mesh on an irregular tetrahedral grid which limits the impact on the accuracy metric due to the larger triangles in the noisy areas, even though it can fail more catastrophically. We also demonstrate a smoother and more detailed surface than both 2DGS and GOF on an example with low frequency texture (See fig. 4.17 (Bottom)). GOF also fails to extract the feet in this specific instance.

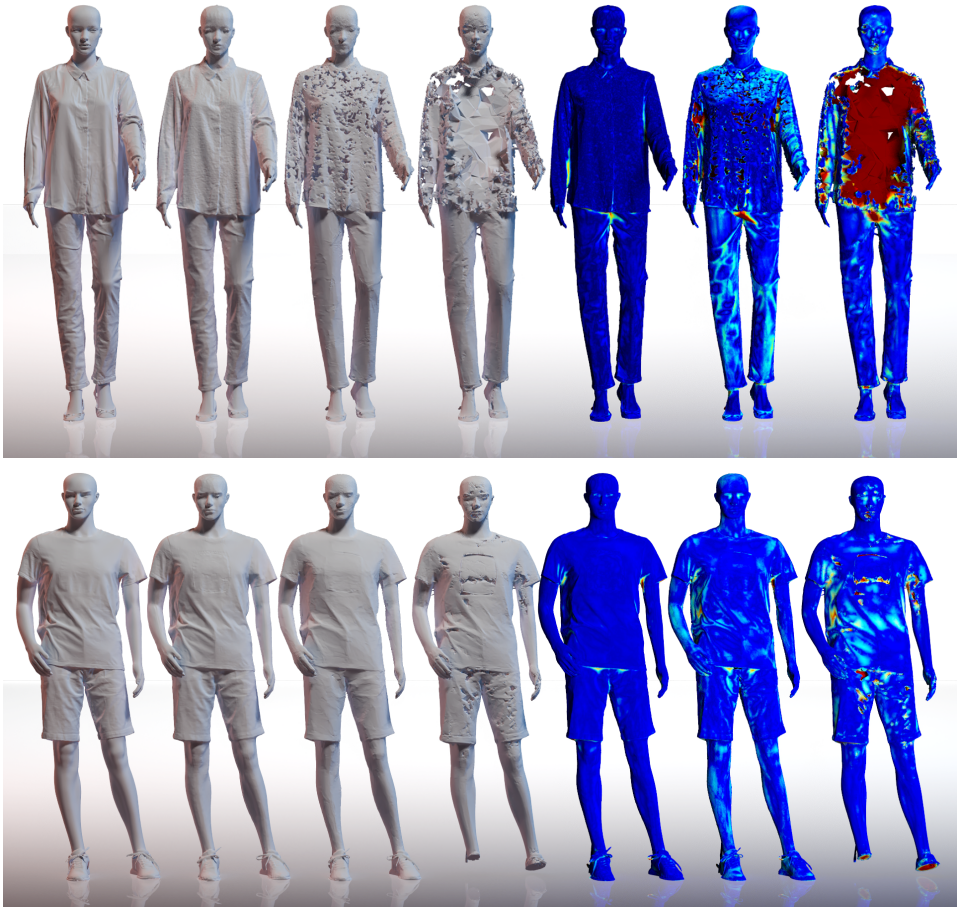

Figure 4.17: Top: Female "jea" scene, bottom: Male "sho" scene. From left to right: Reference scan, Our reconstruction, 2DGS, GOF and their associated accuracy heatmaps. Top: Highly textured areas are not well reconstructed both for 2DGS and GOF, leading to noise and holes. Bottom: We achieve better or equivalent geometric quality than 2DGS and GOF even in regions with less texture.

4.6 Conclusion

This chapter explored how the hybrid surface / volume differential rendering methodology can be improved for performance capture purposes. We showed that storing explicit signed-distance values and neural appearance features at a high resolution gives a definite advantage in quality in this context, over previous works advocating for low resolution embeddings. Despite the higher resolution, our approach converges quickly and with a small memory footprint thanks to a dynamic sparse representation and a re-designed training pipeline. The scene can be volumetrically rendered at several hundreds of frames per second out-of-the-box, with a model size of about 50MB. Trading features of lower dimensionality for a higher spatial density proved successful, and allows to further simplify the neural appearance component in this setting. However, our small MLP can be insufficient to reconstruct strong specularities, which leads to geometry artifacts such as planar polygonalizations on the mannequin arms, and may hamper performance on more general scenes. The next chapter will precisely address these concerns, by investigating how to introduce more expressivity without degrading performance, and evaluating our ideas on more complex and varied scenes. In particular, the low memory consumption hints that our sparse framework has not been pushed to its full extent, and could be applied to input images of much higher resolution.

Compared to the Gaussian splatting baselines, it can be argued that our training is faster because we do not reconstruct the full scene. Modifying these approaches so that they can take masks or backgrounds as input would bring a final answer to this matter, but we leave it as future work. Regardless of training speed, we find that such unstructured representations can struggle to extract a smooth surface in highly textured regions but remain very capable for novel view synthesis.

Our proposed dataset allowed us to perform a fair evaluation of all these approaches on real images representative of an actual person, without incorporating biases due to widely differing materials, camera arrangement or sub-par / misaligned reference geometry. Nonetheless, this dataset is not a perfect substitute for real subjects due to the notable absence of hair, skin and motion. We leave as future work the task of providing a dataset incorporating such elements, which seems highly challenging but could also be very useful.

Finally, another key takeaway is that leveraging background images is much more effective than relying on a binary masks, as the segmentation problem is made continuous, differentiable and consistent across viewpoints. None of the other tested baselines includes this feature, which makes them either less accurate than they could be, or wasteful when reconstructing the whole scene. Unfortunately, many human datasets [3,52] do not provide background images since it is assumed the background segmentation occurs *before* the reconstruction. We hope this study will convince interested researchers of leveraging or providing background images for volumetric capture.

4.7 Acknowledgements

This work was supported by French government funding managed by the National Research Agency under the Investments for the Future program (PIA) grant ANR-21-ESRE-0030 (CONTINUUM). This work was partly supported by JSPS KAKENHI Grant Number JP23H03439. We thank Julien Pansiot for his contribution to the acquisition process and Kinovis platform management.

5

Light Field Probes for Neural Surface Reconstruction

5.1 Introduction

Neural radiance fields have established a new milestone for the task of novel view synthesis and have led to a plethora of variants of broad applicability. The previous chapter focused on performance capture, where we explored a decorrelated representation of shape and appearance that provided significant efficiency gains. We now explore a factorized representation of appearance into its angular and spatial components, addressing the aforementioned limitations with regard to complex materials. We also extend our study beyond human scenes, targeting the larger goal of reconstructing generic objects accurately and quickly.

A core idea of chapter 4 is to use a smaller color prediction neural network than concurrent approaches [10, 30, 31], which turns out to be possible thanks to our decorrelated representation, but also due to the controlled lighting conditions of multi-camera setups and limited specularity of clothes and skin. Compounding speed improvements are unlocked by this observation: the entire color prediction can be fused in a single GPU kernel, both during the forward and backward passes, which also drastically reduces memory usage. Several other optimizations hinge on a cheap color prediction, *e.g.* rendering a complete image at once becomes feasible thanks to its negligible memory cost, which accelerates convergence compared to small batches of rays.

Unfortunately, this revamped pipeline tends to fail when confronted with a more complex view-dependent appearance that cannot be explained by the small MLP. Switching to a bigger network may prevent fusing the backward pass in a single GPU kernel due to hardware constraints. This would require storing intermediate activations in large tensors that are memory-intensive: we would end up with the classic training strategy with small batches of rays.

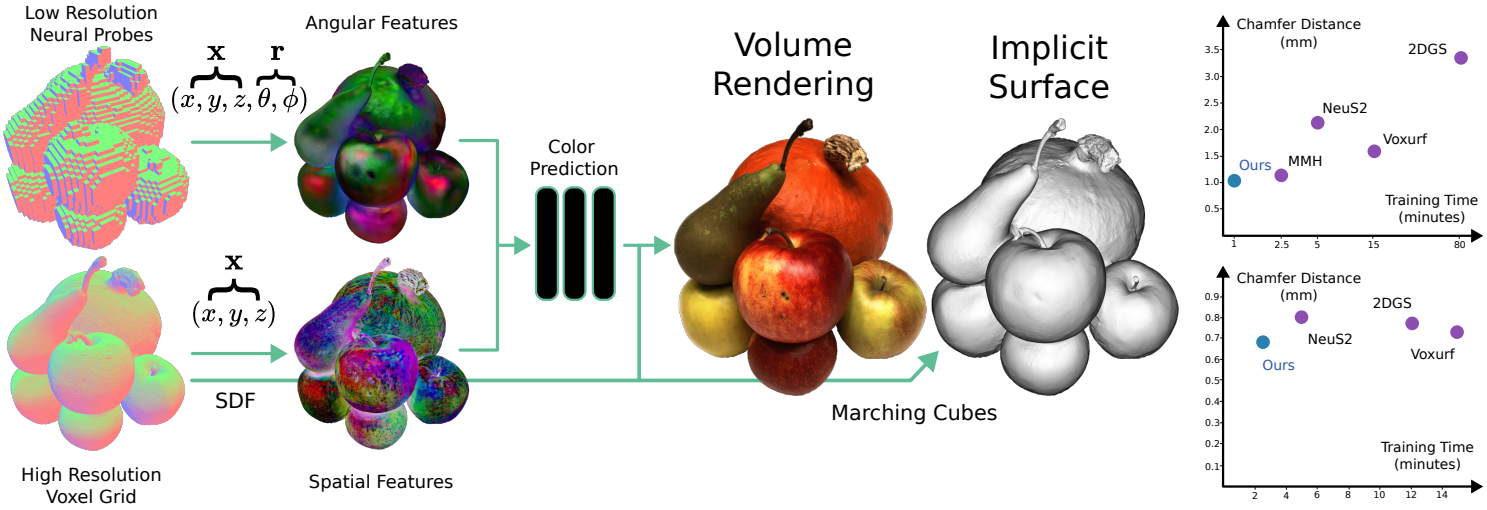

Figure 5.1: We design a new appearance model for neural surface approaches, which combines high resolution spatial features and lower resolution angular features for improved reconstruction quality, training and inference speed. We plot the Chamfer distance as a function of training speed for several baselines on MVMannequins (top) and DTU (bottom).

Going back to a naive spherical harmonics basis, as used in chapter 3, is not satisfactory either due to the large parameter count incurred at high resolutions. Thus, we seek a more expressive appearance model that still fits in the revised pipeline of chapter 4, in order to generalize to a broader class of materials and illumination conditions.

Rephrasing the problem, it amounts to parametrizing the light field efficiently for SDF surface-based radiance fields. This 5-dimensional function maps the spatial coordinates of a point in space and the view direction to a color and signed distance to the reconstructed surface. On one hand, the mapping can be centralized through a global scene-wide MLP decoder that is entirely responsible for the production of view-dependent colors [10, 26]. Accelerated variants [30, 31] decrease the reliance on the neural component with a grid of features, by extracting most of the spatial dependency. One the other hand, the mapping can be fully decentralized: *e.g.* 3D Gaussian splatting [27] uses a local representation such that each separate Gaussian comes with its own opacity and adjoining set of angular color features. Our idea is to strike a balance between these two extremes.

We postulate that, contrary to the spatial components of the light field that mainly encode intrinsic surface texture, the angular components encode phenomena that are predominantly extrinsic to the surface, *i.e.* lighting and environment, and as such can be encoded at a lower spatial resolution as they are locally almost invariant to parallax. This hypothesis is in fact exploited by the rendering community, where angular and environment components contributing to the radiance equation used to compute rendered colors, are typically precomputed at sparse 3D locations coined *light field probes* [195], and interpolated in-between.

Leveraging this intuition, our approach borrows the spherical harmonic parametrization popular with explicit methods *e.g.* [27, 96] to represent angu-

lar components, but instead decorrelates high density spatial variation from lower density angular variation using a mid-resolution probe grid of angular features. Both the spatial and angular varying components encode abstract features that are fed to a tiny MLP, which acts as a minimal BRDF decoder, while we explicitly store SDF values. This scheme advantageously replaces the core appearance and decoding model of SDF approaches [10, 29–31] to predict ray colors. We are careful to also include power terms of the cosine angle between the view direction and the estimated surface normal, allowing our lightweight model to effectively account for increased surface reflectivity at grazing angles. This yields an unprecedented performance combination, superseding both 3D and image metric performances with four popular benchmarks for general object reconstruction [34, 166] and human reconstruction [3, 12] while simultaneously achieving a significant training and rendering performance boost. In summary, our contributions are:

- An explicit embedding of the angular dependency that is simultaneously local, smooth and efficient.
- A simpler color prediction MLP that is agnostic to the surface position and orientation.

5.2 Related Works

In this chapter, we leverage disentangled spatial and angular parameterization of radiance towards high quality appearance and surface modeling. We refer the reader to §2.4 and §2.5 for a general analysis of related 3D scene representations and existing parametrizations of color respectively. We now take a closer look at works directly relevant to the contributions of this chapter.

Angular parametrization Deep MLPs have become ubiquitous to model view-dependent colors, taking as input the view direction [26], the view and normal [10, 30, 31, 45] or the reflected direction [12, 131]. Non-neural approaches such as plenoxels [96] or the Gaussian splatting methods [27, 48] favor a straightforward spherical harmonics decomposition in color space. The main advantage compared to a deep MLP is the computational efficiency, at the cost of a large number of appearance parameters: 27 per voxel for [96] and 48 per Gaussian for [27]. We propose a lighter yet efficient parametrization of color by using two decorrelated voxel structures to encode different components of the observed color, and spherical harmonics encoding of *pre-MLP features* for more parsimonious and expressive color decoding.

In the field of real-time rendering, where lighting and BRDF are known in advance, the idea of caching the incoming light at discrete locations, known as light field probes, has become standard [195–197]. We draw inspiration from this technique for our proposed approach to multi-view 3D reconstruction, with the difference in our case that the probe parameters are optimized along with the voxel parameters.

High Frequency Reflections. A number of approaches specifically deal with high frequency reflections, which requires dedicated representations and algorithms. Guo et al. [198] train an additional NeRF that is composited

with the main image. This approach is under-constrained and thus requires strong priors such as the reflective surfaces being flat. Verbin et al. [131] model lighting as a neural envmap, modulated by a per-point roughness. The roughness and base color of the object can be edited afterwards but a long training time is required due to the large MLP.

Recent works [95, 199] recover detailed reflections using cone tracing, which inherently solves the problem posed by self-reflections. Both approaches give impressive results even when confronted with curved, mirror-like objects but at a large training cost exceeding 10 GPU-hours per scene. In contrast, we target objects with mild specularities, and show that both fast training and real-time rendering can be achieved. More specifically, our neural light field probes can learn information about the surroundings whereas cone tracing gathers it at a greater cost. This assumption proves to be valid for standard acquisition scenes, with the four datasets tested.

Minimal Parametrization. Explicitly disentangling the material parameters from the lighting is particularly challenging but also rewarding because it can yield a minimal parametrization of the appearance. Munkberg et al. [93] show that the approximate rendering equation developed in [200] is sufficiently accurate to recover the geometry using a differentiable rasterizer, along with an approximate material decomposition. Jiang et al. [201] applies a similar strategy but with the 3D Gaussian splatting volume rendering. The NeRFactor pipeline [202] starts from a pre-trained NeRF that is distilled into lighting, albedo and neural BRDF parameters. Jin et al. proposed TensoIR [203], which achieves a more accurate decomposition by evaluating the rendering equation with importance sampling. An efficient tensor representation helps to accelerate the rendering of the secondary rays, but the complete training still takes five hours on a single GPU, with decent results available after half an hour of training. Our proposed method converges in just a few minutes and we find that only 4 coefficients per voxel are sufficient in most cases. The dimensionality is equivalent to the albedo and roughness values of a physically-based material parametrization, but without the cost associated with the inversion of the rendering equation.

5.3 Method

We use an SDF grid as the implicit representation of the 3D shape. SDF values in the grid are optimized via our newly proposed differentiable appearance model and the derived surface normal vectors $\mathbf{n}$.

5.3.1 Decoupled Parametrization of Radiance

Volumetric neural rendering, as proposed in the seminal work NeRF, computes the color at a given pixel by integrating radiances along camera-pixel rays weighted by local opacity values. Follow-up papers condition the opacity on the distance to the surface to improve the reconstruction quality [10, 128, 204]. The radiance C represents the amount of energy that is emitted by a point in space. In physics it is expressed as the integral over the hemisphere Ω of the

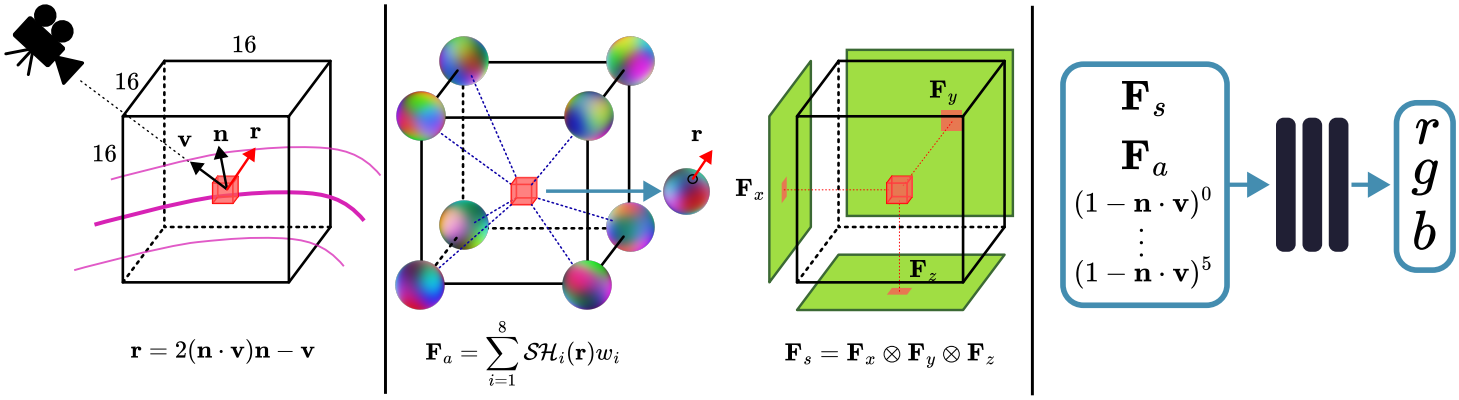

Figure 5.2: Overview of the color prediction for a single voxel inside a 16^3 tile. The angular features F_a are computed by interpolating and evaluating the spherical harmonics from the 8 corner probes with the reflected vector $\mathbf{r}$. The spatial features F_s are computed as the outer product of three orthogonal planes of resolution 16×16, specific to the tile. A small neural network decodes these inputs into a color.

spatially varying bidirectional reflectance distribution function (SVBRDF) f_r multiplied by the incoming radiance L and by the cosine between the incoming direction ω and the surface normal $\mathbf{n}$:

$$C = \int_{\Omega} f_r(\mathbf{v}, \omega, \mathbf{x}) L(\omega, \mathbf{x})(\omega \cdot \mathbf{n}) d\omega \tag{5.1}$$

The (spatially varying) BRDF depends on the point's position $\mathbf{x}$, direction of incoming light ω and viewing direction $\mathbf{v}$. $L(\omega, \mathbf{x})$ is the intensity of light at point $\mathbf{x}$ that comes from the direction ω and $\mathbf{n}$ is the normal vector of the surface at position $\mathbf{x}$. In equation 5.1, both f_r and L are unknown and disentangling these two functions during optimization is extremely difficult. As a consequence, a current research trend is to use an MLP to directly approximate the *result* of this integral and obtain the outgoing radiance with the following equation:

$$C = MLP(\mathbf{F}_s(\mathbf{x}), \mathbf{v}, \mathbf{n}, \mathbf{r}) \tag{5.2}$$

where $\mathbf{F}_s(\mathbf{x})$ represents spatial features encoded at a point position $\mathbf{x}$, and $\mathbf{r}$ is the reflected vector at point $\mathbf{x}$ for the viewing direction $\mathbf{v}$. This approximation works well if the incoming light is positionally invariant, but its accuracy degrades in the presence of local lighting effects such as close point lights, inter-reflections and self-shadows. In such cases, the spatial features must additionally encode some information about the local lighting and thus require a larger dimensionality than one would expect. A large and deep MLP is also necessary to decode this compressed information, which lowers the computational efficiency.

We propose to decouple the radiance model into spatial and angular features. We model a 3D scene with a combination of a high resolution sparse voxel grid that models the SDF field and a coarser grid that models the angular features as shown in fig. 5.2. For each point in space, the spatial features are obtained with a planar factorization in the coarse resolution tile with

$\mathbf{F}_s = \mathbf{F}_x \otimes \mathbf{F}_y \otimes \mathbf{F}_z$ for dimensionality reduction [60]. The angular features $\mathbf{F}_a$ are obtained by tri-linearly interpolating light field probes as explained in sec. 5.3.2. In addition, a dependency on the angle of incidence must be added to account for differences in reflectivity, as shown in fig. 5.3.

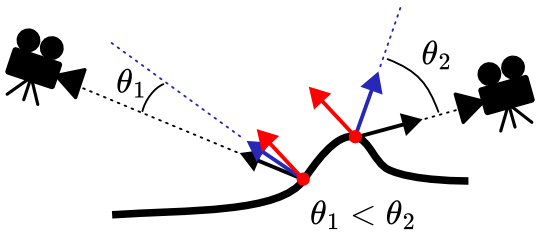

Figure 5.3: The reflectivity cannot be encoded in $\mathbf{F}_a$ alone because different incident vectors can map to the same reflected vector (in red). In that case, similar angular features will be obtained for the two viewpoints due to the spatial and angular proximity of the lookups. The angle of incidence disambiguates the two situations.

The reflectivity, also called the Fresnel term, is commonly approximated as follows for rendering purposes [205]:

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5 \tag{5.3}$$

R_0 is the reflectivity at normal incidence and θ is the angle between the view direction and the half-way vector to the light source. In our case, R_0 is unknown and there is no explicit light source which means that we cannot use this formula directly. Instead, we approximate $\cos\theta$ by $\mathbf{n} \cdot \mathbf{v}$ and let the first layer of the neural network learn a polynomial approximation of the Fresnel term. It should also be noted that eq. (5.3) is only valid in a linear color space, but we work in the non-linear sRGB color space, so it makes sense to learn a distinct approximation. Our revised parametrization is given by eq. 5.4:

$$C = MLP(\mathbf{F}_s(\mathbf{x}), \mathbf{F}_a(\mathbf{x}, \mathbf{r}), (1 - \mathbf{n} \cdot \mathbf{v})^0, \ldots, (1 - \mathbf{n} \cdot \mathbf{v})^5) \tag{5.4}$$

We denote the dimensionality of the features by n_s and n_a such that $\mathbf{F}_s \in \mathbb{R}^{n_s}$ and $\mathbf{F}_a \in \mathbb{R}^{n_a}$. Most materials reflect the light in a cone around the reflected vector which is why we condition $\mathbf{F}_a$ on $\mathbf{r}$, but some materials have retro-reflective properties that break this assumption. In that case, conditioning on the view direction would be more appropriate. Lastly, eq. 5.4 assumes isotropic materials since the reflectivity depends only on $\mathbf{n} \cdot \mathbf{v}$. Anisotropic materials could be handled by adding a dependency on the azimuthal angle.

5.3.2 Light Field Probes on a Coarse Grid

In this work, we replace the generic angular inputs $\mathbf{v}$, $\mathbf{n}$, $\mathbf{r}$ with new angular inputs encoding the local illumination (the light field probes), with the goal to achieve a more efficient parametrization of the appearance in terms of parameter count, training and inference speed. Similarly to previous works [27, 96], we use a spherical harmonics decomposition to model this dependency, but with several important differences: first, we encode abstract features on the

sphere rather than the outgoing radiance which lets a neural network handle the non-linearities. Second, our directional embedding has a low spatial resolution which takes advantage of the smooth spatial variations of the lighting. Third, we sample this embedding with the reflected vector instead of the view vector to link the local surface orientation with the shading function, which has been showed to be beneficial for improved surface quality [131]. The angular features encoded into a single neural probe are given by eq. 5.5:

$$\mathbf{F}_a = \mathcal{SH}(\mathbf{r}) = \sum_{j=1}^{l^2} \mathbf{b}_j Y_j(\mathbf{r}) \tag{5.5}$$

The polynomials of the SH basis are denoted with Y_j. There are l^2 coefficients[1] $\mathbf{b}_j \in \mathbb{R}^{n_a}$. Increasing values of l unlock signals of increasing angular frequency which gives an explicit control on the representable amount of specularities. We observe two guiding principles: (1) The shinier the material, the higher the frequency of $\mathbf{F}_a$ should be in the angular domain. (2) The closer the light source, the higher the frequency $\mathbf{F}_a$ should be in the spatial domain. See fig. 5.4 for an illustration.

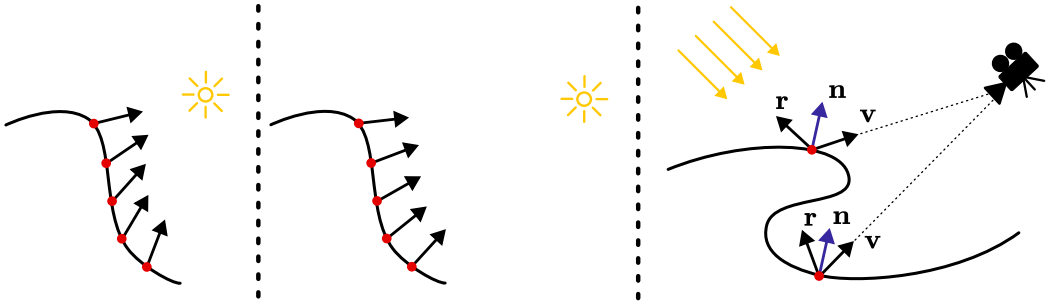

Figure 5.4: Left and center: The lighting direction changes at a lower rate on the surface for distant light sources. Right: Self-shadows impose variations in lighting even with directional illumination.

For the case of common scenes such as humans or everyday objects, assuming materials with high to medium roughness so that $\mathbf{F}_a$ does not need to be of very high angular frequency is generally sufficient. We also assume that the lights are far enough, so that probes at a low spatial frequency can explain the parallax. The probes are stored at 1/16th the resolution of the main voxel grid, as shown in fig. 5.2. The voxels tri-linearly interpolate the 8 nearest probes according to eq. 5.6 and 5.7, with w_i the interpolation weight of the i-th corner.

$$\mathbf{F}_a = \sum_{i=1}^{8} \mathcal{SH}_i(\mathbf{r}) w_i = \sum_{i=1}^{8} \sum_{j=1}^{l^2} \mathbf{b}_{ij} Y_j(\mathbf{r}) w_i \tag{5.6}$$

$$= \sum_{j=1}^{l^2} \hat{\mathbf{b}}_j Y_j(\mathbf{r}) \quad \text{with } \hat{\mathbf{b}}_j = \sum_{i=1}^{8} \mathbf{b}_{ij} w_i \tag{5.7}$$

[1] We use a one-indexed notation for the SH order l instead of the traditional zero-indexing so that we have l^2 coefficients.

5.3.3 SDF Estimation Approach

We here discuss how to embed the new proposed color decoder in an implicit surface reconstruction pipeline. During the forward pass, for a pixel (u, v) in a given viewpoint, the color is computed by a transmittance T_i-weighted average of the colors C_i of N points sampled from the current scene volume estimate along the pixel aligned ray, where T_i is the product of obstructing opacities α_j. We use the NeuS equation to relate volumetric opacities to the SDF s_i at each sample point [10]:

$$c(u,v) = \sum_{i=1}^{N} T_i \alpha_i C_i, \qquad T_i = \prod_{j<i} (1-\alpha_j), \tag{5.8}$$

$$\alpha_i = \max\left(\frac{\Phi_\tau(s_i) - \Phi_\tau(s_{i+1})}{\Phi_\tau(s_i)}, 0\right), \tag{5.9}$$

$$\Phi_\tau(s_i) = \frac{1}{1+\exp^{-\tau s_i}} \tag{5.10}$$

where τ is a scale factor that increases during optimization. The SDF field can then be optimized by backpropagating updates from the observed color loss through the rendering equation, and from the set of regularization losses.

Our losses and regularizations are detailed in equations 5.11 to 5.16. The raw SDF $\hat{s}$ is convolved by a 5^3 Gaussian kernel G into a smoother SDF s to stabilize training [30]. Note that s is used for all subsequent operations such as normal computation and rendering. Eq. 5.11 maintains s and $\hat{s}$ close to each other to promote smoothness. The Eikonal regularization 5.12 ensures that s is a signed distance. Equations 5.13, 5.14, 5.15 promote spatial smoothness for the normals, for the factorized spatial features and for the probe coefficients respectively. In eq. 5.15, $\mathcal{V}_i$ is the set of neighboring probes adjacent to the i-th one and N is the total number of probes. The regularizations are applied on each voxel, factorized feature, probe or image pixel when appropriate.

$$\mathcal{L}_{\text{sdf}} = \sum_{\text{voxel}} |s - \hat{s}|^2, \quad s = G(\hat{s}) \tag{5.11}$$

$$\mathcal{L}_{\text{Eik}} = \sum_{\text{voxel}} (||\nabla s|| - 1)^2 \tag{5.12}$$

$$\mathcal{L}_{\text{normal}} = \sum_{\text{voxel}} ||\nabla \mathbf{n}||^2, \quad \mathbf{n} = \nabla s / ||\nabla s|| \tag{5.13}$$

$$\mathcal{L}_{\text{features}} = \sum_{\text{texel}} ||\nabla \mathbf{F}_x||^2 + ||\nabla \mathbf{F}_y||^2 + ||\nabla \mathbf{F}_z||^2 \tag{5.14}$$

$$\mathcal{L}_{\text{probes}} = \sum_{i=1}^{N} \sum_{j=1}^{l^2} \sum_{k \in \mathcal{V}_i} ||\mathbf{b}_{ij} - \mathbf{b}_{kj}||^2 \tag{5.15}$$

$$\mathcal{L}_{\text{photo}} = \sum_{u,v} ||c(u,v) - c_{\text{gt}}(u,v)||^2 \tag{5.16}$$

In eq. 5.16, c is the rendered image and c_{gt} is the ground truth image. We apply a weight of $(\max(c, c_{\text{gt}}) + \epsilon)^{-1}$ on the photometric gradient of each pixel in eq. 5.16 to penalize relative differences rather than absolute differences [193]. A similar strategy is used on $\mathcal{L}_{\text{sdf}}$ and $\mathcal{L}_{\text{features}}$. We apply an empirical factor

of $(1+|s|*5)^{-1}$ on the gradients of the per-voxel regularizations to lower their importance when the distance to the zero-crossing increases. The idea is to keep the regularizations and the data term balanced for regions near and far from the surface alike.

5.3.4 Training

Our complete loss is a weighted sum of eq. 5.11 to 5.16 that we minimize with the Adam optimizer [206], please refer to the supplementary section for more details. In summary, the trained parameters are the raw SDF values $\hat{s}$, the factorized features $\mathbf{F}_x, \mathbf{F}_y, \mathbf{F}_z$, the probes coefficients $\mathbf{b}$ and the MLP weights. We follow a coarse-to-fine learning strategy, both in the spatial and angular domains. We start with coarse voxels and $l = 2$, under the supervision of low resolution images. We iteratively subdivide the voxels and switch to higher resolution images, and introduce higher angular frequencies by increasing l. Starting the optimization with too many degrees of freedom for the appearance parametrization can lead to local minima, as noted by [27].

5.4 Implementation Details

We use a 2 hidden layer MLP with 32 neurons and ReLU activations with a sigmoid for the last layer. The probes at the shared corners of adjacent tiles are duplicated in memory (but keep shared values) to benefit from hardware-accelerated interpolation. We support up to 16 spherical harmonic coefficients ($l = 4$) per probe. The computation of $\mathbf{F}_a$, $\mathbf{F}_s$ and the MLP call are fused into a single CUDA kernel for efficiency, following chapter 4. Each 16^3 tile contains $3 \times 16 \times 16 \times n_s$ spatial coefficients and $8 \times l^2 \times n_a$ spherical harmonic features. Assuming $l = 4$ and $n_s = n_a$, the probes only require $(8 \times 16)/(3 \times 16 \times 16) = 1/6$th of the memory taken by the spatial features. The sparse voxel grid is almost identical to that of chapter 4, with targeted modifications to implement our probes. The grid resolution is manually chosen so that a voxel roughly projects to an area equivalent to that of a pixel.

5.4.1 Scheduling

The scheduling of the hyper-parameters is summarized in appendix C.1. We used $\beta_1 = 0.9$, $\beta_2 = 0.995$ and $\lambda_{\text{photo}} = 40$ in all experiments, where β_1 and β_2 are ADAM's first and second moments. The gradients are accumulated over 4 to 8 complete images before stepping the optimizer. The photometric loss is divided by the number of images in the batch as a normalization. We adjust the number of levels of detail based on the resolution of the input images.

5.4.2 Neural Architecture comparison

Table 5.1 gives a comparison of several neural architectures used in the context of implicit surface reconstruction. Thanks to our light field probes, our MLP is entirely agnostic to the surface orientation and position, hence we can reduce its size and obtain high quality renderings, in real-time.

Method	Grid Type	SDF MLP Layers / Neurons	Color MLP Layers / Neurons
NeuS	✗	8 / 256	4 / 256
NeuS2	hash-grid	1 / 64	2 / 64
Voxurf	dense	✗	4 / 192
Ours	sparse	✗	2 / 32

Table 5.1: Neural architectures in the literature.

5.5 Experiments

5.5.1 Protocol

We evaluate our method on 4 datasets : MVMannequins [12], ActorsHQ [3], DTU [34] and BlendedMVS [166]. Due to the various evaluation settings employed in the community, we chose to re-train and re-evaluate all the baselines using the same experimental protocol for fairness. Following [48], all available images were given as input and used for evaluation. Timings are measured on a workstation equipped with an RTX A6000 GPU (Ampere series). PSNR values are computed inside the silhouette region of objects of interest and Chamfer error is computed using official scripts. We use the following grid resolutions: DTU (1000^3), BMVS (666^3), MVMannequins ($2500 \times 1250 \times 2500$, 2mm/voxel), ActorsHQ ($3000 \times 4000 \times 3000$, 0.6mm/voxel). Some datasets may contain inconsistent shadows or large exposure changes (DTU and BMVS) that are hard to model by our representation so we optionally train per-camera bias vectors in the MLP. Detailed tables for all our experiments can be found in appendix C.2.

5.5.2 MVMannequins

MVMannequins is a dataset of 14 dressed mannequins for multi-view reconstruction benchmarking with 68 cameras at 2048^2 resolution, that we introduced in chapter 4. We use the same evaluation protocol and report the metrics in table 5.2. We outperform all the baselines while taking only about a minute of training time, 2x to 3x faster than in chapter 4, with 5GB of VRAM, a rendering speed of 300Hz to 400Hz and a model size of 30MB.

Metrics	(4,4,4)✗	(8,8,4)✗	MMH [12]	Voxurf [30]	Neus2 [31]	Colmap [25]	2DGS [48]
Chamfer (mm)	<u>1.04</u>	**1.03**	1.14	1.59	2.13	3.52	3.35
PSNR	<u>36.81</u>	**36.90**	36.33	35.51	34.22	-	34.89
Training	1min	1min	2-3min	15min	5min	>1h	>1h

Table 5.2: Evaluation on MVMannequins with two configurations for (n_s, n_a, l). Per-camera bias vectors are not used (✗-symbol).

5.5.3 ActorsHQ

	$(n_s, n_a, l) = (4,4,4)$✗			Voxurf	NeuS2
Resolution	r/1	r/2	r/4	r/2	r/2
PSNR	**37.48**	<u>36.62</u>	34.74	36.56	34.53
Training	250s	110s	53s	1h	250s
Model size	232MB	57MB	15MB	500MB	25MB

Table 5.3: Evaluation for ActorsHQ. Per-camera bias vectors are not used (✗-symbol). Full, half and quarter-resolution input images are denoted as $r/1$, $r/2$, $r/4$ respectively. Renderings are up-scaled before comparison with the full resolution images (4088×2990).

ActorsHQ is a multiview dataset of humans in motion with 160 cameras at a high resolution of 4088×2990. We evaluate on the 1000th temporal frame of 13 sequences. We compare against Voxurf [30] and NeuS2 [31], which we trained with half resolution images since we ran into issues with full resolution images. We report the PSNR in table 5.3 and show qualitative results in fig. 5.6 and 5.7. Voxurf takes about one hour to fully converge (with $3\times$ the training iterations of the DTU configuration). NeuS2 converges fast but with a lower PSNR. In comparison, our method only takes $\sim$4 minutes to converge on the full resolution images, and we also outperform both when training at the same resolution. This benchmark highlights the capability of our method to gracefully scale to high-resolution scenes. Our method requires approximately 25GB of VRAM during training on the full resolution images, where 8GB are taken by the images. Note that the input images come with pre-baked segmentation masks, as shown in figure 5.5, that tend to have poorer accuracy on the arms and hands. This results in both geometric and photometric artifacts that are difficult to eliminate. The dataset comes with meshes reconstructed by RealityCapture [14] but we cannot compute meaningful geometric metrics since there is no reference obtained independently from the images.



Figure 5.5: Imprecise segmentation example on ActorsHQ.

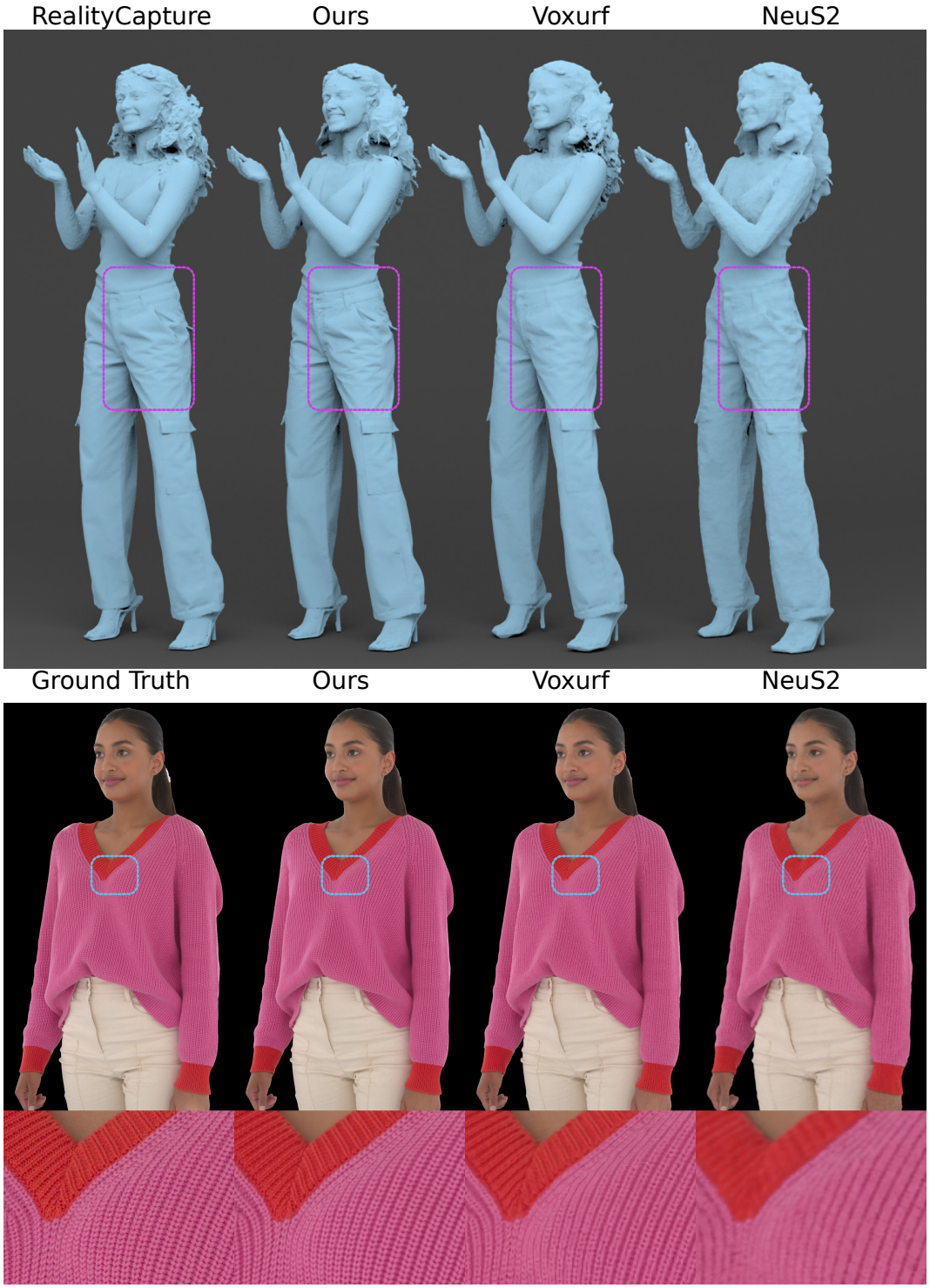

Figure 5.6: High-resolution reconstructions for ActorsHQ.

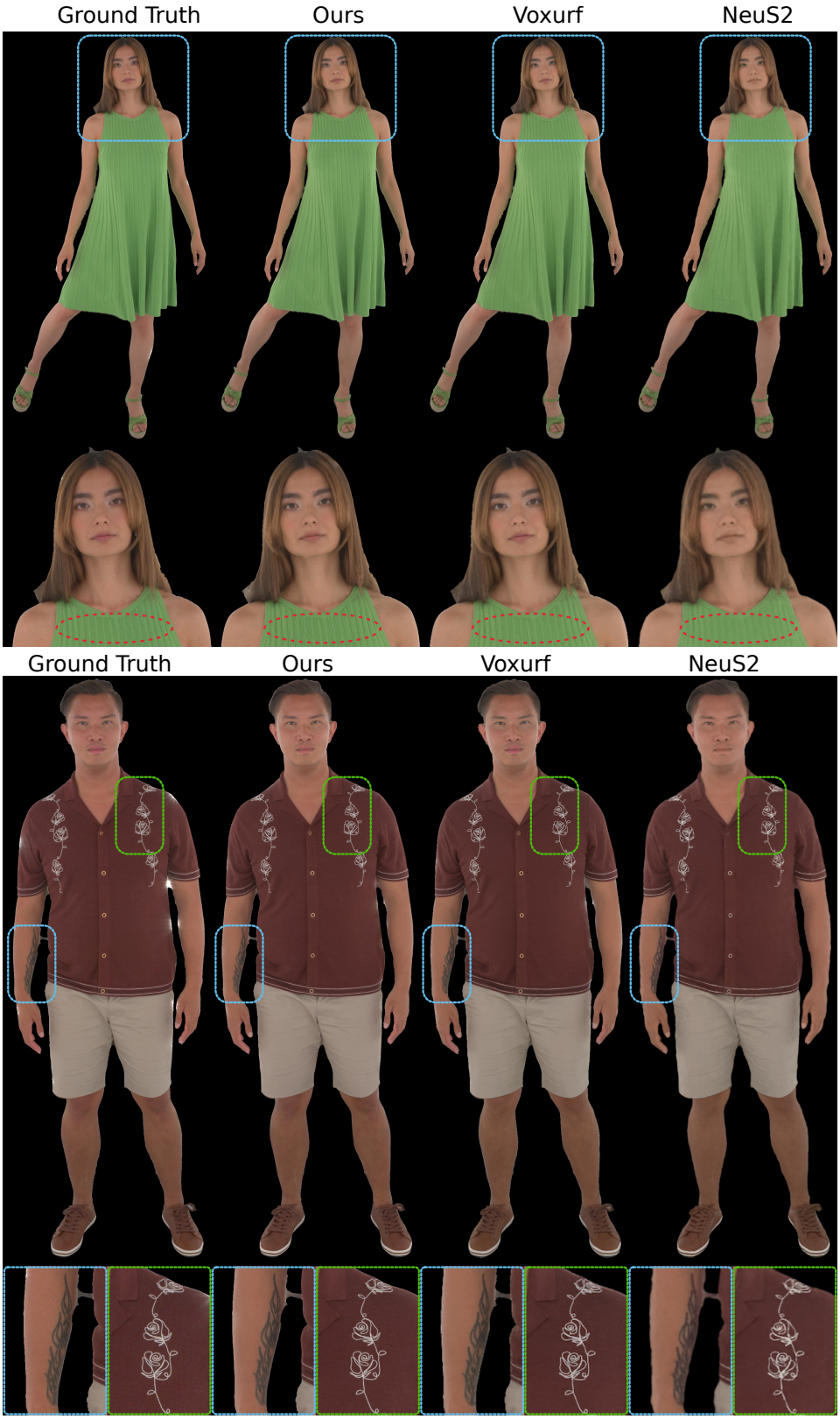

Figure 5.7: Qualitative comparison on ActorsHQ. Our method can handle the full resolution images, which lets us reconstruct textured patterns at a sub-millimetric scale.

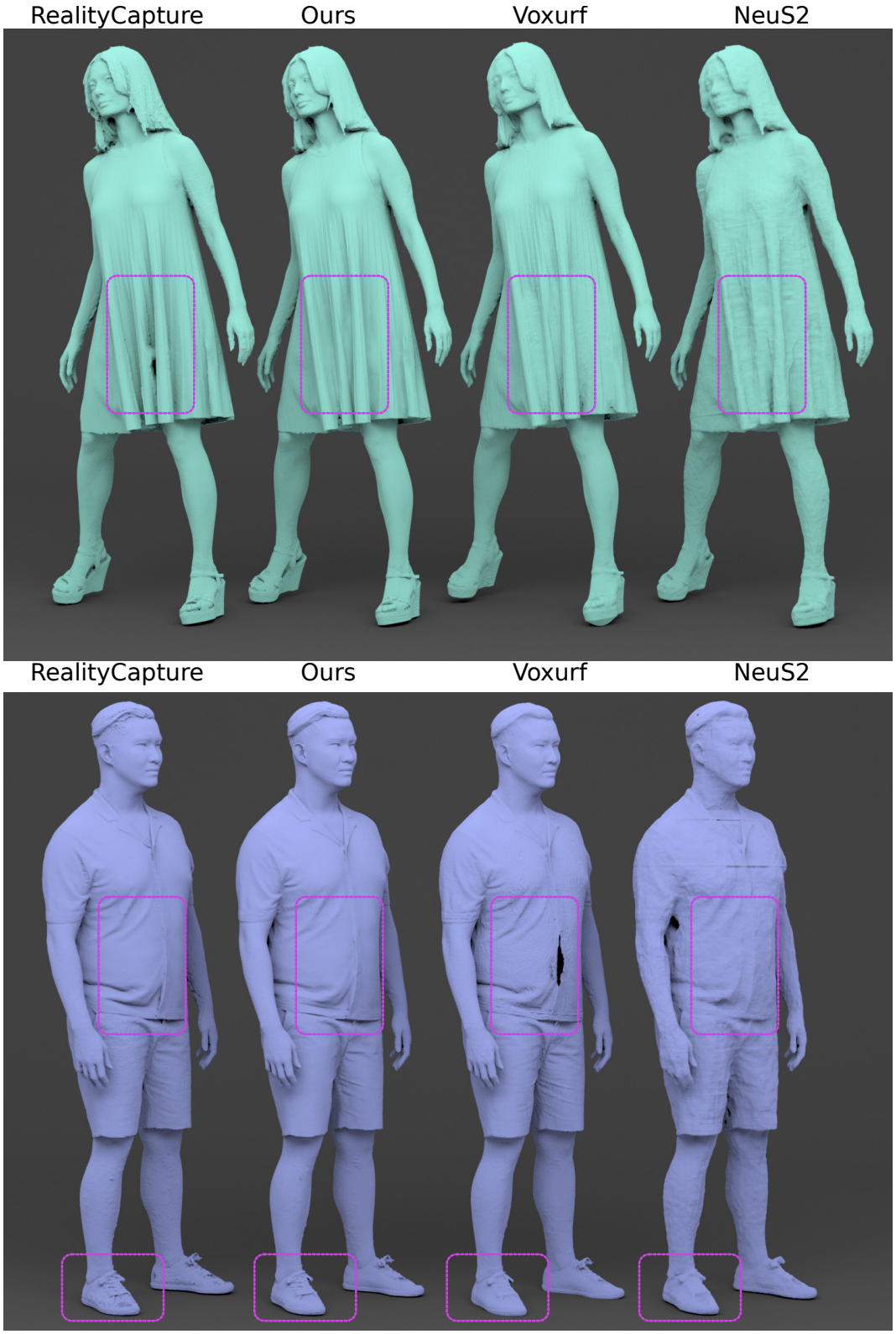

Figure 5.8: Reconstruction results on ActorsHQ.

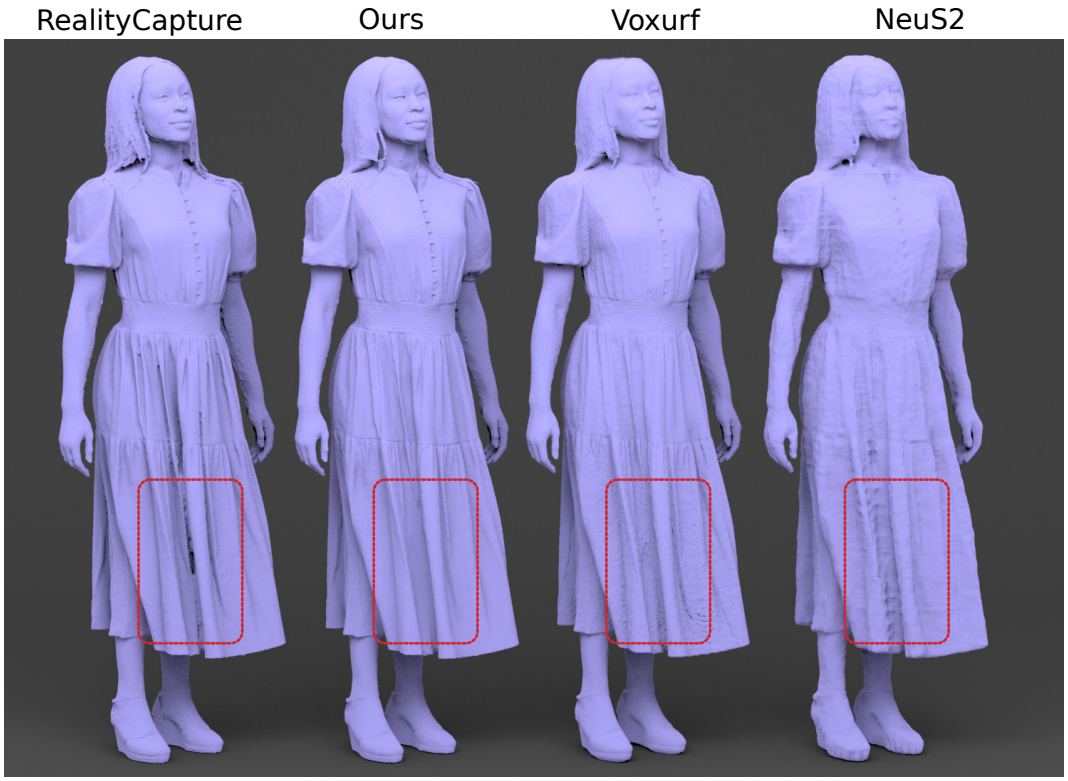

Figure 5.9: Reconstruction results on ActorsHQ.

5.5.4 DTU

The DTU dataset provides 49 to 64 images of various objects at 1600×1200 resolution, along with evaluation point clouds. We use the foreground masks provided by IDR [45] for training and testing. 2DGS [48] does not support masks for training so it uses complete images instead, cropped and at half resolution according to its evaluation protocol. We compare against Voxurf [30], Neus2 [31] and 2DGS [48] on DTU in table 5.4, using the official python script for geometric evaluation. We outperform all three in terms of Chamfer distance and PSNR, twice as fast as NeuS2 on the same hardware. Rendering speed ranges from 200Hz to 300Hz. Qualitative examples are shown in fig. 5.11. Close-ups of the volume rendered images are shown in figures 5.10 and 5.12. Our results are obtained with the (4,4,4) configuration in these figures.

Metrics	(4,4,4)✓	(8,8,4)✓	Voxurf	Neus2	2DGS
Chamfer	**0.68**	<u>0.71</u>	0.73	0.80	0.76
PSNR	37.03	**37.74**	<u>37.08</u>	36	36.03
Training	150s	160s	15min	5min	12min
Model size	56MB	88MB	500MB	25MB	49MB

Table 5.4: DTU evaluation with two configurations for (n_s, n_a, l). Per-camera bias vectors are used (✓-symbol).

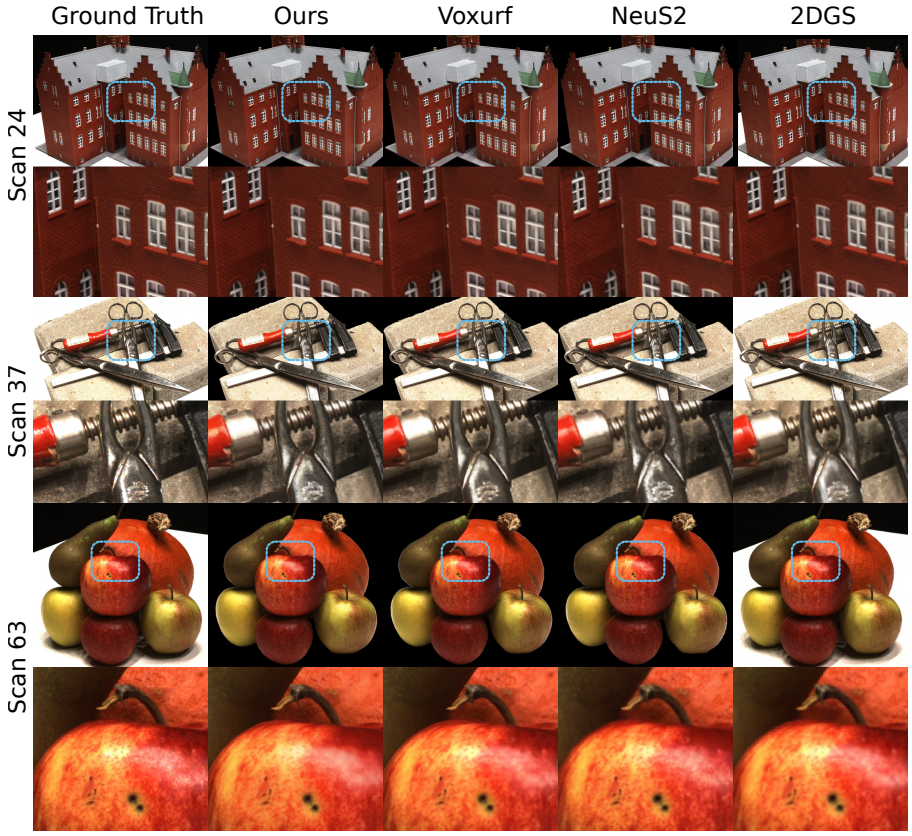

Figure 5.10: Qualitative comparison on DTU. Top: scan 24, middle: scan 37, bottom: scan 63. Left to right: ground truth, Ours, Voxurf, NeuS2, 2DGS.

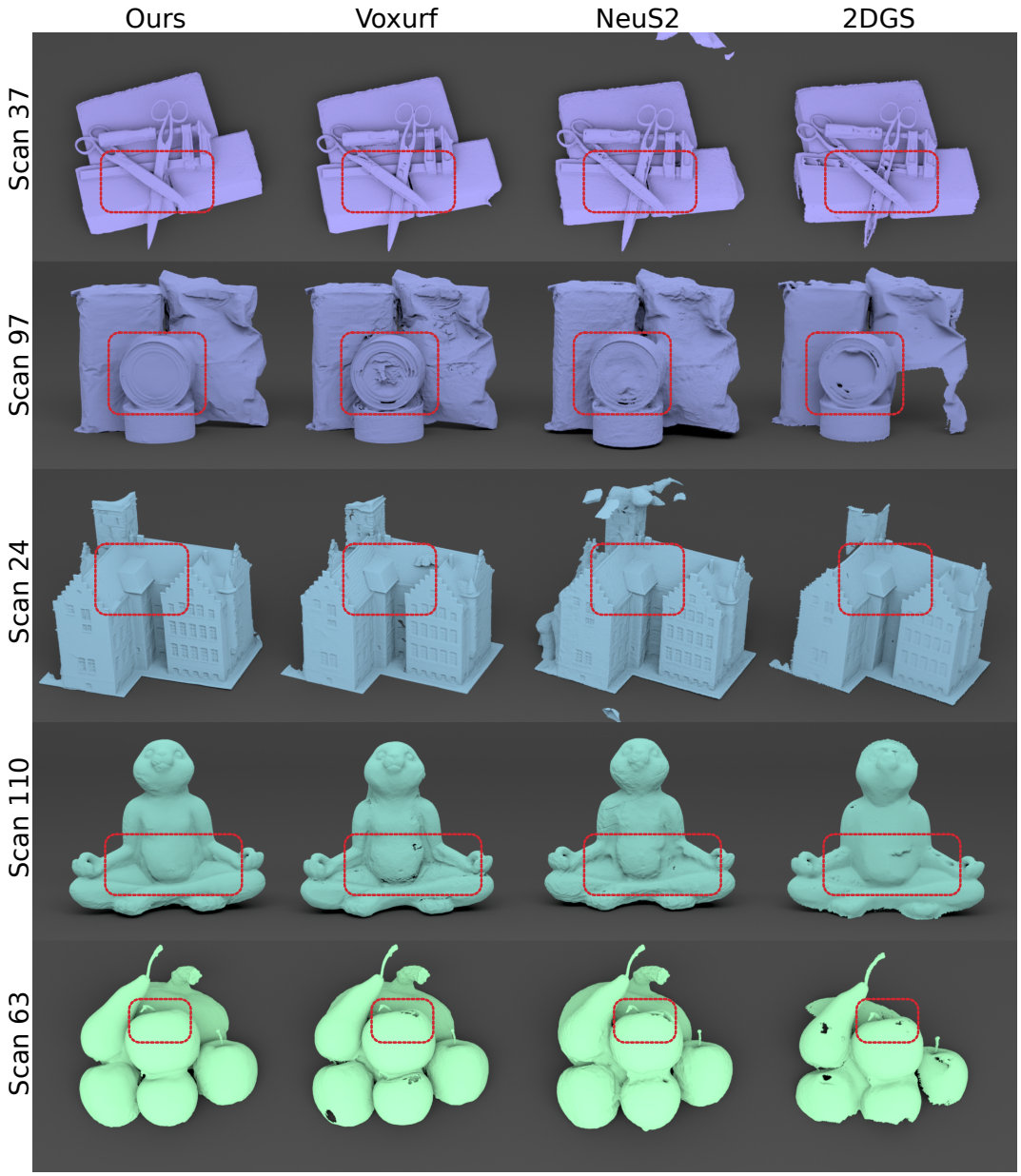

Figure 5.11: Reconstructions on challenging examples of DTU. We find that 2DGS excels at reconstructing flat surfaces (doll house roof) but tends to underperform on reflective materials. 2DGS fails to extract geometry on some parts of the objects (scans 97 and 63). In contrast, our method recovers smooth surfaces even under strong specularities (metal scissors, tuna can and apples). Voxurf struggles on the most shiny materials despite its considerably larger MLP. NeuS2's reconstruction suffers from grid-aligned artifacts, possibly due to discontinuities in its hash-grid interpolation scheme (shoulder of the bunny in scan 110).

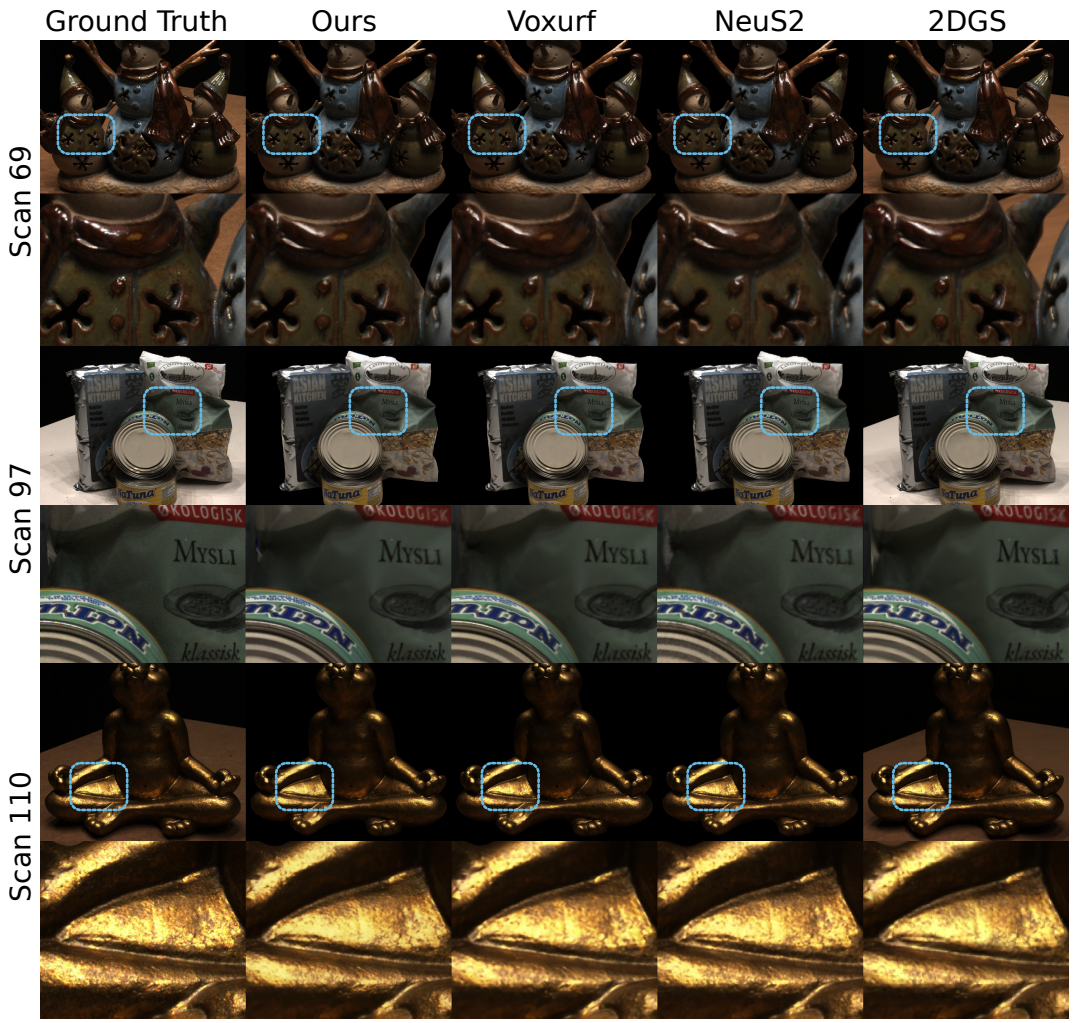

Figure 5.12: Qualitative comparison on DTU. Top: scan 69, middle: scan 97, bottom: scan 110. Left to right: ground truth, Ours, Voxurf, NeuS2, 2DGS.

5.5.5 BlendedMVS

Finally, we test our method on 8 scenes of the BlendedMVS dataset [166] and report the metrics in table 5.5. BlendedMVS is a semi-synthetic dataset were objects have been first reconstructed then reprojected in the images in order to produce reliable ground truth geometry and masks. There is no official script for geometric evaluation so we measure two-ways point to mesh distances in the reference frame used for training in [10], scaled by 1000x. Geometric comparisons are shown in figures 5.13 and 5.14, qualitative comparisons of the volume rendering are shown in figure 5.15. We used the (8,8,4) configuration here as it performed a little better (see table C.14 for a breakdown).

Metrics	(4,4,4)✓	(8,8,4)✓	Voxurf	Neus2
Chamfer	2.37	**2.22**	2.64	2.93
PSNR	35.19	**35.89**	35.11	33.62
Training	110s	120s	15min	5min

Table 5.5: BMVS evaluation with two configurations for (n_s, n_a, l). Per-camera bias vectors are used (✓-symbol).

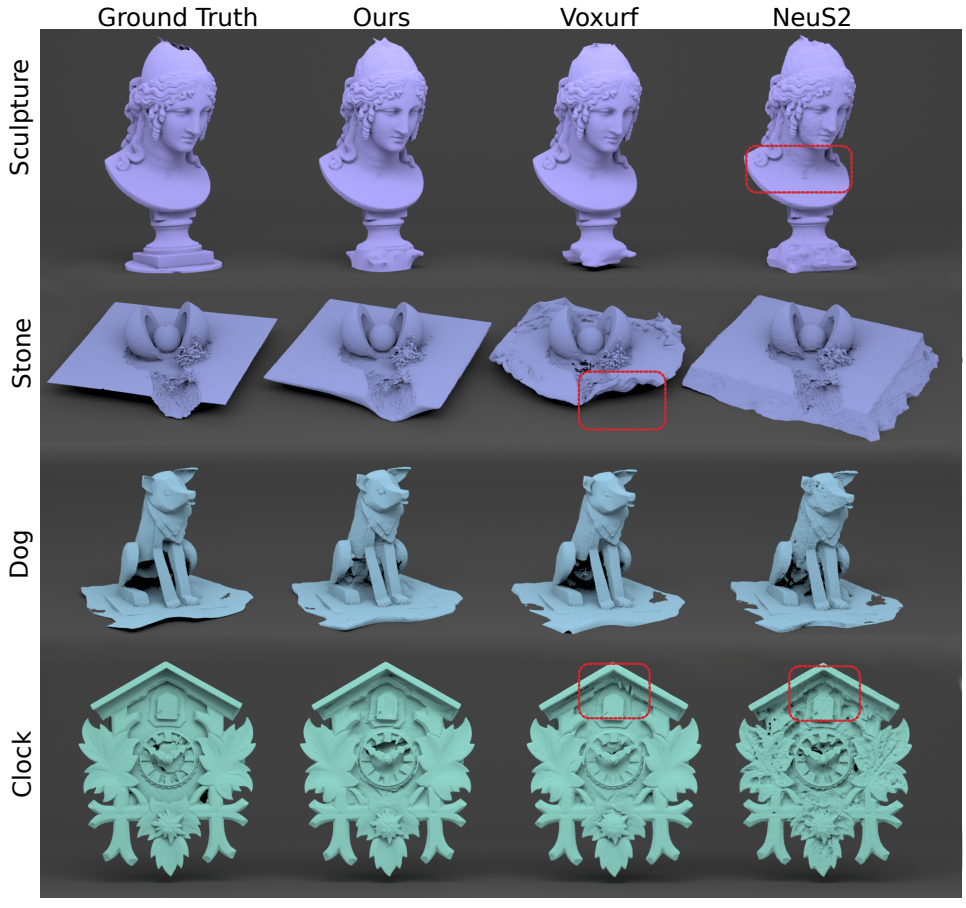

Figure 5.13: Reconstructions examples on BlendedMVS.

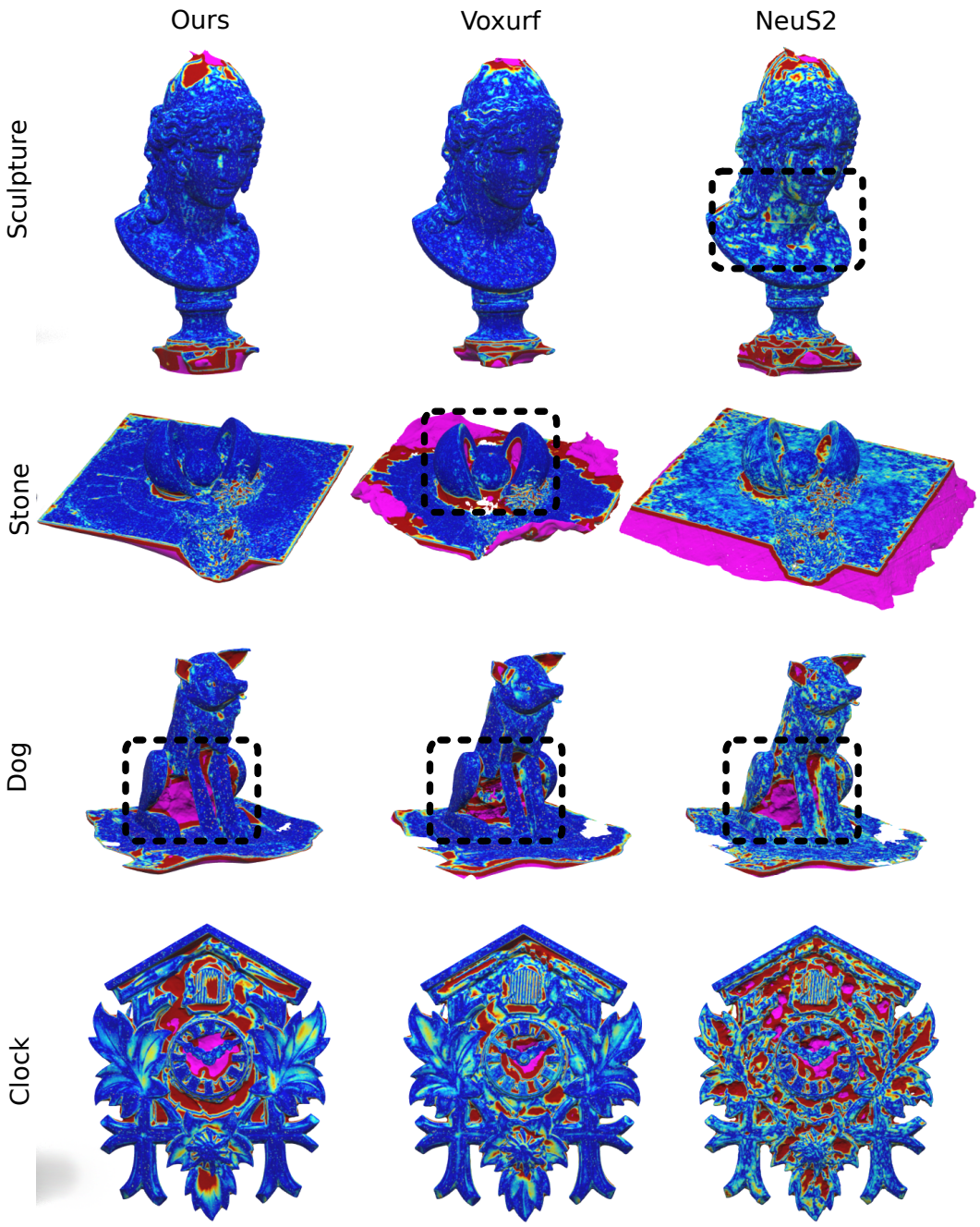

Figure 5.14: Accuracy heatmaps on BlendedMVS. The pink color denotes points too far away from the ground truth, which are ignored in the computation of the metrics. Left to right: Ours, Voxurf, NeuS2. Voxurf fails to carve inside the two hemispheres in the stone example and the corners of the base are missing. However, Voxurf is able to carve under the dog statue whereas both NeuS2 and our method fail on this example. NeuS2 is noticably more noisy on all examples.



Figure 5.15: Qualitative comparison on BlendedMVS.

5.5.6 Ablations

We ablate the probes smoothing and the polynomial approximation of the Fresnel term in tab. 5.6 on the MVMannequins dataset by imposing $\mathbf{n} \cdot \mathbf{v} = 1$ in eq. 5.4 during training, with noticeable incidence on Chamfer error. A visual ablation of the components of our parametrization is presented in fig. 5.16 on the *kino/jea* mannequin, clearly showing the role of each parameter group. Figure 5.17 contains more examples. Various choices of configurations are compared in table 5.7 on the BlendedMVS dataset. A visual ablation of the spherical harmonic order l is presented in fig 5.18.

	(4,4,4)✗	w/o probes smoothing	w/o Fresnel
Chamfer (mm)	**1.04**	1.09	1.18
PSNR	36.81	**36.91**	36.79

Table 5.6: Ablation on MVMannequins

Metrics	(4,4,4)✗	(4,4,4)✓	(8,8,4)✓	(12,12,4)✓	(4,4,1)✓	(4,4,2)✓	(4,4,3)✓
Chamfer	2.47	2.37	**2.22**	2.31	2.58	2.34	2.37
PSNR	34.76	35.19	35.89	**36.17**	34.44	34.86	35.05

Table 5.7: We ablate the number of spatial features n_s, angular features n_a and spherical harmonics order l on BlendedMVS. Each triplet denotes a configuration (n_s, n_a, l), the ✓-symbol indicates training with per-camera bias vectors.

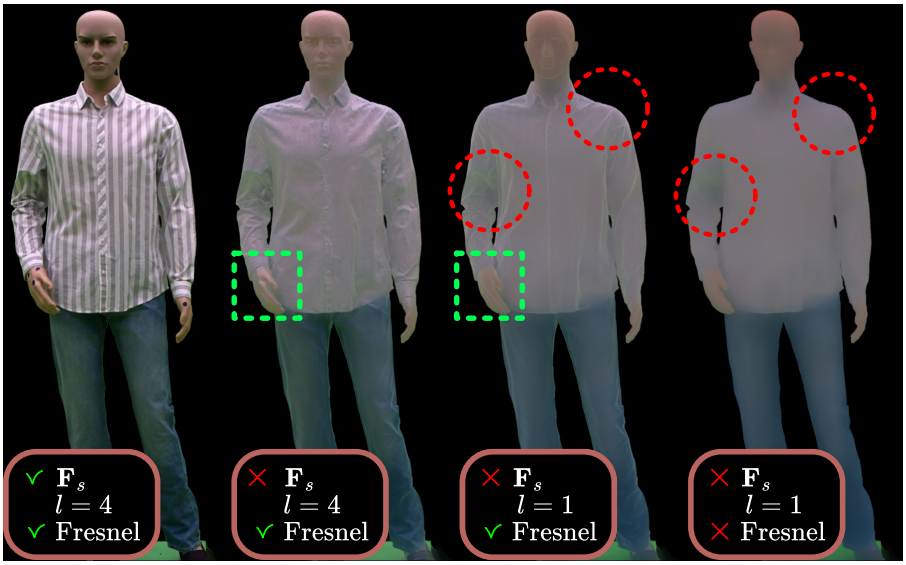

Figure 5.16: Our model can be intuitively explained. We disable some components of a pre-trained example from the MVMannequins dataset [12] (left). From left to right, we remove the spatial features (set $\mathbf{F}_s = 0$), only keep the constant coefficient of the SH basis ($l = 1$), disable the incidence angle embedding (set $\mathbf{n} \cdot \mathbf{v} = 1$ in eq. 5.4). The learned Fresnel term encodes grazing reflections (red circles), the angular features provide the shading (green squares) and the spatial features contain the high frequency details.

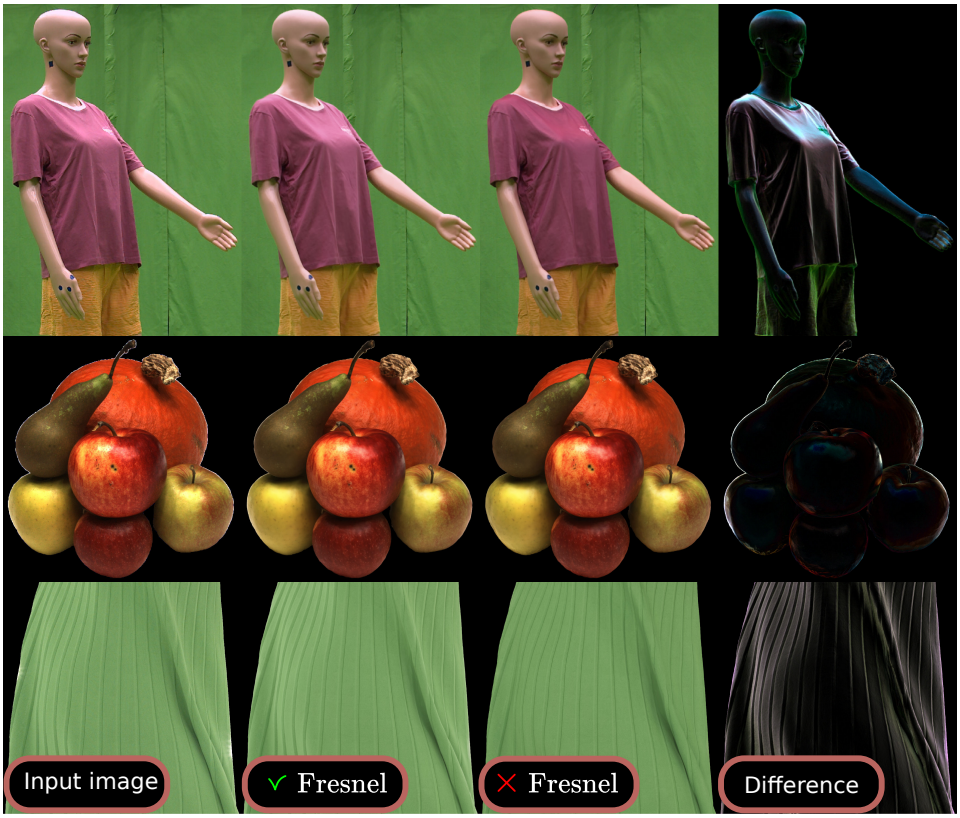

Figure 5.17: Effect of the learned Fresnel term on three examples from MV-Mannequins, DTU and ActorsHQ. Left to right: input image, volume rendering (A), volume rendering w/o the Fresnel term (B) (we set $\mathbf{n} \cdot \mathbf{v} = 1$ in eq. 5.4), difference image ($4 \times |A - B|$).

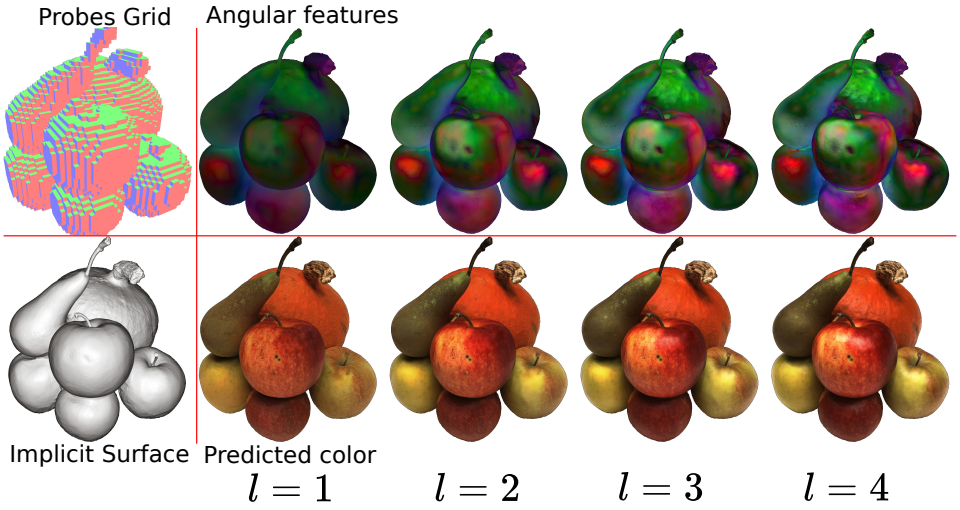

Figure 5.18: We train the scene 63 from DTU with $l = 4$, that we then visualize with l varying from 1 (constant term) to 4, left to right. The corresponding angular features are on the top row, and the predicted color is on the bottom row. We observe that the specularities can be removed simply by disabling the high order spherical harmonics coefficients.

5.5.7 Performance Analysis

We record inference timings on the apples example (scan 63 of DTU, (4,4,4) configuration), with a window of resolution 1920×1163 and present the results in table 5.8. We lock the memory clock to 5001MHz and the gpu clock to 1500MHz to obtain stable performance measurements. The shading column corresponds to the assignment of a color to each voxel. The render column corresponds to the volume rendering kernel, which samples the SDF and color fields along rays to generate the final image. The first 4 rows correspond to the fully-fused color prediction kernel, with the computation of the spatial and angular features enabled or disabled. Thus, the 4th row corresponds to the MLP inference only whereas the 1st row corresponds to the full model. The last two rows correspond to the computation of $\mathbf{F}_s$ or $\mathbf{F}_a$ on their own, in separate kernels, and whose result is interpreted as a per-voxel color for visualization.

Type	Shading	Render
MLP ✓, $\mathbf{F}_s$ ✓, $\mathbf{F}_a$ ✓	2.47	1.34
MLP ✓, $\mathbf{F}_s$ ✓, $\mathbf{F}_a$ ✗	1.92	1.34
MLP ✓, $\mathbf{F}_s$ ✗, $\mathbf{F}_a$ ✓	2.32	1.34
MLP ✓, $\mathbf{F}_s$ ✗, $\mathbf{F}_a$ ✗	1.80	1.34
$\mathbf{F}_a$ only	1.96	1.34
$\mathbf{F}_s$ only	0.82	1.34

Table 5.8: Timings in ms

We observe that just evaluating the MLP (4th row) or computing the angular features on their own (5th row) roughly takes the same amount of time (1.80ms and 1.96ms) but that fusing the two operations together only takes 2.32ms, much less than the sum of the two ($1.80 + 1.96 = 3.76$ms). Including the computation of the spatial features gives the full model at 2.47ms.

5.6 Limitations

We identify several limitations to our approach. First, the fact that the lighting is encoded locally can degrade the extrapolation capabilities compared to modeling the environment with a global representation. This becomes apparent on the ActorsHQ dataset where there is a good overlap of the cameras' field of views along the longitude, but very little overlap along the latitude on the legs. This creates shadows artifacts when looking from above or below rather that at a level angle. Second, although our decomposition is physically-inspired, we do not explicitly simulate light transport which can result in local minima due to shape-radiance ambiguities. Relighting and/or material editing

is not supported but fine-tuning the probes to new lighting seems possible and worthy of future investigations.

5.7 Conclusion

This chapter proposed a novel appearance model, based on a local factorization of angular and spatial components of the light field. This decomposition is explicit, which further decreases the reliance on the neural component and improves expressivity without degrading inference speed. We postulate that surface regions can be learned more independently than with a centralized representation, which is why we observe a much faster convergence than our approach of chapter 4.

Our physically-inspired radiance parametrization lets us reconstruct shiny materials more effectively than typical neural parametrization, thanks to the geometric supervision from the normal vectors. The remarkable parameter efficiency, rivaling with a physically-based parametrization, makes it possible to scale to high-resolution datasets where naive approaches fail.

Compared to decentralized representations [27, 96], we largely decrease the number of parameters per voxel or Gaussian by exploiting the local spatial smoothness of lighting. This is particularly useful for regions with high texture frequency under a distant illumination. We empirically observe that sharp shadows can still be explained through the spatial features if the radiance probes do not have a sufficient spatial density, which makes our representation quite flexible. However, unstructured representations [6, 7, 27] still have an inherent advantage when it comes to featureless regions that can be explained by fewer large primitives.

Disentangling the spatial and angular components with high efficiency also offers promising future avenues of research toward temporal reconstruction. Other future research can build on this work, *e.g.* handling high frequency reflections at a lower cost than current dedicated approaches and unifying neural rendering and physically-based rendering. Other embeddings of the angular dependency representations can be investigated, such as cubemaps or mixtures of spherical Gaussians.

5.8 Acknowledgments

This work was supported by French government funding managed by the National Research Agency under the Investments for the Future program (PIA) grant ANR- 21-ESRE-0030 (CONTINUUM). Research conducted at Kyushu University was supported by JSPS/KAKENHI JP23H03439 and AMED JP24wm0625404. We thank Laurence Boissieux for her valuable help generating 3D renderings, and Edmond Boyer for insightful discussions and comments.

6 Conclusion

6.1 Summary

This thesis investigated the use of differential rendering for volumetric capture with RGB images. We showed that parsimony and accuracy can be combined, achieving millimetric precision in just one minute of computation on multi-view data from a typical capture platform. The recovered radiance field can be rendered in real-time at several hundreds of frames per second, and appears temporally consistent for temporal sequences. Our results also extend beyond this context, to the reconstruction of generic objects, where we outperform many previous works. We targeted the challenging problem of recovering both accurate geometry and appearance, while most approaches only consider one or the other.

Chapter 3 presented a high-resolution framework for volumetric differential rendering, cast as a classic optimization problem with non-stochastic gradient descent. We accelerated convergence using a coarse-to-fine strategy, analytically-derived gradients, and an approximate back-propagation scheme for volume rendering. We proposed to regularize the opacity field for surface extraction, by enforcing a monotonous transition of the opacity from transparent to opaque and explicitly labeling interior and exterior regions.

In chapter 4, we contributed a parsimonious pipeline for implicit surface reconstruction. Our main idea was to redirect more responsibilities to the spatial embedding, in order to lower the reliance on the appearance model compared to previous neural approaches. We do so by storing high-resolution but low-dimensional feature vectors in a sparse grid, along with explicit signed-distance values. Thus, we more effectively disentangle spatial responsibilities (shape, texture) from angular responsibilities (shading), that can be handled by a smaller neural network. An important consequence is that we can fuse color prediction and volume rendering in just two GPU kernels, that do not require temporary memory beyond their inputs and outputs. It follows that our approach supports an unlimited number of samples per ray, whose spacing

is adjusted on-the-fly, and can render complete images at once, leading to faster convergence than approaches restricted to small batches of rays. We also derive allocation bounds so that the topology of the sparse grid can be dynamically updated, as the shape estimate changes. In parallel, we contribute a dataset of clothed mannequins for unbiased evaluation of reconstruction algorithms in the context of a capture platform.

Chapter 5 expanded our approach of chapter 4 to handle shinier materials and scenes with less favorable illumination conditions. We proposed to extract most of the angular dependency, previously encoded in a single neural component, into explicit angular features. This is analogous to extracting the spatial dependency in a voxel grid but transposed to the angular domain. Therefore, our neural component becomes agnostic to surface orientation and position, and acts closer to a BRDF decoder than to a radiance cache. The resulting appearance model is expressive yet efficient and achieves high-quality results on human and object scenes. Thanks to its parsimony, our approach can leverage high-resolution input images and recovers photometric and geometric details at a sub-millimetric scale.

Throughout this thesis, we explored minimal representations for shape and appearance. This lead us to significant and simultaneous savings in compute, memory and model size, without compromising on accuracy.

6.2 Limitations

This work is based on several assumptions which make sense in a volumetric capture context but typically do not apply in a more general setting.

For instance, we assumed that all the cameras observe the scene from equal distances so that all the voxels can have the same size. A less constrained camera placement may require a multi-resolution representation to avoid aliasing artifacts. The balance of the regularizations and of the photometric loss may also be affected if the scene is imaged at multiple scales. This could also explain why our approach requires some fine-tuning when switching to another dataset.

For similar reasons, we cannot reconstruct complete scenes that typically require an adaptive discretization. Note that our ray-marching algorithm relies on a regular subdivision of space, such that directly applying a contraction function [38] is not possible. Handling complete scenes would also be beneficial in surface capture scenarios since clean background images are sometimes unavailable: the ground is covered with a cloth in [52], that moves when the actors walk around.

Our coarse-to-fine training strategy assumes that down-scaled images still contain sufficient information to converge at the corresponding grid level-of-detail. Images that only contain high-frequency details but no low-frequency shading information are particularly difficult (*e.g.* DTU's tiled roof doll house), as they will appear uniform in color once down-scaled. An inconsistent illumination between the viewpoints also prevents exploiting shading cues at low resolutions. Since the width of the voxel band is halved at each level of detail, the receptive field of the volume rendering may not longer overlap with the true

surface position, preventing recovery at higher resolutions. We postulate that human scenes contain a mix of high- and low-frequency signals which makes our approach work in the targeted use case. Most implementations avoid this issue by training directly with full-resolution images and leveraging a multi-scale representation or benefiting from an SfM point cloud initialization.

6.3 Future works

Future works can investigate the problem of obtaining temporally-consistent reconstructions, for which our appearance model of chapter 5 seems promising, as it decorrelates intrinsic (time-independent) and extrinsic (time-dependent) surface properties. Tasks adjacent to the reconstruction problem such as relighting, animating or editing can also be considered.

Substantial efficiency gains seem possible, as our sparse voxel grid appears wasteful in multiple ways. First, we hypothesize that voxels far from the iso-surface could be encoded at a lower resolution. Second, all the voxels of a given tile are kept in memory even though only few of them may be active, which results in wasted computation or bandwidth during color prediction and ray-marching. Our approach converges quickly because each training iteration is cheap, but the total number of iteration can be large. Improving convergence rates using a second-order optimizer [207] or automatically detecting convergence could effectively reduce training time.

Further improving geometric quality without decreasing performance is another interesting direction of research. Relying on an NCC loss [46, 50] or a generalized sdf to opacity conversion [204, 208] are interesting possibilities.

We rely on the signed-distance property for importance sampling and to determine allocation bounds for our voxel grid. This is possible as long as the pseudo-sdf stored in the voxels is well-regularized through the Eikonal constraint. Unfortunately, we observed a coupling between the smoothness regularization and the Eikonal regularization such that they cannot be adjusted independently. Thus, promoting a better signed-distance approximation requires a stronger smoothness that may prevent convergence. We believe that deriving the signed-distance property from a computation (a distance transform) could address the problem.

A

Appendix of Chapter 3

A.1 Detailed definitions

A.1.1 Robust loss

The robust Huber norm and its derivative are:

$$||x||_{\mathcal{H}} = \begin{cases} \frac{x^2}{2} & \text{if } x < \epsilon \\ \epsilon(|x| - \frac{\epsilon}{2}) & \text{if } x \geq \epsilon \end{cases} \tag{A.1}$$

$$||x||'_{\mathcal{H}} = \begin{cases} x & \text{if } x < \epsilon \\ \epsilon \text{ sign}(x) & \text{if } x \geq \epsilon \end{cases} \tag{A.2}$$

If $\mathbf{x}$ is a vector, we take the sum of the norms applied component-wise:

$$||\mathbf{x}||_{\mathcal{H}} = \sum ||x_i||_{\mathcal{H}} \tag{A.3}$$

$$||\mathbf{x}||'_{\mathcal{H}} = \begin{bmatrix} ||x_1||'_{\mathcal{H}} \\ \dots \\ ||x_n||'_{\mathcal{H}} \end{bmatrix} \tag{A.4}$$

A.1.2 Parametrizations

$$\alpha_i \in [0,1] \qquad \text{(transparency of a sample)}$$
$$\mathbf{c}_i \in [0,1]^3 \qquad \text{(color of a sample)}$$
$$\alpha = \text{lerp}(\bar{\alpha}) \qquad \text{(transparency interpolation in the voxel grid)}$$
$$\mathbf{c} = \text{lerp}(\bar{\mathbf{c}}) \qquad \text{(color interpolation in the voxel grid)}$$
$$\bar{\mathbf{c}}(\mathbf{v}) = \sum_{l=0}^{B} \sum_{m=-l}^{l} Y_{l,m}(\mathbf{v}) \beta_{l,m} \qquad \text{(spherical harmonic basis)}$$
$$g, b \in \mathbf{R}^+ \qquad \text{(Per camera gain and bias)}$$
$$\mathbf{C}_{\text{gt}} = g\mathbf{C}_{\text{uncorrected gt}} + b \qquad \text{(Ground truth color)}$$
$$\mathbf{C}_{\text{bg}} = g\mathbf{C}_{\text{uncorrected bg}} + b \qquad \text{(Background color)}$$

A.1.3 Discrete integrals

$$T_k = \prod_{j<k} \alpha_j \qquad \text{(partial transmittance)}$$
$$\mathbf{C}_k = \sum_{j \leq k} (1-\alpha_j)\mathbf{c}_j T_j \qquad \text{(partial color)}$$
$$\mathbf{C} = \sum_{k<\infty} (1-\alpha_k)\mathbf{c}_k T_k + T_\infty \mathbf{C}_{\text{bg}} \qquad \text{(total integral)}$$

A.1.4 Losses

$$\mathcal{L}_{\text{photo}} = \sum_{r \in \text{rays}} ||\mathbf{C}(r) - \mathbf{C}_{\text{gt}}(r)||_{\mathcal{H}} \qquad \text{(robust photometric loss)}$$
$$\mathcal{L}_{\text{samples}} = \sum_{\text{samples}} ||\mathbf{c}_k - \mathbf{C}||_{\mathcal{H}} T_k (1-\alpha_k) \qquad \text{(ray color consistency)}$$
$$\mathcal{L}_{\text{smooth } \alpha} = \sum_{\text{voxels}} ||\nabla \bar{\alpha}||_2^2 \qquad \text{(spatial regularization)}$$
$$\mathcal{L}_{\text{SH parsimony}} = \sum_{\text{voxels}} ||\beta_{l,m}||_{\mathcal{H}} \qquad \text{(spherical harmonics parsimony)}$$
$$\mathcal{L}_{\text{ballooning}} = \sum_{\text{voxels}} \bar{\alpha}^2 \qquad \text{(ballooning occupancy bias)}$$

A.2 Analytical gradient derivations

A.2.1 Gradients of the photometric loss

$$\mathcal{L}_{\text{photo}} = \sum_{r \in \text{rays}} ||\mathbf{C}(r) - \mathbf{C}_{\text{gt}}(r)||_{\mathcal{H}} \tag{A.5}$$
$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \mathbf{c}_k} = \frac{\partial \mathcal{L}_{\text{photo}}}{\partial \mathbf{C}} \frac{\partial \mathbf{C}}{\partial \mathbf{c}_k} \tag{A.6}$$
$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \alpha_k} = \frac{\partial \mathcal{L}_{\text{photo}}}{\partial \mathbf{C}} \frac{\partial \mathbf{C}}{\partial \alpha_k} \tag{A.7}$$
$$\tag{A.8}$$

Where:

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \mathbf{C}} = ||\mathbf{C} - \mathbf{C}_{\text{gt}}||'_{\mathcal{H}} \tag{A.9}$$

$$\frac{\partial \mathbf{C}}{\partial \mathbf{c}_k} = (1-\alpha_k)T_k \tag{A.10}$$

$$\frac{\partial \mathbf{C}}{\partial \alpha_k} = -\mathbf{c}_k T_k + \sum_{i>k}(1-\alpha_i)\mathbf{c}_i \frac{T_i}{\alpha_k} + \frac{T_\infty}{\alpha_k}\mathbf{C}_{\text{bg}} \tag{A.11}$$

$$= -\mathbf{c}_k T_k + \frac{1}{\alpha_k}\left(\sum_{i>k}(1-\alpha_i)\mathbf{c}_i T_i + T_\infty \mathbf{C}_{\text{bg}}\right) \tag{A.12}$$

$$= -\mathbf{c}_k T_k + \frac{1}{\alpha_k}\left(\mathbf{C} - \sum_{i\leq k}(1-\alpha_i)\mathbf{c}_i T_i\right) \tag{A.13}$$

$$= -\mathbf{c}_k T_k + \frac{\mathbf{C} - \mathbf{C}_k}{\alpha_k} \tag{A.14}$$

A.2.2 Gradients of the ray color consistency loss

$$\mathcal{L}_{\text{samples}} = \sum_{\text{samples}} ||\mathbf{c}_k - \mathbf{C}||_{\mathcal{H}} T_k (1-\alpha_k) \tag{A.15}$$

$$\frac{\partial \mathcal{L}_{\text{sample}}}{\partial \mathbf{c}_k} = ||\mathbf{c}_k - \mathbf{C}||'_{\mathcal{H}} T_k (1-\alpha_k)(1 - \frac{\partial \mathbf{C}}{\partial \mathbf{c}_k}) \tag{A.16}$$

$$\frac{\partial \mathcal{L}_{\text{sample}}}{\partial \alpha_k} = ||\mathbf{c}_k - \mathbf{C}||'_{\mathcal{H}} \cdot \frac{\partial \mathbf{C}}{\partial \alpha_k} - ||\mathbf{c}_k - \mathbf{C}||_{\mathcal{H}} T_k \tag{A.17}$$

A.2.3 Gradients of the spatial regularization loss

We convert the discrete sum to an integral and do a Taylor series expansion. We then apply the multi-dimensional integration by parts formula.

$$\mathcal{L}_{\text{smooth }\alpha} = \sum_{\text{voxels}} ||\nabla \bar{\alpha}||_2^2 \approx \int_\Omega ||\nabla \bar{\alpha}||_2^2 d\Omega = \int_\Omega \nabla \bar{\alpha} \cdot \nabla \bar{\alpha} d\Omega \tag{A.18}$$

$$\mathcal{L}_{\text{smooth }\alpha}(\bar{\alpha}+h) = \int_\Omega \nabla \bar{\alpha} \cdot \nabla \bar{\alpha} + 2\nabla h \cdot \nabla \bar{\alpha} + \nabla h \cdot \nabla h d\Omega \tag{A.19}$$

$$= \mathcal{L}_{\text{smooth }\alpha} + 2\int_\Omega \nabla h \cdot \nabla \bar{\alpha} d\Omega + o(h^2) \tag{A.20}$$

$$= \mathcal{L}_{\text{smooth }\alpha} + 2\left(-\int_\Omega h \Delta \bar{\alpha} d\Omega + \int_{\partial\Omega} h \nabla \bar{\alpha} \cdot n d\sigma\right) + o(h^2) \tag{A.21}$$

$$= \mathcal{L}_{\text{smooth }\alpha} - 2\int_\Omega h \Delta \bar{\alpha} d\Omega + o(h^2) \tag{A.22}$$

Where Ω is a volume encompassing the voxel grid and $\partial\Omega$ is its border. We make the assumption that $\bar{\alpha}$ is constant outside the voxel grid so that $\nabla \bar{\alpha} = 0$ on $\partial\Omega$. Thus $\frac{\partial \mathcal{L}_{\text{smooth }\alpha}}{\partial \bar{\alpha}} = -2\Delta\bar{\alpha}$. Note that the same result can be obtained by staying in the discrete world and computing partial derivatives as usual, the notation is just more involved.

A.2.4 Gradients of the spherical harmonics parsimony loss

$$\mathcal{L}_{\text{SH parsimony}} = \sum_{\text{voxels}} ||\beta_{l,m}||_{\mathcal{H}} \tag{A.23}$$

$$\frac{\partial \mathcal{L}_{\text{SH parsimony}}}{\partial \beta_{l,m}} = ||\beta_{l,m}||'_{\mathcal{H}} \tag{A.24}$$

A.2.5 Gradients of the ballooning loss

$$\mathcal{L}_{\text{ballooning}} = \sum_{\text{voxels}} \bar{\alpha}^2 \tag{A.25}$$

$$\frac{\partial \mathcal{L}_{\text{ballooning}}}{\partial \bar{\alpha}} = 2\bar{\alpha} \tag{A.26}$$

A.2.6 Gradients of the exposure regularization

$$\mathbf{C} = \sum_{k<\infty} (1-\alpha_k)\mathbf{c}_k T_k + T_\infty \mathbf{C}_{\text{bg}} \tag{A.27}$$

$$\mathcal{L}_{\text{photo}} = \sum_{r\in\text{rays}} ||\mathbf{C}(r) - \mathbf{C}_{\text{gt}}(r)||_{\mathcal{H}} \tag{A.28}$$

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial g_i} = \sum_{r\in\text{rays}} ||\mathbf{C}(r) - \mathbf{C}_{\text{gt}}(r)||'_{\mathcal{H}} \cdot (T_\infty \mathbf{C}_{\text{uncorrected bg}} - \mathbf{C}_{\text{uncorrected gt}}) \tag{A.29}$$

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial b_i} = \sum_{r\in\text{rays}} ||\mathbf{C}(r) - \mathbf{C}_{\text{gt}}(r)||'_{\mathcal{H}} \cdot \text{vec3}(T_\infty - 1) \tag{A.30}$$

$$\mathcal{L}_{\text{samples}} = \sum_k ||\mathbf{c}_k - \mathbf{C}||_{\mathcal{H}} T_k (1-\alpha_k) \tag{A.31}$$

$$\frac{\partial \mathcal{L}_{\text{sample}}}{\partial g_i} = -\sum_k ||\mathbf{c}_k - \mathbf{C}||'_{\mathcal{H}} \cdot \mathbf{C}_{\text{uncorrected bg}} T_\infty T_k (1-\alpha_k) \tag{A.32}$$

$$\frac{\partial \mathcal{L}_{\text{sample}}}{\partial b_i} = -\sum_k ||\mathbf{c}_k - \mathbf{C}||'_{\mathcal{H}} \cdot \text{vec3}(1) T_\infty T_k (1-\alpha_k) \tag{A.33}$$

We penalize exposure and bias coefficients so that the overall luminosity of the cameras stays roughly unchanged:

$$\mathcal{L}_{\text{expo}} = \sum_{i<n_{\text{cams}}} \log(g_i)^2 + b_i^2 \tag{A.34}$$

$$\frac{\partial \mathcal{L}_{\text{expo}}}{\partial g_i} = 2\frac{\log(g_i)}{g_i} \tag{A.35}$$

$$\frac{\partial \mathcal{L}_{\text{expo}}}{\partial b_i} = 2b_i \tag{A.36}$$

We also normalize the exposures towards one at each iteration:

$$g_i \longleftarrow g_i \left(\prod_{i<n_{\text{cams}}} g_j \right)^{\frac{-1}{n_{\text{cams}}}} \tag{A.37}$$

so that the product of the exposures g_i is 1 after normalization.

A.3 Optimization

We use the ADAM optimizer with $\beta_1 = 0.960$, $\beta_2 = 0.953$, with the following update rule:

$$\delta x = \frac{\partial \mathcal{L}}{\partial x} \tag{A.38}$$

$$g_1 \longleftarrow g_1 \beta_1 + \delta x (1 - \beta_1) \tag{A.39}$$

$$g_2 \longleftarrow \sqrt{(g_2)^2 \beta_2 + (\delta x)^2 (1 - \beta_2)} \tag{A.40}$$

$$x \longleftarrow x - \frac{g_1}{\sqrt{(g_2)^2 + 10^{-}8}} \lambda_{\text{learning rate}} \tag{A.41}$$

We weight the losses by the following parameters:

$$\lambda_{\text{learning rate}} = 0.05781 \tag{A.42}$$

$$\lambda_{\text{photo}} = 1 \tag{A.43}$$

$$\lambda_{\text{smooth } \alpha} = 0.00121 \tag{A.44}$$

$$\lambda_{\text{smooth } \beta_{0,0}} = 0.00216 \tag{A.45}$$

$$\lambda_{\text{smooth } \beta_{l,m}} = 0.00486 \tag{A.46}$$

$$\lambda_{\text{sample}} = 0.04063 \tag{A.47}$$

$$\lambda_{\text{SH parsimony}} = 0.00522 \tag{A.48}$$

$$\lambda_{\text{ballooning}} = 0.00002 \tag{A.49}$$

$$\lambda_{\text{mask}} = 0.05264 \tag{A.50}$$

We also weight the per-sample losses (λ_{photo} and λ_{sample}) by $\frac{1}{n_{\text{active cams}}+1}$ where $n_{\text{active cams}}$ is the number of cameras for which the transmittance to the voxel of interest is above zero.

B

Appendix of Chapter 4

B.1 Optimization and visualization loops

A slightly simplified version of our actual optimization procedure is described in alg. 1, matching the diagram in fig. 4.2. Function calls in red denote the invocation of a GPU kernel from the CPU. Some operations such as initializing or subdividing the grid, deleting or allocating voxels and computing the regularizations gradients require multiple kernel invocations that are not detailed here. We extract a mesh with marching cubes after each iteration for visualization purposes if needed. All kernel launches are fully asynchronous. Most of them run in strict launch order thanks to appropriate execution barriers (not shown here) while some of them are allowed to run concurrently when possible. For instance, the ADAM kernels may run simultaneously. A single CPU / GPU synchronization occurs at the end of each training iteration to monitor the loss, the number of extracted vertices and the number of active voxels and tiles. The photometric loss kernel is tasked to apply the color correction as well as to compute the loss and its per-pixel gradient. Note that we use no auto-differentiation, hard-coding partial derivatives instead for efficiency. Each forward pass kernel has a twin backward pass kernel, annotated with the "_backward" suffix.

After training is complete, we can visualize the results interactively according to alg. 2. We start by uploading the data to the GPU memory, then apply the smoothing kernel on the sdf and pre-compute the normal vectors. Afterwards, only two kernel invocations are necessary in the rendering loop, first to predict the per-voxel colors from the current viewpoint, and then to compute the volume rendering. The color correction is applied if the user wants to view the result from a particular camera's lens.

ALGORITHM 1: Optimization loop

```
foreach level of detail do
    init_or_subdivide_grid() ;                    // [Several kernels]
    foreach training iteration do
        clear_grads() ;                           // [Several kernels]
        smooth_sdf() ;                             // [1 thread/voxel]
        compute_normals() ;                        // [1 thread/voxel]
        batch ← select_views() ;
        foreach camera ∈ batch do
            color_prediction() ;                   // [1 thread/voxel]
            ray_marching() ;                       // [1 thread/pixel]
            photometric_loss() ;                   // [1 thread/pixel]
            ray_marching_backward() ;              // [1 thread/pixel]
            color_prediction_backward() ;          // [1 thread/voxel]
        end
        apply_regularizations() ;                 // [Several kernels]
        compute_normals_backward() ;               // [1 thread/voxel]
        smooth_sdf_backward() ;                    // [1 thread/voxel]
        adam_SDF() ;                               // [1 thread/voxel]
        adam_Features() ;                          // [1 thread/texel]
        adam_MLP() ;                              // [1 thread/weight]
        delete_or_alloc_voxels() ;                // [Several kernels]
        marching_cubes() ;                         // [1 thread/voxel]
    end
end
```

ALGORITHM 2: Visualization loop

```
load_data() ;
smooth_sdf() ;                                    // [1 thread/voxel]
compute_normals() ;                               // [1 thread/voxel]
while true do
    camera ← current_view();
    color_prediction() ;                          // [1 thread/voxel]
    ray_marching() ;                              // [1 thread/pixel]
    color_correction() ;                          // [1 thread/pixel]
    display() ;
end
```

B.2 Data structures comparison

A wide range of scene representations have been used in the literature, this short description will attempt to compare their most prominent features.

1. **Octrees** [150] and [53] fit an octree on an already optimized NeRF. Octrees present ideal sparsity but are complex to update. Furthermore, deep hierarchical structures imply many levels of indirection that can hurt rendering performance since the pointer stack already requires many registers [151].

2. **Dense grids** [30, 51] simply use dense grids that cannot scale beyond resolutions of a few hundred voxels, but can be very efficient at small resolutions.

3. **Factorized grids** [60,142,143] factorize a dense grid into outer products of global planes or vectors which removes the memory bottleneck of a dense grid but prevents local updates. [146] improves significantly over global factorization methods in terms of quality and model size by adopting local factorizations. However, it lacks a proper acceleration structure to allow for fast inference.

4. **Hash grids** [5,31] rely on hierarchical hash grids. An evolving occupancy grid restricts the sampling to non-empty regions so that the hash tables only need to be fitted close to the surface. Though practical, this strategy presents several downsides. First, each hash table has a fixed and predetermined size which means that the number of parameters cannot be increased dynamically to accommodate larger reconstructions. Second, the hash tables inherently hinder cache coherence which creates a sharp drop in performance when their size exceeds that of the GPU cache [5].

B.3 Volume renderings

Figures B.1 through B.13 show the volume rendering results of all the baselines on all the scenes of our dataset, in the same order as the tables 4.3, 4.4 and 4.5. See figures B.4 and B.11 for closeups and additional commentary.



Figure B.1: Qualitative rendering comparisons on the female "jea" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.

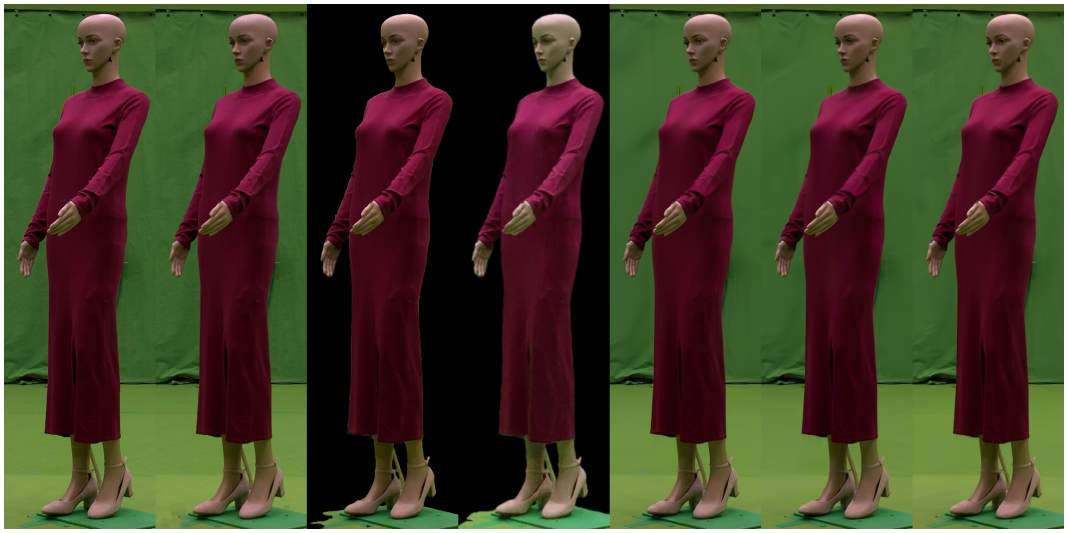

Figure B.2: Qualitative rendering comparisons on the female "opt1" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.

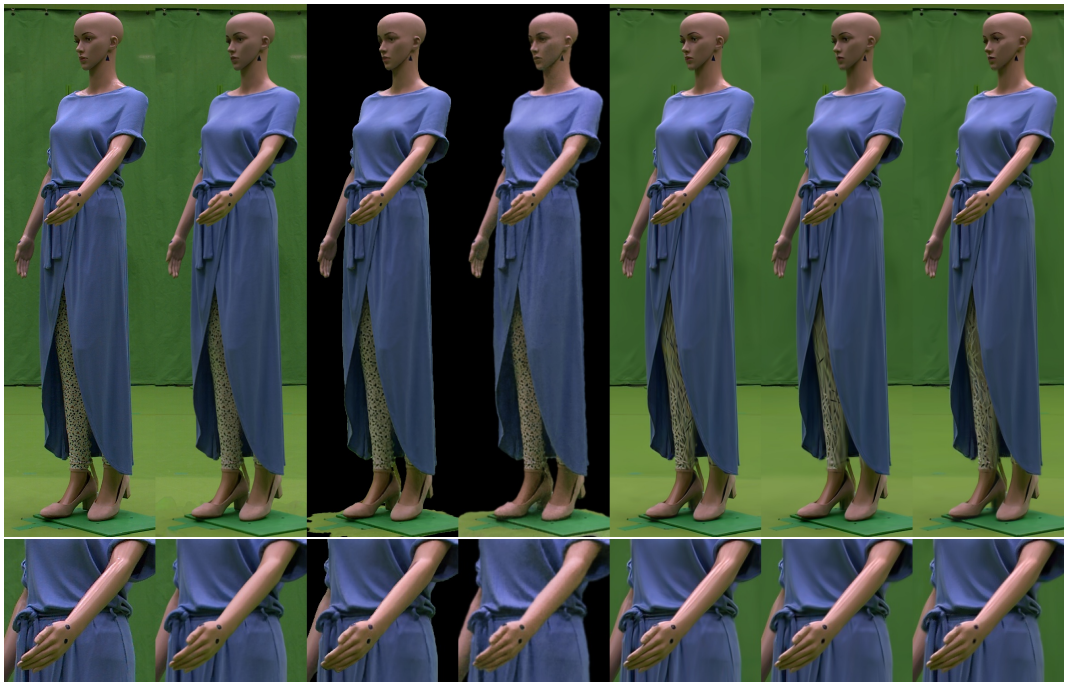

Figure B.3: Qualitative rendering comparisons on the female "opt2" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. The splatting based methods are particularly good at reconstructing high frequency specularities compared to the neural methods, with Voxurf having a noticeable advantage in that regard thanks to its larger MLP.

Figure B.4

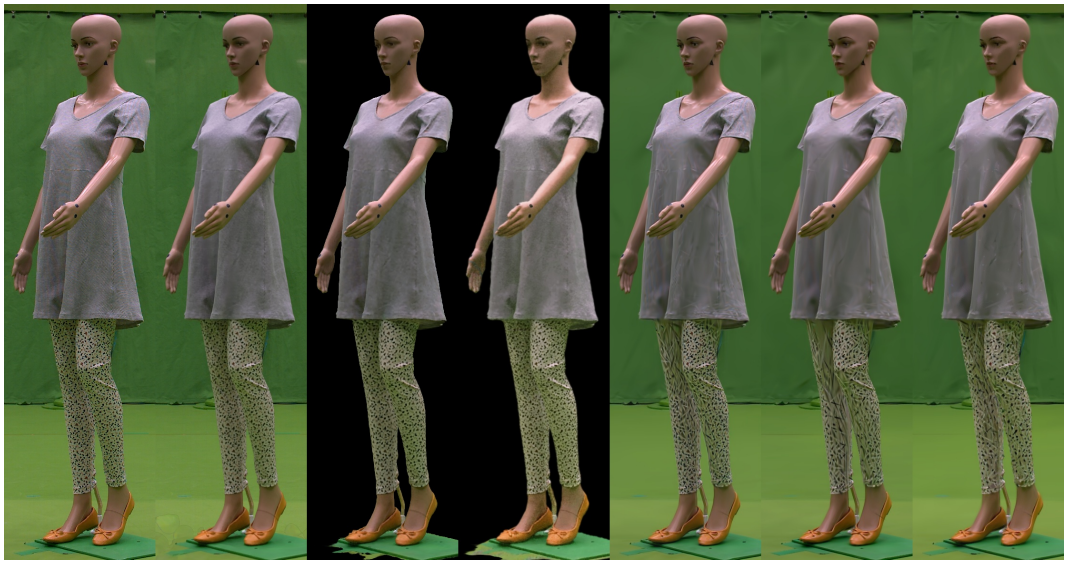

Figure B.5: Qualitative rendering comparisons on the female "opt3" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.



Figure B.6: Qualitative rendering comparisons on the female "sho" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF

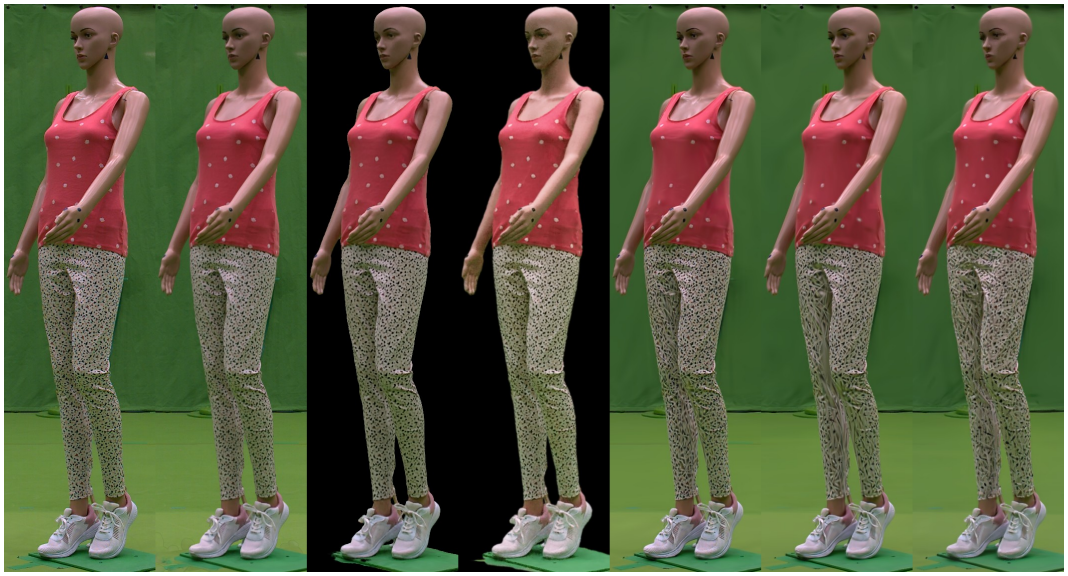

Figure B.7: Qualitative rendering comparisons on the female "tig" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.



Figure B.8: Qualitative rendering comparisons on the male "cos" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.

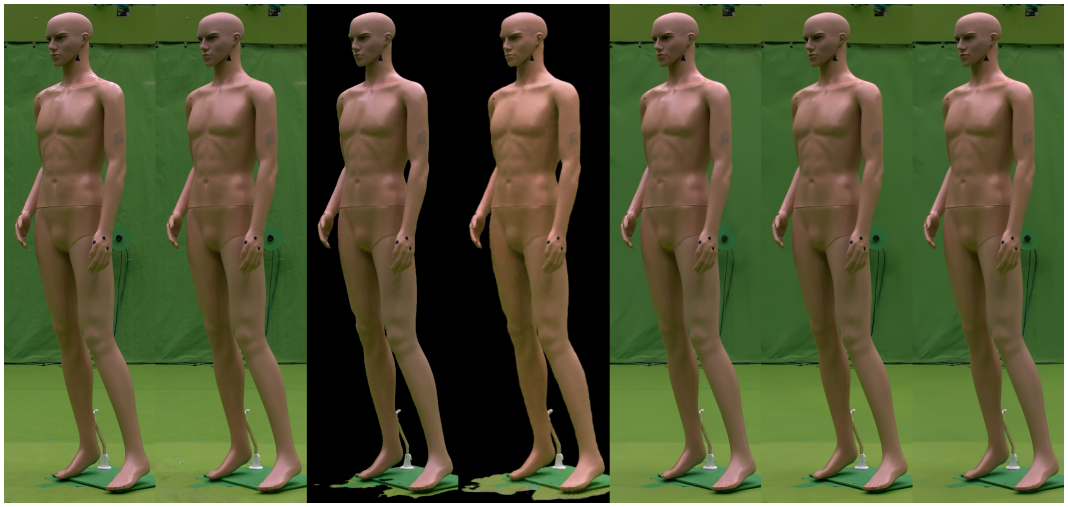

Figure B.9: Qualitative rendering comparisons on the male "naked" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.



Figure B.10: Qualitative rendering comparisons on the male "jea" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.



Figure B.11: Qualitative rendering comparisons on the male "opt" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF. We observe that NeuS2 is sometimes unable to explain the inter-camera color variations from some viewpoints whereas the other baselines do not seem impacted.

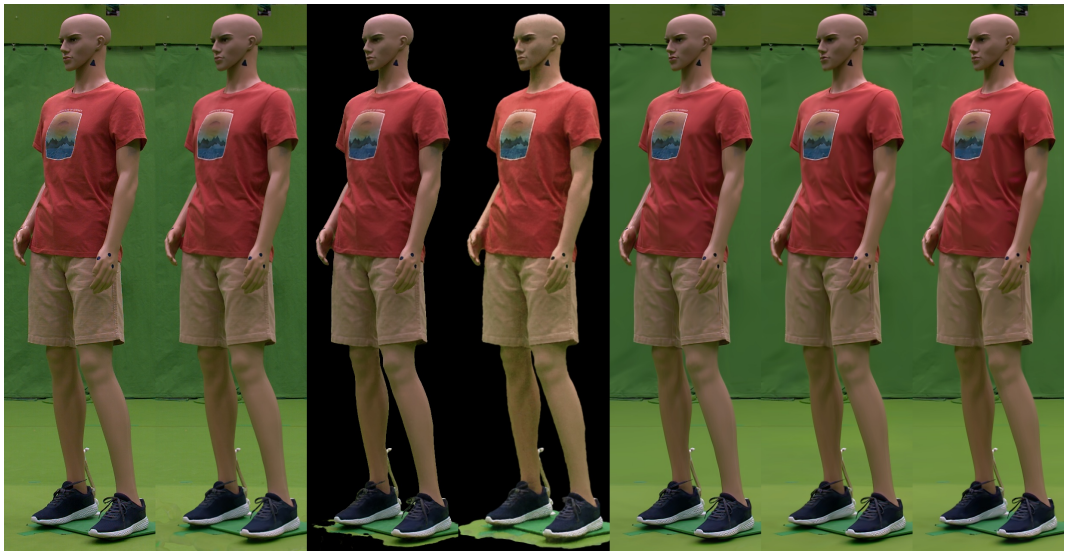

Figure B.12: Qualitative rendering comparisons on the male "sho" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.

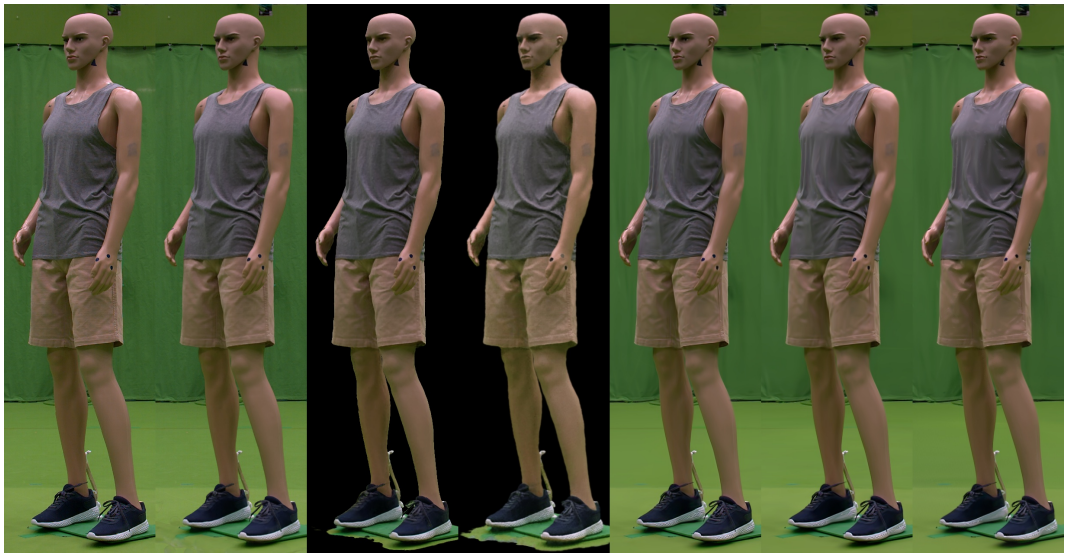

Figure B.13: Qualitative rendering comparisons on the male "tig" scene. Left to right: input image, Ours, Voxurf, Neus2, 3DGS, 2DGS, GOF.

C

Appendix of Chapter 5

C.1 Detailed hyper-parameters scheduling

The scheduling of the hyper-parameters is summarized in tables C.1, C.2, C.3 and C.4. A blank cell indicates that the value is identical to that of the row above. Values between brackets are linearly interpolated based on the current training iteration of the given level of detail. Our general strategy is to start with a high learning rate and high regularizations that are progressively halved during training. We lower the learning rate in the final level of detail to stabilize convergence. The learning rate is also linearly increased starting from zero during the first 50 iterations of each level of detail as a warm-up. We use stronger regularizations at the coarsest level of detail on DTU and BMVS since the initialization from a sphere is poorer compared to the visual hull initialization of the other two datasets. The MLP weights and probe features are trained with a lower learning rate compared to the spatial features and voxel sdf values, also for stability.

C.2 Detailed tables

Tables C.5 to C.17 contain detailed metrics from all our experiments. The dimensionality of the spatial features $\mathbf{F}_s$ is n_s and the dimensionality of the angular features $\mathbf{F}_a$ is n_a. We denote a training configuration with a triplet (n_s, n_a, l). A ✓-symbol denotes the training of per-camera bias vectors, an ✗-symbol is used otherwise.

Level of detail	Learning rate (Voxels)	Learning rate (MLP)	λ_{Eik}	λ_{sdf}	$\lambda_{\text{features}}$	λ_{normal}	λ_{probes}	Images per batch	Iterations
LOD 4	[0.025, 0.01]	0.01	[1.0, 0.1, 0.1]	[2.0, 0.2, 0.2]	[0.5, 0.05, 0.05]	[0.5, 0.05, 0.05]	[2.0, 0.2, 0.2]	8	3000
LOD 3			[0.2, 0.1]	[0.4, 0.2]	[0.1, 0.05]	[0.1, 0.05]	[0.4, 0.2]		
LOD 2									
LOD 1									1000
LOD 0	[0.01, 0.001]	[0.01, 0.001]						4	500

Table C.1: Schedule for DTU

Level of detail	Learning rate (Voxels)	Learning rate (MLP)	λ_{Eik}	λ_{sdf}	$\lambda_{\text{features}}$	λ_{normal}	λ_{probes}	Images per batch	Iterations
LOD 4	[0.1, 0.01]	0.01	[0.25, 0.025, 0.025]	[0.5, 0.05, 0.05]	[0.25, 0.025, 0.025]	[0.025, 0.0025, 0.0025]	[2.0, 0.2, 0.2]	8	4000
LOD 3			[0.05, 0.025]	[0.1, 0.05]	[0.05, 0.025]	[0.005, 0.0025]	[0.4, 0.2]		
LOD 2	[0.025, 0.01]								3000
LOD 1									1500
LOD 0	[0.01, 0.001]	[0.01, 0.001]						4	1000

Table C.2: Schedule for BlendedMVS

Level of detail	Learning rate (Voxels)	Learning rate (MLP)	λ_{Eik}	λ_{sdf}	$\lambda_{\text{features}}$	λ_{normal}	λ_{probes}	Images per batch	Iterations
LOD 4	[0.025, 0.01]	0.01	[0.05, 0.025]	[0.2, 0.1]	[0.1, 0.05]	0	[0.4, 0.2]	4	1500
LOD 3									
LOD 2									
LOD 1									
LOD 0	[0.01, 0.001]	[0.01, 0.001]	0.0125	0.05	0.025	0.0063	0.1		1000

Table C.3: Schedule for MVMannequins

Level of detail	Learning rate (Voxels)	Learning rate (MLP)	λ_{Eik}	λ_{sdf}	$\lambda_{\text{features}}$	λ_{normal}	λ_{probes}	Images per batch	Iterations
LOD 6	[0.05, 0.025]	0.01	[0.05, 0.025]	[0.2, 0.1]	[0.025, 0.0125]	0	[0.4,0.2]	8	500
LOD 5									1000
LOD 4									1500
LOD 3									
LOD 2						[0.025, 0.0125]			1000
LOD 1	[0.01, 0.0025]	[0.01, 0.001]	0.0125	0.05	0.0063	0.0125	0.1	4	
LOD 0	[0.01, 0.001]	0.001				0.025			500

Table C.4: Schedule for ActorsHQ

		kinette								kino					
Chamfer (mm)	Mean	cos	naked	jea	opt1	opt2	opt3	sho	tig	cos	naked	jea	opt	sho	tig
(4,4,4)✗	**1.04**	**1.51**	**0.54**	**1.09**	**1.14**	**1.75**	**1.22**	**0.95**	**0.66**	**1.53**	0.60	**0.75**	**1.51**	**0.67**	**0.67**
MMH	1.15	1.55	**0.54**	1.10	1.16	2.12	1.45	1.07	0.68	1.77	**0.59**	0.80	1.72	0.75	0.75
Voxurf	1.59	1.70	1.28	1.62	1.62	2.06	1.61	1.48	0.98	2.60	1.30	1.27	2.54	1.12	1.06
Neus2	2.13	3.71	1.43	2.10	2.13	2.90	2	2.12	1.25	3.55	1.61	1.64	2.61	1.43	1.37
Colmap	3.51	2.69	4.27	2.99	4.59	4.18	2.81	3.71	2.50	4.11	4.20	2.48	4.34	3.09	3.23
2DGS	3.35	5.05	2.27	8.30	3.25	3.40	3.63	2.66	3.52	4.18	2.25	1.75	2.80	1.98	1.87

Table C.5: MVMannequins per-scene Chamfer (mm)

		kinette								kino					
PSNR (db)	Mean	cos	naked	jea	opt1	opt2	opt3	sho	tig	cos	naked	jea	opt	sho	tig
(4,4,4)✗	**36.81**	**29.48**	**40.61**	**30.73**	**40.48**	**38.69**	**34.13**	**37.70**	**31.74**	**39.64**	**41.57**	**35.80**	**39.29**	**37.50**	**37.99**
MMH	36.33	29.24	40.03	30.55	39.94	38.01	33.82	37.18	31.38	39.22	40.88	35.25	38.59	37.11	37.49
Voxurf	35.51	28.39	37.51	30.20	39.26	37.34	32.82	36.73	30.81	38.17	40.09	34.45	37.57	36.69	37.16
Neus2	34.22	28.09	37.23	29.44	36.98	35.55	32.58	34.98	30.14	36.65	38.05	33.39	35.78	35.01	35.23
2DGS	34.89	27.26	37.92	28.88	37.62	36.84	32.22	36.24	29.86	37.97	38.82	34.99	37.28	36.14	36.50

Table C.6: MVMannequins per-scene PSNR

		kinette								kino					
Chamfer (mm)	Mean	cos	naked	jea	opt1	opt2	opt3	sho	tig	cos	naked	jea	opt	sho	tig
(8,8,4)✗	**1.03**	**1.48**	**0.54**	**1.08**	1.12	**1.70**	1.24	**0.93**	**0.65**	1.52	**0.59**	0.74	1.51	**0.66**	0.67
(12,12,4)✗	1.04	1.52	0.54	1.09	1.13	1.72	1.23	0.96	0.66	1.52	0.59	**0.74**	1.52	0.68	**0.67**
(4,4,1)✗	1.30	1.56	1.22	1.22	1.25	1.78	1.62	1.50	1.05	1.78	1.16	0.81	1.48	0.86	0.95
(4,4,2)✗	1.04	1.48	0.59	1.09	**1.10**	1.71	**1.22**	0.94	0.67	**1.51**	0.64	0.74	**1.45**	0.69	0.69
(4,4,3)✗	1.04	1.49	0.56	1.09	1.12	1.73	1.22	0.96	0.66	1.52	0.60	0.75	1.50	0.68	0.68
(4,4,4)✗	1.04	1.51	0.54	1.09	1.14	1.75	1.22	0.95	0.66	1.53	0.60	0.75	1.51	0.67	0.67

Table C.7: Detailed Ablation Table. MVMannequins per-scene Chamfer (mm)

		kinette								kino					
PSNR (db)	Mean	cos	naked	jea	opt1	opt2	opt3	sho	tig	cos	naked	jea	opt	sho	tig
(8,8,4)✗	36.90	29.56	40.76	**30.80**	40.62	38.80	34.23	37.76	**31.86**	39.76	41.66	35.88	39.35	37.56	38.07
(12,12,4)✗	**36.93**	**29.57**	**40.79**	30.79	**40.63**	**38.82**	**34.24**	**37.78**	31.83	**39.79**	**41.73**	**35.88**	**39.41**	**37.58**	**38.20**
(4,4,1)✗	35.84	29.24	38.61	30.52	39.03	37.47	33.55	36.24	31.24	38.81	39.46	35.37	38.68	36.58	36.98
(4,4,2)✗	36.53	29.31	40.09	30.62	40.07	38.38	33.95	37.42	31.55	39.35	41.06	35.67	39.01	37.26	37.71
(4,4,3)✗	36.68	29.38	40.39	30.66	40.30	38.54	34.05	37.56	31.62	39.50	41.37	35.72	39.14	37.41	37.86
(4,4,4)✗	36.81	29.48	40.61	30.73	40.48	38.69	34.13	37.70	31.74	39.64	41.57	35.80	39.29	37.50	37.99

Table C.8: Detailed Ablation Table. MVMannequins per-scene PSNR

PSNR	Resolution	Mean	A1S1	A2S1	A3S1	A4S1	A5S1	A6S1	A7S1	A8S1	A1S2	A4S2	A5S2	A6S2	A8S2
(4,4,4)✗	r/1	**37.48**	**37.64**	**38.21**	**37.58**	**35.78**	**38.36**	36.80	**37.80**	**38.09**	**37.91**	**35.59**	**38.56**	36.59	**38.32**
(4,4,4)✗	r/2	36.62	36.86	36.90	36.59	34.55	37.54	36.39	37.03	37.28	37.19	34.32	37.75	36.24	37.47
(4,4,4)✗	r/4	34.75	35.43	34.05	35.21	32.08	35.59	34.96	35.41	35.16	35.67	32.14	35.94	34.75	35.34
Voxurf	r/2	36.56	37.04	36.93	36.17	34.58	36.96	**36.85**	36.69	36.99	37.12	34.33	37.53	**36.81**	37.31
Neus2	r/2	34.53	35.22	34.67	33.28	32.50	35.20	33.86	35.43	35.41	35.71	32.52	35.38	34.02	35.72

Table C.9: ActorsHQ per-scene PSNR

Chamfer (mm)	Mean	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
(4,4,4)✓	**0.68**	0.65	0.74	0.34	**0.34**	1.02	0.71	**0.62**	1.34	**0.94**	0.70	**0.53**	0.96	0.36	0.45	0.47
(8,8,4)✓	0.71	0.56	**0.71**	0.34	**0.34**	1.38	0.74	0.64	1.34	0.95	0.70	0.54	1.07	**0.35**	0.45	0.47
Voxurf	0.73	0.76	0.72	0.67	**0.34**	0.95	**0.62**	0.79	1.35	0.96	0.74	0.61	1.17	**0.35**	**0.44**	0.49
Neus2	0.80	0.55	0.81	1.66	0.38	**0.92**	0.72	0.79	1.31	1.07	0.80	0.61	**0.89**	0.46	0.52	0.58
2DGS	0.76	**0.47**	0.82	**0.32**	0.36	1.06	0.89	0.81	**1.30**	1.23	**0.66**	0.65	1.34	0.42	0.66	**0.46**

Table C.10: DTU per-scene Chamfer (mm)

PSNR (db)	Mean	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
(4,4,4)✓	37.03	35.58	30.59	35.59	35.56	38.89	38.06	34.81	39.90	33.97	38.84	40.03	36.23	34.90	40.87	41.63
(8,8,4)✓	**37.74**	**36.73**	**31.53**	**36.31**	**36.74**	39.38	38.93	35.05	40.25	**34.69**	**39.51**	**40.45**	**36.79**	**35.60**	41.59	**42.49**
Voxurf	37.08	34.97	30.70	33.82	35.02	**39.45**	**39.22**	35.48	**41.03**	34.35	38.78	39.43	35.36	35.20	**41.79**	41.69
Neus2	36.00	34.57	29.82	34.30	34.64	37.93	36.87	33.79	38.95	32.79	38.13	38.37	35.14	34.37	39.93	40.32
2DGS	36.03	35.01	30.61	34.47	33.77	38.27	36.11	**35.84**	39.53	34.26	38.33	37.86	34.82	33.46	39.10	39.05

Table C.11: DTU per-scene PSNR

Chamfer (mm)	Mean	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
(4,4,4)✗	0.71	0.68	0.82	**0.34**	0.35	1.20	0.76	**0.59**	1.34	0.91	0.74	0.57	**0.91**	0.39	0.50	0.50
(8,8,4)✓	0.71	**0.56**	**0.71**	**0.34**	**0.34**	1.38	0.74	0.64	1.34	0.95	0.70	0.54	1.07	**0.35**	0.45	**0.47**
(12,12,4)✓	0.70	0.58	0.74	**0.34**	**0.34**	1.37	0.73	0.66	1.32	**0.87**	0.71	0.54	0.93	**0.35**	0.45	**0.47**
(4,4,1)✓	0.85	0.64	0.80	0.35	**0.34**	1.79	0.71	0.77	**1.30**	1.12	**0.69**	**0.52**	2.34	0.41	**0.43**	**0.47**
(4,4,2)✓	0.69	0.64	0.74	**0.34**	**0.34**	1.09	**0.69**	0.67	1.33	1.01	**0.69**	0.53	1.02	0.37	0.44	**0.47**
(4,4,3)✓	**0.67**	0.64	0.75	**0.34**	**0.34**	1.04	0.70	0.62	1.33	0.95	**0.69**	0.53	0.92	0.36	0.44	**0.47**
(4,4,4)✓	0.68	0.65	0.74	**0.34**	**0.34**	**1.02**	0.71	0.62	1.34	0.94	0.70	0.53	0.96	0.36	0.45	**0.47**

Table C.12: Detailed Ablation Table. DTU per-scene Chamfer (mm)

PSNR (db)	Mean	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
(4,4,4)✗	36.18	34.73	29.81	34.87	34.52	38.23	36.38	34.52	39.35	33.17	38.34	38.80	36	34.03	39.62	40.38
(8,8,4)✓	37.74	36.73	31.53	36.31	36.74	39.38	38.93	35.05	40.25	34.69	39.51	**40.45**	36.79	35.60	41.59	42.49
(12,12,4)✓	**38.03**	**37.20**	**31.96**	**36.64**	**36.95**	**39.55**	**39.33**	**35.71**	**40.42**	**35.27**	**39.58**	40.32	**36.97**	**35.89**	**41.97**	**42.75**
(4,4,1)✓	35.92	35.13	29.81	34.97	34.66	36.40	36.77	32.90	38.86	32.29	38.03	39.43	34.36	34	40.36	40.92
(4,4,2)✓	36.44	35.25	29.97	35.31	35.27	37.66	37.32	33.62	39.48	32.98	38.53	39.71	35.49	34.39	40.53	41.11
(4,4,3)✓	36.76	35.36	30.27	35.40	35.46	38.46	37.66	34.40	39.71	33.59	38.67	39.80	35.95	34.67	40.69	41.32
(4,4,4)✓	37.03	35.58	30.59	35.59	35.56	38.89	38.06	34.81	39.90	33.97	38.84	40.03	36.23	34.90	40.87	41.63

Table C.13: Detailed Ablation Table. DTU per-scene PSNR

Chamfer	Mean	dog	bear	clock	durian	man	sculpture	stone	jade
(4,4,4)✓	2.36	2.51	2.27	1.90	3.63	**1.79**	**1.61**	**1.30**	3.88
(8,8,4)✓	**2.21**	**2.24**	**2.03**	**1.69**	3.26	1.81	**1.61**	1.36	**3.71**
Voxurf	2.64	2.28	2.20	1.88	**2.98**	2.11	1.75	3.96	3.99
Neus2	2.93	2.78	2.71	2.63	4.23	2.25	2.50	1.91	4.43

Table C.14: BlendedMVS per-scene Chamfer

PSNR	Mean	dog	bear	clock	durian	man	sculpture	stone	jade
(4,4,4)✓	35.19	35.49	30.55	34.68	31.28	42.94	40.80	30.84	34.92
(8,8,4)✓	**35.89**	**36.30**	**30.96**	**35.42**	**31.65**	**43.72**	**41.34**	31.01	**36.69**
Voxurf	35.11	35.24	30.88	34.49	29.69	43.35	41.06	30.23	35.93
Neus2	33.62	34.56	29.99	31.04	29.21	40.88	39.10	**31.36**	32.79

Table C.15: BlendedMVS per-scene PSNR

Chamfer	Mean	dog	bear	clock	durian	man	sculpture	stone	jade
(4,4,4)✗	2.47	2.18	2.71	2.05	3.81	1.97	1.75	1.36	3.95
(8,8,4)✓	**2.21**	2.24	2.03	**1.69**	**3.26**	1.81	1.61	1.36	**3.71**
(12,12,4)✓	2.31	**2.10**	2.31	1.86	3.58	1.96	**1.59**	1.34	3.74
(4,4,1)✓	2.58	3.27	**2.02**	2.44	3.88	1.92	1.88	1.33	3.93
(4,4,2)✓	2.35	2.59	2.03	2.04	3.66	1.80	1.62	**1.29**	3.75
(4,4,3)✓	2.36	2.56	2.35	1.94	3.59	**1.79**	1.64	1.31	3.74
(4,4,4)✓	2.36	2.51	2.27	1.90	3.63	**1.79**	1.61	1.30	3.88

Table C.16: Detailed Ablation Table. BlendedMVS per-scene Chamfer

(4,4,4)✗	34.76	35.71	30.35	33.76	31	42.65	40.08	30.84	33.72
(8,8,4)✓	35.89	36.30	30.96	35.42	31.65	43.72	41.34	31.01	36.69
(12,12,4)✓	**36.17**	**36.50**	**30.97**	**36.55**	**32.12**	**43.75**	**41.41**	**31.04**	**37.00**
(4,4,1)✓	34.44	34.48	30.34	32.81	31.12	42.44	39.89	30.75	33.72
(4,4,2)✓	34.86	35	30.53	33.85	31.38	42.71	40.40	30.82	34.19
(4,4,3)✓	35.05	35.18	30.64	34.34	31.36	42.90	40.69	30.82	34.48
(4,4,4)✓	35.19	35.49	30.55	34.68	31.28	42.94	40.80	30.84	34.92

Table C.17: Detailed Ablation Table. BlendedMVS per-scene PSNR

Bibliography

[1] "Kinovis. 2024. Kinovis, Inria 4D Modeling Multi-Camera Platform.." https://kinovis.inria.fr.

[2] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Trans. Graph.*, vol. 34, July 2015.

[3] M. Işık, M. Rünz, M. Georgopoulos, T. Khakhulin, J. Starck, L. Agapito, and M. Nießner, "HumanRF: High-Fidelity Neural Radiance Fields for Humans in Motion," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–12, 2023.

[4] T. Yu, Z. Zheng, K. Guo, P. Liu, Q. Dai, and Y. Liu, "Function4d: Real-time human volumetric capture from very sparse consumer rgbd sensors," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5746–5756, 2021.

[5] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Trans. Graph.*, vol. 41, pp. 102:1–102:15, July 2022.

[6] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-nerf: Point-based neural radiance fields," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5438–5448, 2022.

[7] S. Govindarajan, D. Rebain, K. M. Yi, and A. Tagliasacchi, "Radiant Foam: Real-Time Differentiable Ray Tracing," *arXiv:2502.01157*, 2025.

[8] V. Leroy, J.-S. Franco, and E. Boyer, "Volume Sweeping: Learning Photoconsistency for Multi-View Shape Reconstruction," *Int. J. Comput. Vision*, vol. 129, p. 284–299, Feb. 2021.

[9] M. Armando, L. Boissieux, E. Boyer, J.-S. Franco, M. Humenberger, C. Legras, V. Leroy, M. Marsot, J. Pansiot, S. Pujades, R. Rekik Dit Nekhili, G. Rogez, A. Swamy, and S. Wuhrer, "4DHumanOutfit: a multi-subject 4D dataset of human motion sequences in varying outfits exhibiting large displacements," *Computer Vision and Image Understanding*, vol. 237, p. 103836, Dec. 2023.

[10] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: learning neural implicit surfaces by volume rendering for multi-view reconstruction," in *Proceedings of the 35th International Conference*

on Neural Information Processing Systems, NIPS '21, (Red Hook, NY, USA), Curran Associates Inc., 2021.

[11] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, p. 163–169, Aug. 1987.

[12] B. Toussaint, L. Boissieux, D. Thomas, E. Boyer, and F. Jean-Sébastien, "Millimetric Human Surface Capture in Minutes," in *SIGGRAPH Asia 2024 - 17th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia*, (Tokyo, Japan), pp. 1–12, ACM, Dec. 2024.

[13] NVidia, "What Is Photogrammetry?." https://blogs.nvidia.com/blog/what-is-photogrammetry/. Accessed: 2025-05-26.

[14] "Inc. Epic Games. 2022. RealityCapture.." https://www.capturingreality.com. Accessed: 2025-03-10.

[15] "4DViews." https://www.4dviews.com/. Accessed: 2025-05-26.

[16] "RenderPeople.." https://renderpeople.com.

[17] "Volucap." https://volucap.com/. Accessed: 2025-05-26.

[18] "Meta Human." https://www.unrealengine.com/en-US/metahuman. Accessed: 2025-05-26.

[19] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black, "Expressive body capture: 3D hands, face, and body from a single image," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 10975–10985, 2019.

[20] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pp. 851–866, 2023.

[21] P. Palafox, A. Božič, J. Thies, M. Nießner, and A. Dai, "Npms: Neural parametric models for 3d deformable shapes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12695–12705, 2021.

[22] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv preprint arXiv:2209.14988*, 2022.

[23] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20697–20709, 2024.

[24] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. New York, NY, USA: Cambridge University Press, 2 ed., 2003.

[25] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *ECCV*, 2020.

[27] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, vol. 42, July 2023.

[28] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields," *ICCV*, 2023.

[29] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin, "Neuralangelo: High-Fidelity Neural Surface Reconstruction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[30] T. Wu, J. Wang, X. Pan, X. Xu, C. Theobalt, Z. Liu, and D. Lin, "Voxurf: Voxel-based Efficient and Accurate Neural Surface Reconstruction," in *International Conference on Learning Representations (ICLR)*, 2023.

[31] Y. Wang, Q. Han, M. Habermann, K. Daniilidis, C. Theobalt, and L. Liu, "NeuS2: Fast Learning of Neural Implicit Surfaces for Multi-view Reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[32] A. Mustafa, M. Volino, J.-Y. Guillemaut, and A. Hilton, "4D Temporally Coherent Light-field Video," in *3DV*, 2017.

[33] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, pp. 519–528, 2006.

[34] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, "Large Scale Multi-view Stereopsis Evaluation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 406–413, 2014.

[35] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building Rome in a day," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 72–79, 2009.

[36] "IOI volumetric capture cameras." https://www.ioindustries.com/volumetric-capture. Accessed: 2025-05-26.

[37] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, July 2017.

[38] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields," *CVPR*, 2022.

[39] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," *ACM Transactions on Graphics (ToG)*, vol. 37, no. 6, pp. 1–15, 2018.

[40] S. Sinha, D. Steedly, and R. Szeliski, "Piecewise planar stereo for image-based rendering," in *2009 International Conference on Computer Vision*, pp. 1881–1888, 2009.

[41] C. Reiser, R. Szeliski, D. Verbin, P. Srinivasan, B. Mildenhall, A. Geiger, J. Barron, and P. Hedman, "MERF: Memory-Efficient Radiance Fields for Real-time View Synthesis in Unbounded Scenes," *ACM Trans. Graph.*, vol. 42, July 2023.

[42] C. Sun, J. Choe, C. Loop, W.-C. Ma, and Y.-C. F. Wang, "Sparse Voxels Rasterization: Real-time High-fidelity Radiance Field Rendering," *arXiv preprint arXiv:2412.04459*, 2024.

[43] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural Volumes: Learning Dynamic Renderable Volumes from Images," *ACM Trans. Graph.*, vol. 38, pp. 65:1–65:14, July 2019.

[44] F. Darmon, B. Bascle, J. Devaux, P. Monasse, and M. Aubry, "Improving neural implicit surfaces geometry with patch warping," in *CVPR*, 2022.

[45] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman, "Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[46] Q. Fu, Q. Xu, Y.-S. Ong, and W. Tao, "Geo-Neus: Geometry-Consistent Neural Implicit Surfaces Learning for Multi-view Reconstruction," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[47] A. Guédon and V. Lepetit, "SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering," *CVPR*, 2024.

[48] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2D Gaussian Splatting for Geometrically Accurate Radiance Fields," in *SIGGRAPH 2024 Conference Papers*, Association for Computing Machinery, 2024.

[49] Z. Yu, T. Sattler, and A. Geiger, "Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes," *ACM Transactions on Graphics*, 2024.

[50] D. Chen, H. Li, W. Ye, Y. Wang, W. Xie, S. Zhai, N. Wang, H. Liu, H. Bao, and G. Zhang, "PGSR: Planar-based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, pp. 1–12, 01 2024.

[51] C. Sun, M. Sun, and H. Chen, "Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction," in *CVPR*, 2022.

[52] W. Cheng, R. Chen, W. Yin, S. Fan, K. Chen, H. He, H. Luo, Z. Cai, J. Wang, Y. Gao, Z. Yu, Z. Lin, D. Ren, L. Yang, Z. Liu, C. C. Loy, C. Qian, W. Wu, D. Lin, B. Dai, and K.-Y. Lin, "DNA-Rendering: A Diverse Neural Actor Repository for High-Fidelity Human-centric Rendering," *arXiv preprint*, vol. arXiv:2307.10173, 2023.

[53] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5752–5761, 2021.

[54] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking Neural Radiance Fields for Real-Time View Synthesis," *ICCV*, 2021.

[55] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs," in *International Conference on Computer Vision (ICCV)*, 2021.

[56] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi, "MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures," in *The Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[57] D. Duckworth, P. Hedman, C. Reiser, P. Zhizhin, J.-F. Thibert, M. Lučić, R. Szeliski, and J. T. Barron, "Smerf: Streamable memory efficient radiance fields for real-time large-scene exploration," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 4, pp. 1–13, 2024.

[58] C. Reiser, S. Garbin, P. Srinivasan, D. Verbin, R. Szeliski, B. Mildenhall, J. Barron, P. Hedman, and A. Geiger, "Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 4, pp. 1–14, 2024.

[59] NVidia, "NVIDIA RTX A6000." https://www.nvidia.com/en-us/design-visualization/rtx-a6000/. Accessed: 2025-05-26.

[60] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "TensoRF: Tensorial Radiance Fields," in *European Conference on Computer Vision (ECCV)*, 2022.

[61] T. Takikawa, A. Evans, J. Tremblay, T. Müller, M. McGuire, A. Jacobson, and S. Fidler, "Variable bitrate neural fields," in *ACM SIGGRAPH 2022 Conference Proceedings*, pp. 1–9, 2022.

[62] S. Yuan and H. Zhao, "Slimmerf: Slimmable radiance fields," in *2024 International Conference on 3D Vision (3DV)*, pp. 64–74, IEEE, 2024.

[63] J. Starck and A. Hilton, "Surface capture for performance-based animation," *IEEE computer graphics and applications*, vol. 27, no. 3, pp. 21–31, 2007.

[64] D. Vlasic, P. Peers, I. Baran, P. Debevec, J. Popović, S. Rusinkiewicz, and W. Matusik, "Dynamic shape capture using multi-view photometric stereo," in *ACM SIGGRAPH Asia 2009 papers*, pp. 1–11, 2009.

[65] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi, "Motion2fusion: Real-time volumetric performance capture," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 6, pp. 1–16, 2017.

[66] V. Leroy, J.-S. Franco, and E. Boyer, "Volume Sweeping: Learning Photoconsistency for Multi-View Shape Reconstruction," *International Journal of Computer Vision*, vol. 129, pp. 284–299, Feb. 2021.

[67] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, *et al.*, "Fusion4d: Real-time performance capture of challenging scenes," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 4, pp. 1–13, 2016.

[68] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black, "Dyna: A model of dynamic human shape in motion," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–14, 2015.

[69] W. Xu, A. Chatterjee, M. Zollhöfer, H. Rhodin, D. Mehta, H.-P. Seidel, and C. Theobalt, "MonoPerfCap: Human Performance Capture From Monocular Video," *ACM Trans. Graph.*, vol. 37, pp. 27:1–27:15, May 2018.

[70] M. Habermann, W. Xu, M. Zollhoefer, G. Pons-Moll, and C. Theobalt, "Livecap: Real-time human performance capture from monocular video," *ACM Transactions On Graphics (TOG)*, vol. 38, no. 2, pp. 1–17, 2019.

[71] M. Habermann, W. Xu, M. Zollhofer, G. Pons-Moll, and C. Theobalt, "Deepcap: Monocular human performance capture using weak supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5052–5063, 2020.

[72] A. Caliskan, A. Mustafa, and A. Hilton, "Temporal consistency loss for high resolution textured and clothed 3d human reconstruction from monocular video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1780–1790, 2021.

[73] A. Mustafa, A. Caliskan, L. Agapito, and A. Hilton, "Multi-person implicit reconstruction from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14474–14483, 2021.

[74] Z. Zhang, "Microsoft Kinect Sensor and Its Effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, 2012.

[75] R. J. Woodham, "Photometric Method for Determining Shape from Shading," tech. rep., CAN, 1984.

[76] C. Hernandez, G. Vogiatzis, G. J. Brostow, B. Stenger, and R. Cipolla, "Non-rigid Photometric Stereo with Colored Lights," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007.

[77] M. Bleyer, C. Rhemann, and C. Rother, "PatchMatch Stereo - Stereo Matching with Slanted Support Windows," in *British Machine Vision Conference*, 2011.

[78] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011.

[79] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[80] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, *VolumeDeform: Real-Time Volumetric Non-rigid Reconstruction*, pp. 362–379. Cham: Springer International Publishing, 2016.

[81] M. Zollhöfer, A. Dai, M. Innmann, C. Wu, M. Stamminger, C. Theobalt, and M. Nießner, "Shading-based refinement on volumetric signed distance functions," *ACM Transactions on Graphics (ToG)*, vol. 34, no. 4, pp. 1–14, 2015.

[82] M. Pollefeys, R. Koch, and L. Van Gool, "Self-Calibration and Metric Reconstruction in Spite of Varying and Unknown Internal Camera Parameters.," pp. 90–95, 01 1998.

[83] Q.-T. Luong and O. D. Faugeras, "Self-Calibration of a Moving Camera from PointCorrespondences and Fundamental Matrices," *Int. J. Comput. Vision*, vol. 22, p. 261–289, Mar. 1997.

[84] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[85] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 2, pp. 150–162, 1994.

[86] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[87] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, July 2013.

[88] R. Martin-Brualla, R. Pandey, S. Yang, P. Pidlypenskyi, J. Taylor, J. Valentin, S. Khamis, P. Davidson, A. Tkach, P. Lincoln, *et al.*, "Lookingood: Enhancing performance capture with real-time neural re-rendering," *arXiv preprint arXiv:1811.05029*, 2018.

[89] F. Prada, M. Kazhdan, M. Chuang, A. Collet, and H. Hoppe, "Spatiotemporal atlas parameterization for evolving meshes," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–12, 2017.

[90] H. Chen, B. Miller, and I. Gkioulekas, "3D Reconstruction with Fast Dipole Sums," *ACM Trans. Graph.*, vol. 43, Nov. 2024.

[91] Y. Furukawa, C. Hernández, *et al.*, "Multi-view stereo: A tutorial," *Foundations and trends® in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, 2015.

[92] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila, "Modular primitives for high-performance differentiable rendering," *ACM Trans. Graph.*, vol. 39, Nov. 2020.

[93] J. Munkberg, J. Hasselgren, T. Shen, J. Gao, W. Chen, A. Evans, T. Müller, and S. Fidler, "Extracting Triangular 3D Models, Materials, and Lighting From Images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8280–8290, June 2022.

[94] S. Liu, Y. Zhang, S. Peng, B. Shi, M. Pollefeys, and Z. Cui, "DIST: Rendering Deep Implicit Signed Distance Function with Differentiable Sphere Tracing," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[95] D. Verbin, P. P. Srinivasan, P. Hedman, B. Mildenhall, B. Attal, R. Szeliski, and J. T. Barron, "NeRF-Casting: Improved View-Dependent Appearance with Consistent Reflections," *arXiv preprint arXiv:2405.14871*, 2024.

[96] Sara Fridovich-Keil and Alex Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance Fields without Neural Networks," in *CVPR*, 2022.

[97] J. Hasselgren, N. Hofmann, and J. Munkberg, "Shape, light, and material decomposition from images using monte carlo rendering and denoising," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22856–22869, 2022.

[98] K. Zhang, F. Luan, Q. Wang, K. Bala, and N. Snavely, "PhySG: Inverse rendering with spherical gaussians for physics-based material editing and relighting," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[99] S. Saito, G. Schwartz, T. Simon, J. Li, and G. Nam, "Relightable Gaussian Codec Avatars," in *CVPR*, 2024.

[100] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, "Mitsuba 2: A Retargetable Forward and Inverse Renderer," *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, vol. 38, Dec. 2019.

[101] Z. Huang, Z. Liang, and K. Jia, "Sur2f: A Hybrid Representation for High-Quality and Efficient Surface Reconstruction from Multi-view Images," in *European Conference on Computer Vision*, pp. 1–18, Springer, 2024.

[102] S. Govindarajan, D. Rebain, K. M. Yi, and A. Tagliasacchi, "Radiant Foam: Real-Time Differentiable Ray Tracing," *arXiv preprint arXiv:2502.01157*, 2025.

[103] S. Bangaru, J. Michel, K. Mu, G. Bernstein, T.-M. Li, and J. Ragan-Kelley, "Systematically Differentiating Parametric Discontinuities," *ACM Trans. Graph.*, vol. 40, no. 107, pp. 107:1–107:17, 2021.

[104] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable Monte Carlo ray tracing through edge sampling," *ACM Trans. Graph.*, vol. 37, Dec. 2018.

[105] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), p. 31–42, Association for Computing Machinery, 1996.

[106] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), p. 43–54, Association for Computing Machinery, 1996.

[107] J. R. Bergen and E. H. Adelson, "The plenoptic function and the elements of early vision," *Computational models of visual processing*, vol. 1, no. 8, p. 3, 1991.

[108] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995.

[109] M. Nimier-David, T. Müller, A. Keller, and W. Jakob, "Unbiased Inverse Volume Rendering with Differential Trackers," *ACM Trans. Graph.*, vol. 41, pp. 44:1–44:20, July 2022.

[110] B. Nicolet, F. Wechsler, J. Madrid-Wolff, C. Moser, and W. Jakob, "Inverse Rendering for Tomographic Volumetric Additive Manufacturing," *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, vol. 43, pp. 1–17, Dec. 2024.

[111] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields," *ICCV*, 2021.

[112] C. Lassner and M. Zollhofer, "Pulsar: Efficient sphere-based neural rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1440–1449, 2021.

[113] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "EWA splatting," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 223–238, 2002.

[114] K. Teotia, H. Kim, P. Garrido, M. Habermann, M. Elgharib, and C. Theobalt, "GaussianHeads: End-to-End Learning of Drivable Gaussian Head Avatars from Coarse-to-fine Representations," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 6, pp. 1–12, 2024.

[115] S. Qian, T. Kirschstein, L. Schoneveld, D. Davoli, S. Giebenhain, and M. Nießner, "Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20299–20309, 2024.

[116] S. Giebenhain, T. Kirschstein, M. Rünz, L. Agapito, and M. Nießner, "Npga: Neural parametric gaussian avatars," in *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

[117] T. Kirschstein, J. Romero, A. Sevastopolsky, M. Nießner, and S. Saito, "Avat3r: Large Animatable Gaussian Reconstruction Model for High-fidelity 3D Head Avatars," *arXiv preprint arXiv:2502.20220*, 2025.

[118] A. Doi and A. Koide, "An Efficient Method of Triangulating Equi-Valued Surfaces by Using Tetrahedral Cells," *IEICE Transactions on Information and Systems*, vol. 74, pp. 214–224, 1991.

[119] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang, "4D Gaussian Splatting for Real-Time Dynamic Scene Rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20310–20320, June 2024.

[120] Y. Lin, Z. Dai, S. Zhu, and Y. Yao, "Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21136–21145, 2024.

[121] C. Stearns, A. Harley, M. Uy, F. Dubost, F. Tombari, G. Wetzstein, and L. Guibas, "Dynamic gaussian marbles for novel view synthesis of casual monocular videos," in *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

[122] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable Volumetric Rendering: Learning Implicit 3D Representations without

3D Supervision," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[123] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy Networks: Learning 3D Reconstruction in Function Space," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[124] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

[125] J. Hart, "Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces," *The Visual Computer*, vol. 12, 06 1995.

[126] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," *Proceedings of Machine Learning and Systems*, 2020.

[127] M. Oechsle, S. Peng, and A. Geiger, "UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction," in *International Conference on Computer Vision (ICCV)*, 2021.

[128] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[129] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains," *NeurIPS*, 2020.

[130] A. Chen, Z. Xu, X. Wei, S. Tang, H. Su, and A. Geiger, "Dictionary Fields: Learning a Neural Basis Decomposition," *ACM Trans. Graph.*, 2023.

[131] D. Verbin, P. Hedman, B. Mildenhall, T. Zickler, J. T. Barron, and P. P. Srinivasan, "Ref-NeRF: Structured view-dependent appearance for neural radiance fields," *CVPR*, 2022.

[132] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," in *CVPR*, 2021.

[133] T. Müller, "tiny-cuda-nn," 4 2021.

[134] T. Müller, F. Rousselle, J. Novák, and A. Keller, "Real-time Neural Radiance Caching for Path Tracing," *ACM Trans. Graph.*, vol. 40, pp. 36:1–36:16, Aug. 2021.

[135] K. Vaidyanathan, M. Salvi, B. Wronski, T. Akenine-Möller, P. Ebelin, and A. Lefohn, "Random-Access Neural Compression of Material Textures," in *Proceedings of SIGGRAPH*, 2023.

[136] T. Zeltner*, F. Rousselle*, A. Weidlich*, P. Clarberg*, J. Novák*, B. Bitterli*, A. Evans, T. Davidovič, S. Kallweit, and A. Lefohn, "Real-time neural appearance models," *ACM Transactions on Graphics*, vol. 43, no. 3, pp. 1–17, 2024.

[137] A. Karnewar, T. Ritschel, O. Wang, and N. Mitra, "ReLU Fields: The Little Non-Linearity That Could," in *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, (New York, NY, USA), Association for Computing Machinery, 2022.

[138] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.

[139] R. Szeliski, "Rapid octree construction from image sequences," *CVGIP: Image understanding*, vol. 58, no. 1, pp. 23–32, 1993.

[140] Y. Hu, R. Song, and Y. Li, "Efficient coarse-to-fine patchmatch for large displacement optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5704–5712, 2016.

[141] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015.

[142] Sara Fridovich-Keil and Giacomo Meanti, F. R. Warburg, B. Recht, and A. Kanazawa, "K-Planes: Explicit Radiance Fields in Space, Time, and Appearance," in *CVPR*, 2023.

[143] A. Cao and J. Johnson, "HexPlane: A Fast Representation for Dynamic Scenes," *CVPR*, 2023.

[144] Y. Zhuang, Q. Zhang, Y. Feng, H. Zhu, Y. Yao, X. Li, Y.-P. Cao, Y. Shan, and X. Cao, "Anti-aliased neural implicit surfaces with encoding level of detail," in *SIGGRAPH Asia 2023 Conference Papers*, pp. 1–10, 2023.

[145] B. Yi, W. Zeng, S. Buchanan, and Y. Ma, "Canonical Factors for Hybrid Neural Fields," in *International Conference on Computer Vision (ICCV)*, 2023.

[146] Q. Gao, Q. Xu, H. Su, U. Neumann, and Z. Xu, "Strivec: Sparse Tri-Vector Radiance Fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[147] F. Williams, J. Huang, J. Swartz, G. Klar, V. Thakkar, M. Cong, X. Ren, R. Li, C. Fuji-Tsang, S. Fidler, E. Sifakis, and K. Museth, "fVDB: A Deep-Learning Framework for Sparse, Large-Scale, and High-Performance Spatial Intelligence," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 43, July 2024.

[148] D. Kim, M. Lee, and K. Museth, "NeuralVDB: High-resolution Sparse Volume Representation using Hierarchical Neural Networks," *ACM Transactions on Graphics (ToG)*, vol. 43, feb 2024.

[149] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, Nov. 2013.

[150] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt, "Neural Sparse Voxel Fields," *NeurIPS*, 2020.

[151] S. Laine and T. Karras, "Efficient sparse voxel octrees," in *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, (New York, NY, USA), p. 55–63, Association for Computing Machinery, 2010.

[152] S. Lefebvre, S. Hornus, and F. Neyret, "Texture sprites: texture elements splatted on surfaces," in *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, (New York, NY, USA), p. 163–170, Association for Computing Machinery, 2005.

[153] K. Museth, "VDB: High-resolution sparse volumes with dynamic topology," *ACM Trans. Graph.*, vol. 32, July 2013.

[154] D. Thomas, B. Toussaint, J.-S. Franco, and E. Boyer, "VortSDF: 3D Modeling with Centroidal Voronoi Tesselation on Signed Distance Field," *arXiv preprint arXiv:2407.19837*, 2024.

[155] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pp. 501–518, Springer, 2016.

[156] R. Ramamoorthi and P. Hanrahan, "An efficient representation for irradiance environment maps," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 497–500, 2001.

[157] J. Kautz, J. Snyder, and P.-P. J. Sloan, "Fast arbitrary brdf shading for low-frequency lighting using spherical harmonics.," *Rendering Techniques*, vol. 2, no. 291-296, p. 1, 2002.

[158] J. T. Kajiya and B. P. Von Herzen, "Ray tracing volume densities," *ACM SIGGRAPH computer graphics*, vol. 18, no. 3, pp. 165–174, 1984.

[159] P.-P. Sloan, "Stupid Spherical Harmonics (SH) Tricks," *Game Developers Conference*, 01 2008.

[160] A. Gilbert, M. Volino, J. Collomosse, and A. Hilton, "Volumetric performance capture from minimal camera viewpoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 566–581, 2018.

[161] P. Zins, Y. Xu, E. Boyer, S. Wuhrer, and T. Tung, "Data-driven 3d reconstruction of dressed humans from sparse views," in *2021 International Conference on 3D Vision (3DV)*, pp. 494–504, IEEE, 2021.

[162] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, "Multi-view supervision for single-view reconstruction via differentiable ray consistency," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2626–2634, 2017.

[163] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2304–2314, 2019.

[164] A. E. Nocentino and P. J. Rhodes, "Optimizing memory access on GPUs using morton order indexing," in *Proceedings of the 48th annual ACM Southeast Conference*, pp. 1–4, 2010.

[165] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, "Large-scale data for multiple-view stereopsis," *International Journal of Computer Vision*, vol. 120, pp. 153–168, 2016.

[166] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan, "BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks," *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[167] A. Chen, Z. Xu, X. Wei, S. Tang, H. Su, and A. Geiger, "Dictionary fields: Learning a neural basis decomposition," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–12, 2023.

[168] Y. Kwon, D. Kim, D. Ceylan, and H. Fuchs, "Neural human performer: Learning generalizable radiance fields for human performance rendering," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24741–24752, 2021.

[169] A. A. Osman, T. Bolkart, D. Tzionas, and M. J. Black, "SUPR: A sparse unified part-based human body model," in *European conference on computer vision (ECCV)*, vol. 2, 2022.

[170] S. Saito, G. Schwartz, T. Simon, J. Li, and G. Nam, "Relightable gaussian codec avatars," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 130–141, 2024.

[171] Z. Zheng, X. Zhao, H. Zhang, B. Liu, and Y. Liu, "Avatarrex: Real-time expressive full-body avatars," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–19, 2023.

[172] G. Pons-Moll, S. Pujades, S. Hu, and M. J. Black, "ClothCap: Seamless 4D clothing capture and retargeting," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–15, 2017.

[173] A. Casado-Elvira, M. C. Trinidad, and D. Casas, "PERGAMO: Personalized 3d garments from monocular video," in *Computer Graphics Forum*, vol. 41, pp. 293–304, Wiley Online Library, 2022.

[174] Y. Li, M. Habermann, B. Thomaszewski, S. Coros, T. Beeler, and C. Theobalt, "Deep physics-aware inference of cloth deformation for monocular human performance capture," in *2021 International Conference on 3D Vision (3DV)*, pp. 373–384, IEEE, 2021.

[175] M. Habermann, L. Liu, W. Xu, G. Pons-Moll, M. Zollhoefer, and C. Theobalt, "HDhumans: A hybrid approach for high-fidelity digital humans," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 6, no. 3, pp. 1–23, 2023.

[176] S. Wang, K. Schwarz, A. Geiger, and S. Tang, "Arah: Animatable volume rendering of articulated human sdfs," in *European conference on computer vision*, pp. 1–19, Springer, 2022.

[177] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen, "On benchmarking camera calibration and multi-view stereo for high resolution imagery," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[178] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3260–3269, 2017.

[179] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "AMASS: Archive of motion capture as surface shapes," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5442–5451, 2019.

[180] Q. Fang, Q. Shuai, J. Dong, H. Bao, and X. Zhou, "Reconstructing 3d human pose by watching humans in the mirror," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12814–12823, 2021.

[181] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, "Dynamic FAUST: Registering human bodies in motion," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[182] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014.

[183] S. S. Jinka, A. Srivastava, C. Pokhariya, A. Sharma, and P. Narayanan, "Sharp: Shape-aware reconstruction of people in loose clothing," *International Journal of Computer Vision*, vol. 131, no. 4, pp. 918–937, 2023.

[184] M. Habermann, L. Liu, W. Xu, M. Zollhoefer, G. Pons-Moll, and C. Theobalt, "Real-time deep dynamic characters," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–16, 2021.

[185] Z. Zheng, H. Huang, T. Yu, H. Zhang, Y. Guo, and Y. Liu, "Structured local radiance fields for human avatar modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15893–15903, 2022.

[186] W. Wang, H.-I. Ho, C. Guo, B. Rong, A. Grigorev, J. Song, J. J. Zarate, and O. Hilliges, "4D-DRESS: A 4D Dataset of Real-world Human Clothing with Semantic Annotations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[187] Z. Xiong, C. Li, K. Liu, H. Liao, J. Hu, J. Zhu, S. Ning, L. Qiu, C. Wang, S. Wang, *et al.*, "Mvhumannet: A large-scale dataset of multi-view daily dressing human captures," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19801–19811, 2024.

[188] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 109–117, 2017.

[189] Y. Ge, W. Wang, Y. Chen, H. Chen, and C. Shen, "3d human reconstruction in the wild with synthetic data using generative models," *arXiv preprint arXiv:2403.11111*, 2024.

[190] Z. Yang, Z. Cai, H. Mei, S. Liu, Z. Chen, W. Xiao, Y. Wei, Z. Qing, C. Wei, B. Dai, W. Wu, C. Qian, D. Lin, Z. Liu, and L. Yang, "SynBody: Synthetic Dataset with Layered Human Models for 3D Human Perception and Modeling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20282–20292, October 2023.

[191] K. Museth, "Hierarchical digital differential analyzer for efficient raymarching in OpenVDB," in *ACM SIGGRAPH 2014 Talks*, pp. 1–1, 2014.

[192] R. Li, H. Gao, M. Tancik, and A. Kanazawa, "Nerfacc: Efficient sampling accelerates nerfs," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 18537–18546, 2023.

[193] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron, "NeRF in the Dark: High Dynamic Range View Synthesis from Noisy Raw Images," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16169–16178, 2021.

[194] P. Babin, P. Giguere, and F. Pomerleau, "Analysis of robust functions for registration algorithms," in *2019 International conference on robotics and automation (ICRA)*, pp. 1451–1457, IEEE, 2019.

[195] D. N. . M. Pettineo, "SIGGRAPH 2015 Course: Physically Based Shading in Theory and Practice - Advanced Lighting R&D at Ready At Dawn Studios." https://blog.selfshadow.com/publications/s2015-shading-course/. Accessed: 2024-11-07.

[196] Z. Majercik, A. Marrs, J. Spjut, and M. McGuire, "Scaling probe-based real-time dynamic global illumination for production," *arXiv preprint arXiv:2009.10796*, 2020.

[197] M. McGuire, M. Mara, D. Nowrouzezahrai, and D. Luebke, "Real-time global illumination using precomputed light field probes," in *Proceedings of the 21st ACM SIGGRAPH symposium on interactive 3D graphics and games*, pp. 1–11, 2017.

[198] Y.-C. Guo, D. Kang, L. Bao, Y. He, and S.-H. Zhang, "NeRFReN: Neural Radiance Fields With Reflections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18409–18418, June 2022.

[199] L. Wu, S. Bi, Z. Xu, F. Luan, K. Zhang, I. Georgiev, K. Sunkavalli, and R. Ramamoorthi, "Neural Directional Encoding for Efficient and Accurate View-Dependent Appearance Modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[200] B. Karis and E. Games, "Real shading in unreal engine 4," *Proc. Physically Based Shading Theory Practice*, vol. 4, no. 3, p. 1, 2013.

[201] Y. Jiang, J. Tu, Y. Liu, X. Gao, X. Long, W. Wang, and Y. Ma, "GaussianShader: 3D Gaussian Splatting with Shading Functions for Reflective Surfaces," *arXiv preprint arXiv:2311.17977*, 2023.

[202] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron, "NeRFactor: neural factorization of shape and reflectance under an unknown illumination," *ACM Transactions on Graphics (TOG)*, vol. 40, Dec. 2021.

[203] H. Jin, I. Liu, P. Xu, X. Zhang, S. Han, S. Bi, X. Zhou, Z. Xu, and H. Su, "TensoIR: Tensorial Inverse Rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[204] B. Miller, H. Chen, A. Lai, and I. Gkioulekas, "Objects as Volumes: A Stochastic Geometry View of Opaque Solids," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 87–97, June 2024.

[205] C. M. Schlick, "An Inexpensive BRDF Model for Physically-based Rendering," *Computer Graphics Forum*, vol. 13, 1994.

[206] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014.

[207] B. Nicolet, A. Jacobson, and W. Jakob, "Large Steps in Inverse Rendering of Geometry," *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, vol. 40, Dec. 2021.

[208] Z. Wang, T. Shen, M. Nimier-David, N. Sharp, J. Gao, A. Keller, S. Fidler, T. Müller, and Z. Gojcic, "Adaptive Shells for Efficient Neural Radiance Field Rendering," *ACM Trans. Graph.*, vol. 42, no. 6, 2023.