Fast Gradient Descent for Surface Capture Via Differentiable Rendering

Briac Toussaint, Maxime Genisson, Jean-Sébastien Franco

Fast Gradient Descent for Surface Capture Via Differentiable Rendering

Briac Toussaint Maxime Genisson Jean-Sébastien Franco

Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP (Institute of Engineering Univ. Grenoble Alpes), LJK, France
firstname.lastname@inria.fr

Abstract

Differential rendering has recently emerged as a powerful tool for image-based rendering or geometric reconstruction from multiple views, with very high quality. Up to now, such methods have been benchmarked on generic object databases and promisingly applied to some real data, but have yet to be applied to specific applications that may benefit. In this paper, we investigate how a differential rendering system can be crafted for raw multi-camera performance capture. We address several key issues in the way of practical usability and reproducibility, such as processing speed, explainability of the model, and general output model quality. This leads us to several contributions to the differential rendering framework. In particular we show that a unified view of differential rendering and classic optimization is possible, leading to a formulation and implementation where complete non-stochastic gradient steps can be analytically computed and the full per-frame data stored in video memory, yielding a straightforward and efficient implementation. We also use a sparse storage and coarse-to-fine scheme to achieve extremely high resolution with contained memory and computation time. We show that results rivaling or exceeding the quality of state of the art multi-view human surface capture methods are achievable in a fraction of the time, typically around a minute per frame.

1. Introduction

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 human motion and clothing, hair, self-occlusion, and other corrupting factors. 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 [9, 37], or even monocular input [31, 25]. To be useful, a surface reconstruction pipeline in this context must target several desirable properties. First, visual fidelity and quality of the produced mesh models are of course a priority. Second, efficiency of the algorithm and ease of use are typical requirements, since such data is often acquired in one or multiple sequences for reconstruction of motions, which for practical reasons calls for reasonably fast per-frame processing times.

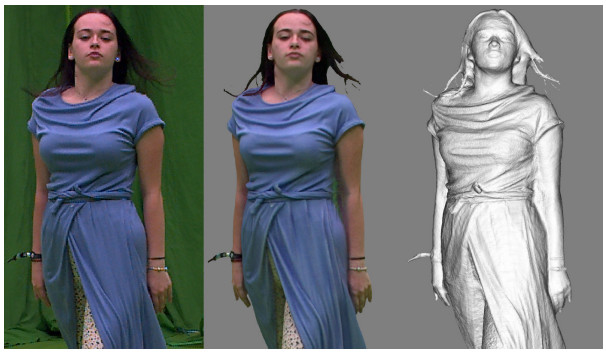

Figure 1. Typical results of our method: input image (left), colored mesh (center), geometry (right)

Figure 2. Close-up on another mesh: input image (left), colored mesh (center), geometry (right)

In recent years, differential rendering methods have gathered significant attention from the research community, as a new angle to address multi-view reconstruction and image-based rendering problems with high fidelity outputs. This makes them particularly palatable candidates to address the aforementioned quality requirement. Among those, volumetric differential rendering approaches such as the popular NeRF [18] have shown that scene estimation can be cast as a light field problem parameterized by a neural network, and have given rise to a number of more recent volumetric approaches. A rough surface can be extracted from the latter by thresholding the inferred opacity field, but better surface reconstructions can be extracted with dedicated differential rendering approaches centered on a surface parameterization [35, 23, 33]. While all of these methods produce amazing results, they typically require several hours per frame, and tens of hours for some, which rules out the approaches on the basis of tractability, for sequences that can be typically several hundred frames.

Contributions. We take inspiration in a stream of recent works showing that volumetric differential rendering can be significantly accelerated [26, 19], such that they achieve equivalent results to the original Nerf, in a matter of minutes. We show that a volumetric framework can be modified to output quality meshes thanks to the adjunction of appropriate losses. We also show that, with a simple non-neural parameterization analogous to [26] 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 straightforward and fully explainable implementation of differential rendering offering a new quality/computation time tradeoff, 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 [13].

2. Related Work

Multi-view capture. A very large corpus of work examines the problem of retrieving 3D models of human subjects from multiple cameras. We can roughly classify these methods along two lines. First, *model-based methods* use a pre-defined shape template or human deformation model and fit it to observations. Initially pioneered with subject-specific articulated templates using different image features such as stereo matches or silhouettes [5, 28, 32, 7], later methods have used human-generic parameterization based on shape spaces [2, 16] allowing them to estimate both pose and shape identity parameters simultaneously [4]. Later models may include clothing to complete the initially naked mesh [36], and have been increasingly enhanced with hybrid neural components to represent surface details on top of a human core parameterization [1]. In so doing the approaches target the ability to fit surfaces with more variability and detail, but they are still inherently constrained by the precision limits of the underlying represented shape space. For this reason lower-level *surface reconstruction approaches* are still metrologically relevant and used for detailed surface capture. Finding their roots in classic reconstruction approaches such as multi-view stereo [8, 30, 11, 27], they have been demonstrated to yield very high quality results in the multi-view capture case [20, 6, 24, 13]. Yet they are sometimes prone to artifacts and incompleteness issues, and are still generally slow, typically taking from several minutes to a few hours depending on the scene complexity, both issues we target improvements for in our proposal.

Differential rendering methods. Recent methods exploiting differentiable rendering are able to recover complete scenes with great accuracy, even in challenging conditions with reflective and translucent materials. They can be roughly split in two categories: surface or volume-based. *Surface methods* model the surface implicitly as an isosurface of a level set. In [21], the level set is sampled at regular intervals along each ray to find the first intersection with the surface, where the color is evaluated. [35, 14] replaced the level set with a signed distance function to find the isosurface more efficiently with sphere tracing. These methods are susceptible to local minima because the gradients are computed only at the current location of the surface. For this reason, silhouette masks are typically used for initialization and to constrain estimates inside the visual hull. In contrast, *volumetric methods* do not aim at recovering a surface at all. The scene is modelled by a radiance field and opacity volume, which contain color and opacity information at each point in space. An image is created by accumulating color and opacity along rays by marching in the volume. Lambertian materials can be approximated by a constant color, as shown by [15], but direction-dependent encodings are necessary to handle reflective surfaces. [18], later improved by [3], optimize a view-dependent radiance field for novel view synthesis, at which they excel. [23, 33, 34] can be classified as hybrid methods. They parameterize the density of a radiance field

by an underlying signed distance function, which can be rendered and optimized as a radiance field. They avoid local minima more easily than surface methods since the gradients are defined in a band of controllable width near the surface. Their main downside is that they need hours of optimization. The new volumetric methods [26, 29, 19] addressed that issue and obtain results comparable to [18] in a few minutes and even a few seconds in the case of [19]. However, they do not recover meshes of good quality since the the focus is on a radiance field scene encoding. We capitalize on the volumetric representation's flexibility and potential for efficiency, while focusing our method on the underlying surface thanks to appropriate regularizations.

3. 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), such that a surface is easily extractible from the resulting opacity volume.

3.1. 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 [29], a sparse grid [26], hash tables [19], or a fully connected neural network [18]. The color field has special parameterizations to achieve view-dependent effects: a neural network [18], neural features fed to a smaller neural net [19, 29] or spherical harmonics [26]. Independently from the actual parameterization, all volumetric differentiable rendering methods are based on the non-linear optimization of the following color similarity loss:

$$\mathcal{L}_{\text{photo}} = \sum_{r \in \text{rays}} ||\mathbf{C}(r) - \mathbf{C}_{\text{gt}}(r)|| \tag{1}$$

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:

$$\alpha_i \in [0,1] \quad \text{transparency of a sample} \tag{2}$$

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

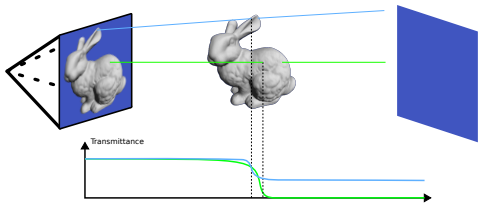

Figure 3. Volumetric integral example

We opt for a sparse grid representation, where color and transparency are obtained by interpolating from neighboring grid samples similarly to [26]. Since the samples are regularly spaced, they can be indexed by an integer k along the ray.

$$T_k = \prod_{j<k} \alpha_j \quad \text{partial transmittance} \tag{4}$$

$$\mathbf{C}_k = \sum_{j \leq k} (1-\alpha_j)\mathbf{c}_j T_j \quad \text{partial color} \tag{5}$$

$$\mathbf{C} = \sum_{k<\infty} (1-\alpha_k)\mathbf{c}_k T_k + T_\infty \mathbf{C}_{\text{bg}} \tag{6}$$

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 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 [26], we achieve view dependent effects, *e.g.* diffuse lighting and some specular lighting, by using spherical harmonics (SH):

$$\bar{\mathbf{c}}(\mathbf{v}) = \sum_{l=0}^{B} \sum_{m=-l}^{l} Y_{l,m}(\mathbf{v}) \boldsymbol{\beta}_{l,m} \tag{7}$$

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. Losses and regularization

Robust photometric loss. Let $||x||_{\mathcal{H}} = x^2$ if $x < \epsilon$ else $\epsilon(|x| - \frac{\epsilon}{2})$ the robust Huber norm. If x is a vector, we take the sum of the norms applied component-wise.

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

$$\mathcal{L}_{\text{photo}} = \sum_{r \in \text{rays}} ||\mathbf{C}(r) - \mathbf{C}_{\text{gt}}(r)||_{\mathcal{H}} \tag{8}$$

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

$$\mathcal{L}_{\text{smooth }\alpha} = \sum_{\text{voxels}} ||\nabla \bar{\alpha}||_2^2 \tag{9}$$

We also use a similar term on the voxels' color coefficients $\boldsymbol{\beta}_{l,m}$. Our coarse-to-fine scheme (4.2) 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 [29], we also use a per-sample regularization:

$$\mathcal{L}_{\text{sample}} = \sum_{\text{samples}} ||\mathbf{c}_k - \mathbf{C}||_{\mathcal{H}} \;\; T_k(1 - \alpha_k) \tag{10}$$

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:

$$\mathcal{L}_{\text{SH\_parsimony}} = \sum_{\text{voxels}} ||\boldsymbol{\beta}_{l,m}||_{\mathcal{H}} \tag{11}$$

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 spreaded 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:

$$\mathcal{L}_{\text{ballooning}} = \sum_{\text{voxels}} \bar{\alpha}^2 \tag{12}$$

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 [15]:

$$\mathbf{C}_{\text{gt}} = g\mathbf{C}_{\text{uncorrected gt}} + b \tag{13}$$

$$\mathbf{C}_{\text{bg}} = g\mathbf{C}_{\text{uncorrected bg}} + b \tag{14}$$

3.3. Gradient descent

A key insight of [26] is to show that the neural component in NeRF is less important to the result than the differential rendering logic. We take this notion a step further by also removing auto-differentiation for additional control and finetuning, since the losses admit simple analytical gradients. This elegantly leverages the benefit of previously explored differential rendering methods while casting inverse differential rendering as a classic optimization. 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 $\mathbf{C}$:

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \alpha_k} = \frac{\partial \mathcal{L}_{\text{photo}}}{\partial \mathbf{C}} \frac{\partial \mathbf{C}}{\partial \alpha_k} \tag{15}$$

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \mathbf{C}} = ||\mathbf{C} - \mathbf{C}_{\text{gt}}||'_{\mathcal{H}} \tag{16}$$

$$\frac{\partial \mathbf{C}}{\partial \alpha_k} = -\mathbf{c}_k T_k + \frac{\mathbf{C} - \mathbf{C}_k}{\alpha_k} \tag{17}$$

We march once per ray to build T_∞ and $\mathbf{C}$, then march a second time 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:

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \bar{\alpha}} = \sum_{r \in \text{rays}} \sum_{k \in \mathcal{S}(r)} \frac{\partial \mathcal{L}_{\text{photo}}}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial \bar{\alpha}} \tag{18}$$

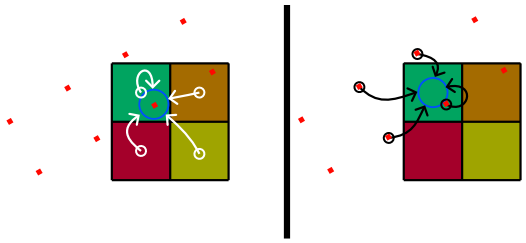

Figure 4. Gradient interpolation

$\mathcal{S}(r)$ is the set of samples on ray r to which the voxel of interest contributes. Instead of taking all these contributions into account explicitly, we read interpolated values of the per-sample gradients:

$$\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \bar{\alpha}} \approx \sum_{\text{cameras}} lerp(\frac{\partial \mathcal{L}_{\text{photo}}}{\partial \alpha}) \tag{19}$$

Figure 4 shows our double interpolation: on the left, each sample interpolates values from the 8 nearest voxels' centers to produce a gradient. On the right, each voxel reads gradients from the 8 nearest samples.

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:

$$\frac{\partial \mathcal{L}_{\text{smooth }\alpha}}{\partial \bar{\alpha}} = -\Delta\bar{\alpha} \tag{20}$$

We provide all gradient derivations as supplemental.

3.4. Surface extraction

We extract meshes from the voxel opacity grid using marching cubes [17] 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 very effective, provided the optimization converged to an SDF-like distribution of transparency values.

4. 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 [12] for a rough estimate of the location of the surface, and second, by precomputing 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.

4.1. 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 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 forming 2D tile bundles. We precompute 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 [22].

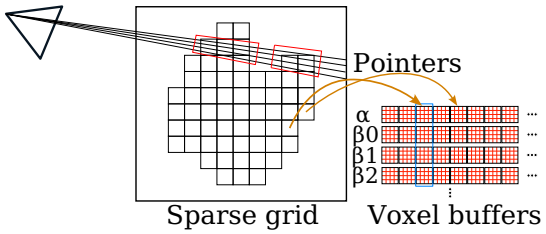

Figure 5. Sparse structure

4.2. 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.

4.3. Implementation details

We implement our optimization and visualization framework with C++ and OpenGL compute shaders.

For integration, the rays are bundled in groups of 8 for better cache coherency. We use an integration 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 supplemental). 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.

5. 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 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. More qualitative results are shown as supplementary content.

5.1. Validation with general-purpose benchmarks

Nerf synthetic dataset comparisons. We perform a comparison with [18] and [26] 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 1. We report the averaged PSNR values given in [26]. Our method performs encouragingly except for very strong reflective surfaces and concavities.

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 1. Quantitative results on NeRF synthetic dataset.

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 2. Quantitative evaluation on the DTU dataset.

Real data comparisons on DTU. We present results on the classic DTU dataset [10], 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 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 [35]. 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 2 shows some quantitative results for a scene subset with the DTU average chamfer-$L1$ distances metric. We compare against [35, 33] 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 supplemental). Our chamfer distance is consequently slightly below [35, 33] but our method achieves higher PSNR, and is 2-3 orders of magnitude faster.

5.2. 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. Points further than 5 centimeters are ignored similarly to the DTU evaluation protocol (green points in Figure 6). Our method performs on par or better than SOTA [13] 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 de-

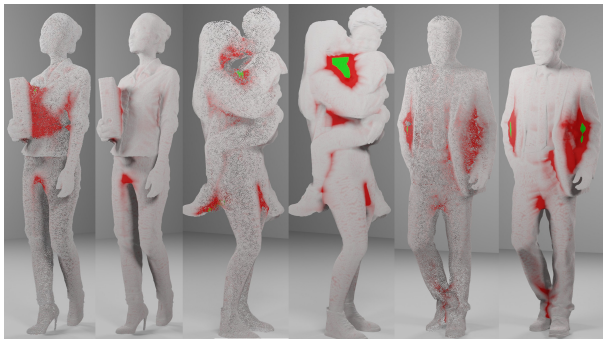

Figure 6. Chamfer heatmaps error comparison on synthetic human benchmark with 3 models, [13] vs ours.

tails (*e.g. rinkles 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. Chamfer evaluation (lower is better).

Qualitative comparison on real data. We compare with the state of the art surface capture method [13] in Figure 7, 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 [13]. The distinct geometry of fingers and veins are clearly visible in Figure 2, despite the noisy input. We want to emphasize that all of these results 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 thanks to our sparse coarse-to-fine scheme. We use 4 levels of detail in our coarse-to-fine strategy, with voxels of 1.25 mm at the finest resolution.

5.3. Transverse performance insights

Ablation of regularization terms. We ablate our regularizations and coarse-to-fine scheme in table 4 and Figure 8 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.

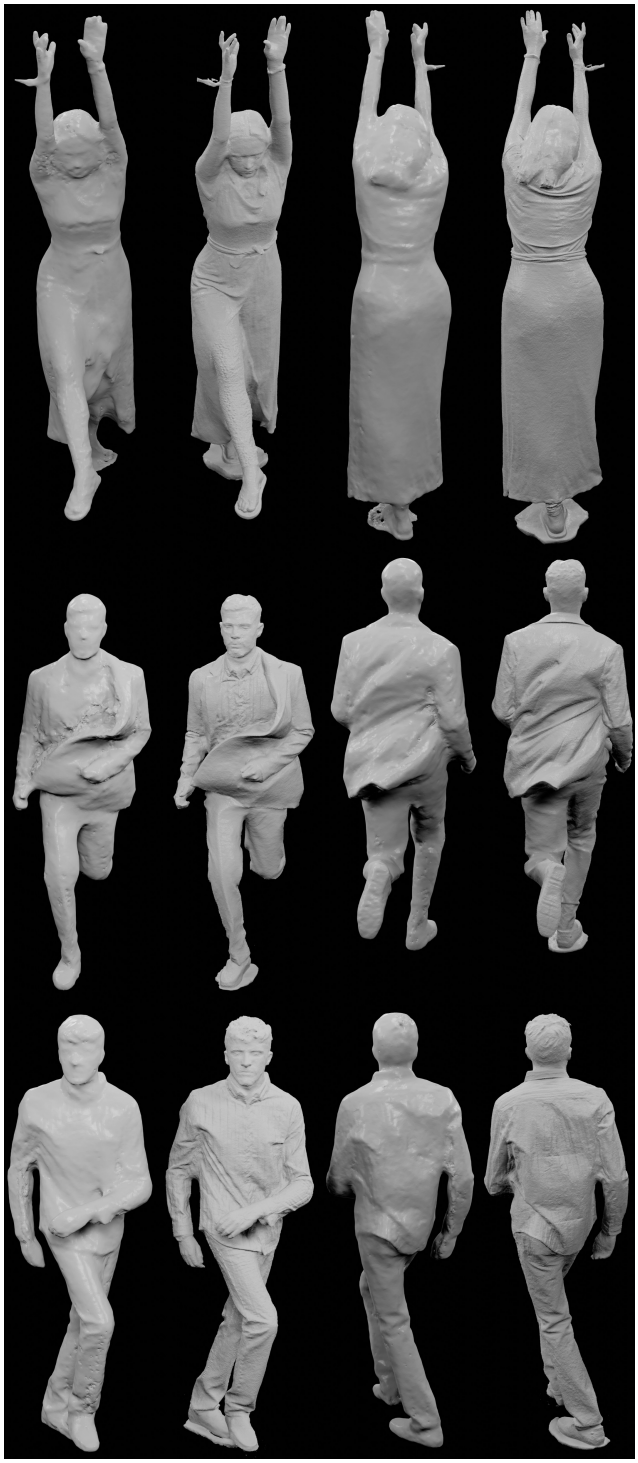

Figure 7. Ours (columns 2 & 4) vs [13] (columns 1 & 3)

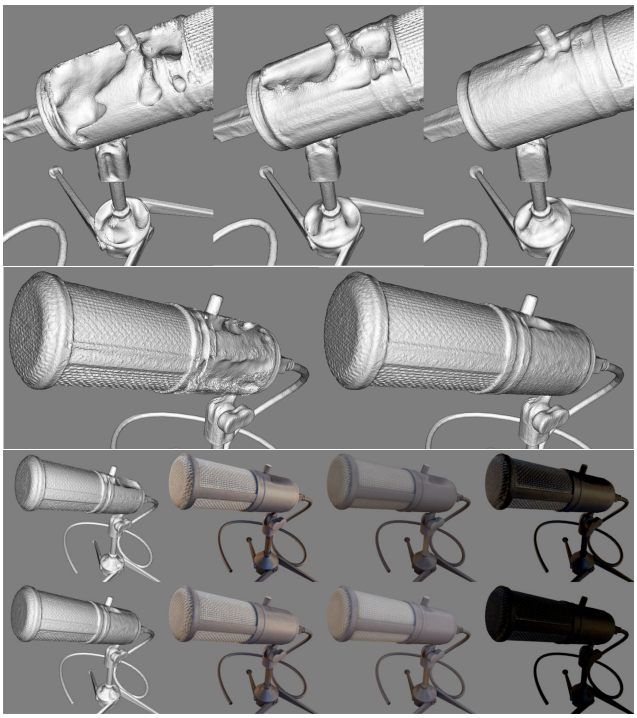

Figure 8. Top: $B = 0$ (right), $B = 1$ (center), $B = 2$ (left) Center: No ballooning term (right), with ballooning (left) Bottom: Without SH parsimony term (top row) vs with (bottom row). From left to right: geometry, mesh colors, $\beta_{0,0}$ only, specularity only

	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 4. Ablations of regularization terms on the mic model

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 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 5. PSNR on real data

Tradeoffs 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 9 illustrates. Our method then aggressively culls inner volumes deemed far away from the surface (in green in Figure 9, 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 5.1, or *e.g.* seen with over-merging under armpits of subjects in Figure 7. 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.

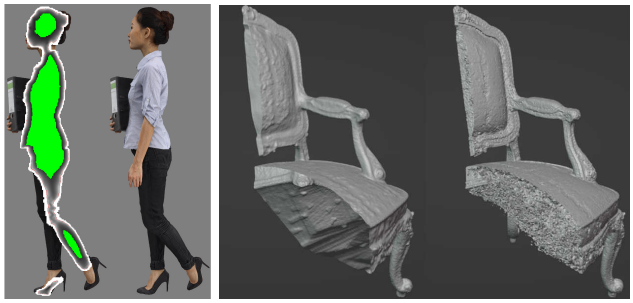

Figure 9. Sectional view of the opacity field (left), Ours (center), [19] (right)

6. Conclusion and future work

We have presented a simple and efficient differentiable rendering implementation, targeted at multi-view surface capture problems. Our coarse-to-fine strategy and custom gradient descent optimization allows fast convergence on a single GPU, typically in a minute. 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. Additionally, we show that computing analytical gradients is not particularly complex thanks to a simple parameterization, casting differential rendering as a classic optimization. The framework can be extended in many directions, with other appearance models to better handle specular reflections. In particular, the high pixel precision achieved in our capture experiments suggest that the method would also benefit from better sub-pixel denoising models and camera calibration refinement.

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

References

[1] Thiemo Alldieck, Hongyi Xu, and Cristian Sminchisescu. imghum: Implicit generative models of 3d human shape and articulated pose. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5441–5450. IEEE, 2021. 2

[2] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: shape completion and animation of people. *ACM Transactions on Graphics*, 24(3):408–416, 2005. 2

[3] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844, 2021. 2

[4] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[5] Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. In *ACM SIGGRAPH 2003 Papers*, pages 569–577, 2003. 2

[6] 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 Transactions on Graphics*, 34, 2015. 2

[7] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. *ACM Transactions on Graphics*, 27(3), 2008. 2

[8] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1362–1376, 2010. 2

[9] Andrew Gilbert, Marco Volino, John Collomosse, and Adrian Hilton. Volumetric performance capture from minimal camera viewpoints. In *European Conference on Computer Vision (ECCV)*, September 2018. 1

[10] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 6

[11] Rasmus Ramsbol Jensen, Anders Lindbjerg Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 406–413, 2014. 2

[12] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, Feb 1994. 5

[13] Vincent Leroy, Jean-Sébastien Franco, and Edmond Boyer. Volume Sweeping: Learning Photoconsistency for Multi-View Shape Reconstruction. *International Journal of Computer Vision*, 129:284–299, Feb. 2021. 2, 6, 7

[14] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2016–2025, 2020. 2

[15] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transaction on Graphics*, 38(4):65:1–65:14, July 2019. 2, 4

[16] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics*, 2015. 2

[17] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, Aug 1987. 5

[18] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 3, 6

[19] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15, July 2022. 2, 3, 8

[20] Armin Mustafa, Hansung Kim, Jean-Yves Guillemaut, and Adrian Hilton. General dynamic scene reconstruction from multiple view video. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[21] Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3501–3512, 2020. 2

[22] Anthony E. Nocentino and Philip J. Rhodes. Optimizing memory access on gpus using morton order indexing. In *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, New York, NY, USA, 2010. Association for Computing Machinery. 5

[23] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5569–5579, 2021. 2

[24] Nadia Robertini, Dan Casas, Edilson Aguiar, and Christian Theobalt. Multi-view performance capture of surface details. *International Journal of Computer Vision*, 124(1), 2017. 2

[25] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Hao Li, and Angjoo Kanazawa. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2304–2314, 2019. 1

[26] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 4, 6

[27] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[28] Jonathan Starck and Adrian Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3), May 2007. 2

[29] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3, 4

[30] Engin Tola, Vincent Lepetit, and Pascal Fua. DAISY: an efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, 2010. 2

[31] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1

[32] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popovic. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3), 2008. 2

[33] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems*, 2021. 2, 6

[34] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems*, 2021. 2

[35] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 6

[36] Chao Zhang, Sergi Pujades, Michael J. Black, and Gerard Pons-Moll. Detailed, accurate, human shape estimation from clothed 3d scan sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[37] Pierre Zins, Yuanlu Xu, Edmond Boyer, Stefanie Wuhrer, and Tony Tung. Data-Driven 3D Reconstruction of Dressed Humans From Sparse Views. In *3DV 2021 - International Conference on 3D Vision*, Dec. 2021. 1