# A General Differentiable Mesh Renderer for Image-based 3D Reasoning

Shichen Liu, Tianye Li, Weikai Chen*, and Hao Li

**Abstract**—Rendering bridges the gap between 2D vision and 3D scenes by simulating the physical process of image formation. By inverting such renderer, one can think of a learning approach to infer 3D information from 2D images. However, standard graphics renderers involve a fundamental step called rasterization, which prevents rendering to be differentiable. Unlike the state-of-the-art differentiable renderers [25], [35], which only approximate the rendering gradient in the backpropagation, we propose a natually differentiable rendering framework that is able to (1) directly render colorized mesh using differentiable functions and (2) back-propagate efficient supervisions to mesh vertices and their attributes from various forms of image representations. The key to our framework is a novel formulation that views rendering as an aggregation function that fuses the probabilistic contributions of all mesh triangles with respect to the rendered pixels. Such formulation enables our framework to flow gradients to the occluded and distant vertices, which cannot be achieved by the previous state-of-the-arts. We show that by using the proposed renderer, one can achieve significant improvement in 3D unsupervised single-view reconstruction both qualitatively and quantitatively. Experiments also demonstrate that our approach can handle the challenging tasks in image-based shape fitting, which remain nontrivial to existing differentiable renders.

**Index Terms**—Vision and Scene Understanding, Modeling and recovery of physical attributes, Perceptual reasoning; Computer Graphics, Picture/Image generation

## 1 Introduction

Understanding and reconstructing 3D scenes and structures from 2D images has been one of the fundamental goals in computer vision. The key to image-based 3D reasoning is to find sufficient supervisions flowing from the pixels to the 3D properties. To obtain image-to-3D correlations, prior approaches mainly rely on the matching losses based on 2D key points/contours [4], [32], [39], [48] or shape/appearance priors [2], [9], [29], [34], [64]. However, the above approaches are either limited to task-specific domains or can only provide weak supervision due to the sparsity of the 2D features. In contrast, as the process of producing 2D images from 3D assets, rendering relates each pixel with the 3D parameters by simulating the physical mechanism of image formulation. Hence, by inverting a renderer, one can obtain *dense* pixel-level supervision for *general-purpose* 3D reasoning tasks, which cannot be achieved by conventional approaches.

However, the rendering process is not differentiable in conventional graphics pipelines. In particular, a standard mesh renderer involves a non-differentiable sampling operation, called *rasterization*, which prevents the gradient to be backpropagated into the mesh vertices. To achieve differentiable rendering, recent advances [25], [35] only approximate the backward gradient with hand-crafted functions while directly employing a standard graphics renderer in the forward pass. While promising results have been shown for the task of image-based 3D reconstruction, they are not able to propagate gradient to distant vertices in all directions in the image space and fail to handle occlusions. We show in Section 5.3 that these limitations would cause problematic situations in image-based shape fitting where the 3D parameters cannot be efficiently optimized.

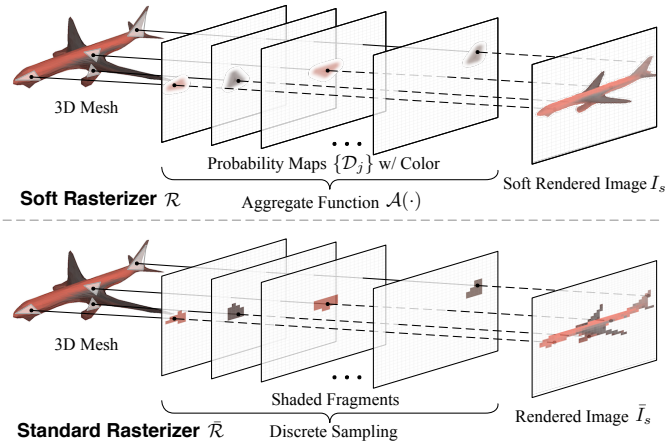



Fig. 1: We propose Soft Rasterizer $\mathcal{R}$ (upper), a natually differentiable renderer, which formulates rendering as a differentiable aggregating process $\mathcal{A}(\cdot)$ that fuses per-triangle contributions $\{\mathcal{D}_i\}$ in a "soft" probabilistic manner. Our approach attacks the core problem of differentiating the standard rasterizer, which cannot propagate gradients from pixels to geometry due to the discrete sampling operation (below).

In this paper, instead of studying a better form of rendering gradient, we attack the key problem of differentiating the forward rendering function. Specifically, we propose a *natually differentiable* rendering framework that is able to render a colorized mesh in the forward pass (Figure 1). In addition, our framework can

- *Shichen Liu and Tianye Li are with the Department of Computer Science, University of Southern California, and USC Institute for Creative Technologies, Los Angeles, CA. E-mail: {liushichen95, tianye.focus}@gmail.com*
- *Weikai Chen is with Tencent America, Los Angeles, CA. E-mail: chenwk891@gmail.com*
- *Hao Li is with Pinscreen, Los Angeles, CA. E-mail: hao@hao-li.com*

** indicates corresponding author*

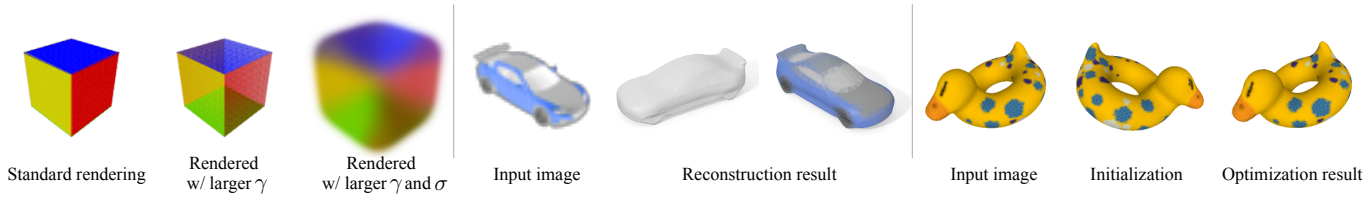



Fig. 2: Forward rendering (left): various rendering effects generated by SoftRas by tuning the degree of transparency and blurriness. Applications based on the backward gradients provided by SoftRas: (1) 3D unsupervised mesh reconstruction from a single input image (middle) and (2) 3D pose fitting to the target image by flowing gradient to the occluded triangles (right).

consider texture and a variety of 3D properties, including mesh geometry, vertex attributes (color, normal *etc.*), camera parameters and illuminations and is able to propagate efficient gradients from pixels to mesh vertices and their attributes. Our renderer can be plugged into either a neural network or a non-learning optimization framework for 3D reasoning.

The key to our approach is the formulation that views rendering as a "soft" probabilistic process. Unlike the popular z-buffer rendering approach, which only selects the color of the closest triangle in the viewing direction (Figure 1 below), we apply a differentiable, order-independent transparency rendering pipeline. In particular, we propose that all triangles have probabilistic contributions to each rendered pixel, which can be modeled as probability maps in the screen space. While conventional OpenGL rendering pipelines merge shaded fragments by selecting the closest triangle, we propose a differentiable aggregation function that fuses the per-triangle color maps based on the probability maps and the triangles' relative depths to obtain the final rendering result (Figure 1 upper). The proposed aggregating mechanism enables our renderer to propagate gradients to all mesh triangles, including the occluded ones. In addition, our framework can pass supervision signals from pixels to distant triangles in the image space because of its probabilistic formulation. We call our framework *Soft Rasterizer (SoftRas)* as it "softens" the discrete rasterization to enable differentiability.

SoftRas is able to provide high-quality gradients that supervise a variety of tasks on image-based 3D reasoning. To evaluate the performance of SoftRas, we show applications in 3D unsupervised single-view mesh reconstruction and image-based shape fitting (Figure 2, Section 5.2 and 5.3). In particular, as SoftRas provides strong error signals to the mesh generator simply based on the rendering loss, one can achieve mesh reconstruction from a single image without any 3D supervision. To faithfully texture the mesh, we further propose to extract representative colors from input image and formulates the color regression as a classification problem. Regarding the task of image-based shape fitting, we show that our approach is able to (1) handle occlusions using the aggregating mechanism that considers the probabilistic contributions of all triangles; and (2) provide much smoother energy landscape, compared to other differentiable renderers, that avoids local minima by using the smooth rendering (Figure 2 left). Experimental results demonstrate that our approach significantly outperforms the state-of-the-arts both quantitatively and qualitatively. The code of our paper is available at https://github.com/ShichenLiu/SoftRas.

The contributions of our paper can be summarized as follows:

- We propose a differentiable mesh rendering technique by introducing smoothing operations in both the spatial and depth extent to make rasterization differentiable.
- The proposed SoftRas renderer can directly render colorized mesh using differentiable functions with the ability of tuning the sharpness and blurriness of the rendering results. It enables gradients to be back-propagated from rendered image pixels to far-range and even occluded vertices, making challenging image-based 3D reasoning tasks (see Section 5.3) possible.
- With specific illumination models, e.g. Phong model or spherical harmonics, our framework is general enough to reason all 3D properties, e.g. geometry, camera, texture, material and lighting, by back-propagating supervision signals from the image. Our approach can also handle a variety of image representations including silhouette, shading and color images.

## 2 Related Work

**Differentiable Rendering.** To relate the changes in the observed image with that in the 3D shape manipulation, a number of existing techniques have utilized the derivatives of rendering [14], [15], [21], [37]. Recently, Loper and Black [35] introduced an approximate differentiable renderer which generates derivatives from projected pixels to the 3D parameters. Kato et al. [25] propose to approximate the backward gradient of rasterization with a hand-crafted function to achieve differentiable rendering for triangular mesh. Rhodin et al. [51] pioneers in leveraging transparency and spatial smooth for pose estimation in the context of point cloud rendering. By modeling an object as a collection of translucent Gaussian blobs whose color, transparency, center and magnitude can be optimized to fit a given silhouette and appearance, their approach scales well to a variety of representations, e.g. mesh, skeleton, etc., for the task of pose estimation. However, due to the loss of inter-connectivity between the Gaussians, it is non-trivial for their method to model the fine-scale deformations of mesh surface. Hence, such limitation hampers it from handling 3D mesh reconstruction and deformation tasks. Our work pushes the idea of transparency and spatial smoothing and extend it to mesh rendering. With the proposed SoftRas framework, we are able to handle general image-based 3D mesh reasoning tasks without the aforementioned limitation.

More recently, Li et al. [30] introduce a novel edge sampling algorithm that is able to compute derivatives of scalar functions over a ray-traced images. Built on Monte Carlo ray tracing, they propose new spatial acceleration and importance sampling solutions to efficiently sample derivatives for arbitrary boundces of light transport, including the secondary effects such as shadows or global illumination. While Li et al. [30] focus on images rendered using ray tracing technique, our work aims to differentiate the rasterization based rendering approach. Loubet et al. [36]

propose an alternative differentiable path tracer that achieves much lower variance at the cost of some bias. They avoid the explicit sampling of discontinuities by applying carefully chosen changes and re-parameterization of variables that remove the dependence of discontinuities of scene parameters. Nimier-David et al. [46] introduce Mitsuba 2, a versatile rendering framework that implements differentiable rendering along with other advanced rendering features, by leveraging modern C++ and template metaprogramming that enables retargetable implementation to a variety of application domains at compile time. To compute the derivatives of radiometric measures with respect to arbitrary scene parameterizations, Zhang et al. [63] introduce a new differential theory of radiative transfer that allows differentiation of radiative transfer equation while handling a large range of light transport phenomena. Recent advances in 3D face reconstruction [13], [52], [53], [54], [55], material inference [10], [33] and other 3D reconstruction tasks [18], [28], [44], [45], [50], [66] have also leveraged some other forms of differentiable rendering layers to obtain gradient flows in the neural networks. However, these rendering layers are usually designed for special purpose and thus cannot be generalized to other applications. In this paper, we focus on a general-purpose differentiable rendering framework that is able to directly render a given mesh using differentiable functions instead of only approximating the backward derivatives.

**Image-based 3D Reasoning.** 2D images are widely used as the media for reasoning about 3D properties. In particular, image-based reconstruction has received the most attention. Conventional approaches mainly leverage the stereo correspondence based on the multi-view geometry [12], [17] but are restricted to the coverage provided by the multiple views. With the availability of large-scale 3D shape dataset [7], learning-based approaches [16], [19], [58] are able to consider single or few images thanks to the shape prior learned from the data. To simplify the learning problem, recent works reconstruct 3D shape via predicting intermediate 2.5D representations, such as depth map [31], image collections [24], displacement map [20] or normal map [49], [59]. Pose estimation is another key task to understanding the visual environment. For 3D rigid pose estimation, while early approaches attempt to cast it as classification problem [56], recent approaches [26], [61] can directly regress the 6D pose by using deep neural networks. Estimating the pose of non-rigid objects, e.g. human face or body, is more challenging. By detecting the 2D key points, great progress has been made to estimate the 2D poses [5], [38], [60]. To obtain 3D pose, shape priors [2], [34] have been incorporated to minimize the shape fitting errors in recent approaches [3], [4], [5], [23]. Our proposed differentiable renderer can provide dense rendering supervision to 3D properties, benefitting a variety of image-based 3D reasoning tasks.

**Rendering transparency and smoothness.** Alpha blending [6] is widely used in computer graphics to create the appearance of partial or full transparency. Geometric elements are rendered in separated passes or layers and then composited into a single final image guided by the alpha channel. To avoid the expensive operations of alpha blending that require rendering geometry in sorted order, order-independent transparency (OIT) was later proposed to sort geometry per-pixel after rasterization and store all fragments for exact computing. While the spirit of OIT has inspired a number of subsequent works [11], [22], [40], [43] focusing on avoiding the cost of storing and sorting primitives or fragments, we focus on the branch of blended order independent transparency which leverages a similar idea with our work. Meshkin [42] was the first to introduce blended OIT by formulating a compositing operator based on weighted sum. Bavoil and Myers [1] improve Meshkin's operator using a better approximation of both coverage and color with a weighted average operator. Closer to our approach, McGuire and Bavoil [41] propose weighted blended OIT that leverages depth-based weights which decrease with the distance from the camera. Our aggregation function uses a similar idea and shows that the depth weights play a key role in achieving depth-based transparency and rendering differentiability. Smoothness-wise, conservative rasterization technique in graphics introduces an uncertainty region around the boundary of triangles to handle rounding errors and other issues that could add uncertainty to the exact dimensions of the triangle. In contrast, we extend this concept of uncertainty to handling the non-differentiability of rasterization. Our probability map can be viewed as a more general extension of uncertainty which allows the communication between distant pixels and triangles.

## 3 Soft Rasterizer

### 3.1 Differentiable Rendering Pipeline

As shown in Figure 3, we consider both extrinsic variables (camera $\mathbf{P}$ and lighting conditions $\mathbf{L}$) that define the environmental settings, and intrinsic properties (triangle meshes $\mathbf{M}$ and per-vertex appearance $\mathbf{A}$, including color, material *etc*.) that describe the model-specific properties. Following the standard rendering pipeline, one can obtain the mesh normal $\mathbf{N}$, image-space coordinate $\mathbf{U}$ and view-dependent depths $\mathbf{Z}$ by transforming input geometry $\mathbf{M}$ based on camera $\mathbf{P}$. With specific assumptions of illumination (e.g. spherical harmonics) and material models (e.g. Phong model), we can compute color $\mathbf{C}$ given $\{\mathbf{A}, \mathbf{N}, \mathbf{L}\}$. These two modules are *differentiable with automatic differentiation*. However, the subsequent operations: *rasterization* and *z-buffering*, in the standard graphics pipeline (Figure 3 red blocks) are not differentiable with respect to $\mathbf{U}$ and $\mathbf{Z}$ due to the discrete sampling operations.

Analytically speaking, following a similar spirit of [35], according to the computation graph in Figure 3, our gradient from rendered image $\mathbf{I}$ to vertices in mesh $\mathbf{M}$ is obtained by

$$\frac{\partial \mathbf{I}}{\partial \mathbf{M}} = \frac{\partial \mathbf{I}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{M}} + \frac{\partial \mathbf{I}}{\partial \mathbf{Z}} \frac{\partial \mathbf{Z}}{\partial \mathbf{M}} + \frac{\partial \mathbf{I}}{\partial \mathbf{N}} \frac{\partial \mathbf{N}}{\partial \mathbf{M}}. \tag{1}$$

While $\frac{\partial \mathbf{U}}{\partial \mathbf{M}}, \frac{\partial \mathbf{Z}}{\partial \mathbf{M}}, \frac{\partial \mathbf{I}}{\partial \mathbf{N}}$ and $\frac{\partial \mathbf{N}}{\partial \mathbf{M}}$ can be easily obtained by inverting the projection matrix and the illumination models, $\frac{\partial \mathbf{I}}{\partial \mathbf{U}}$ and $\frac{\partial \mathbf{I}}{\partial \mathbf{Z}}$ do not exist in conventional rendering pipelines. Our framework introduces an intermediate representation, probability map $\mathcal{D}$, that factorizes the gradient $\frac{\partial \mathbf{I}}{\partial \mathbf{U}}$ to $\frac{\partial \mathbf{I}}{\partial \mathcal{D}} \frac{\partial \mathcal{D}}{\partial \mathbf{U}}$, enabling the differentiability of $\frac{\partial \mathbf{I}}{\partial \mathbf{U}}$. Further, we obtain $\frac{\partial \mathbf{I}}{\partial \mathbf{Z}}$ via the proposed aggregation function. We will detail the gradient $\frac{\partial \mathcal{D}}{\partial \mathbf{U}}$ in Section 3.2 and gradient $\frac{\partial \mathbf{I}}{\partial \mathcal{D}}$ and $\frac{\partial \mathbf{I}}{\partial \mathbf{Z}}$ in Section 3.3, respectively.

**Our differentiable formulation.** We take a different perspective that the rasterization can be viewed as *binary masking* that is determined by the relative positions between the pixels and triangles, while z-buffering merges the rasterization results $\mathbf{F}$ in a pixel-wise *one-hot* manner (only selecting the closest triangle) based on the relative depths of triangles. The problem is then formulated as modeling the discrete binary masks and the one-hot merging operation in a soft and differentiable fashion. We therefore propose two

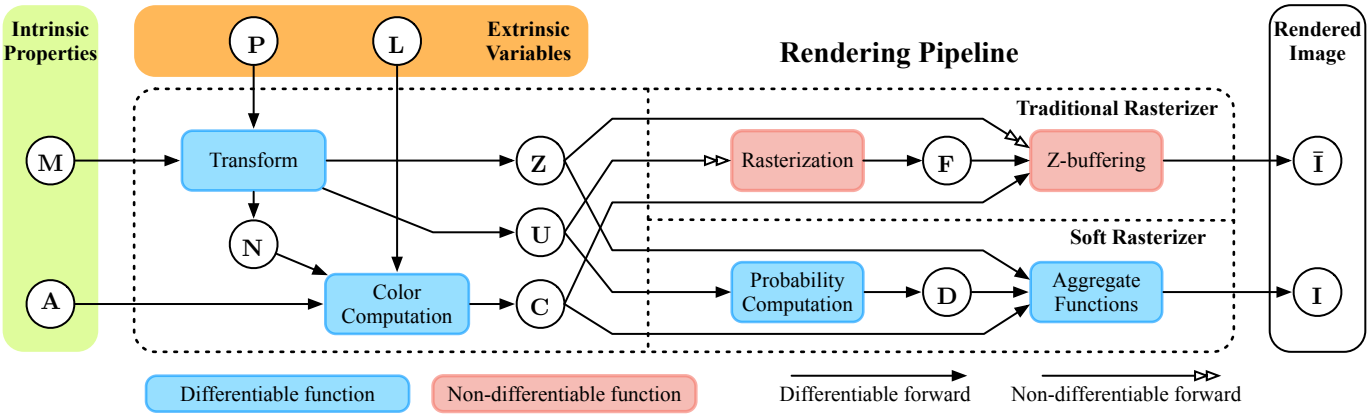
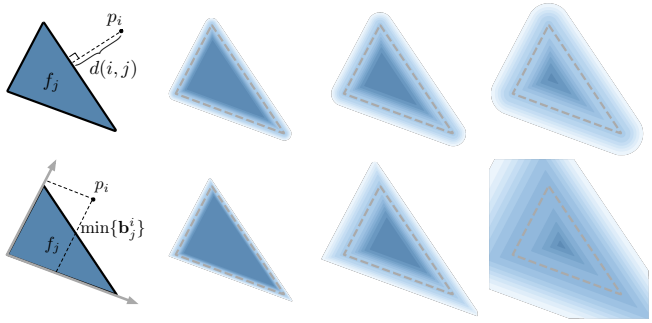



Fig. 3: Comparisons between the standard rendering pipeline (upper branch) and our rendering framework (lower branch).

major components that smooth the operations over the *spatial* and *depth* extent, respectively. Spatial-wise, we propose probability maps $\mathbf{D} = \{\mathcal{D}_j\}$ that model the probability of each pixel staying inside a specific triangle $f_j$. Depth-wise, aggregate function $\mathcal{A}(\cdot)$ is introduced to fuse per-triangle color maps based on $\{\mathcal{D}_j\}$ and the relative depths among triangles. With such formulation, all 3D properties, e.g. *camera*, *texture*, *material*, *lighting* and *geometry*, could receive gradients from the image.




(a) ground truth (b) $\sigma = 0.003$ (c) $\sigma = 0.01$ (d) $\sigma = 0.03$

Fig. 4: Probability maps of a triangle under Euclidean (upper) and barycentric (lower) metric. (a) definition of pixel-to-triangle distance; (b)-(d) probability maps generated with different $\sigma$.

### 3.2 Probability Map Computation

We model the influence of triangle $f_j$ on image plane by probability map $\mathcal{D}_j$. To estimate the probability of $\mathcal{D}_j$ at pixel $p_i$, the function is required to take into account both the relative position and the distance between $p_i$ and $\mathcal{D}_j$. To this end, we define $\mathcal{D}_j$ at pixel $p_i$ as follows:

$$\mathcal{D}_j^i = \text{sigmoid}(\delta_j^i \cdot \frac{d^2(i,j)}{\sigma}), \tag{2}$$

where $\sigma$ is a positive scalar that controls the sharpness of the probability distribution while $\delta_j^i$ is a sign indicator $\delta_j^i = \{+1, \text{if } p_i \in f_j; -1, \text{otherwise}\}$. We set $\sigma$ as $1 \times 10^{-4}$ unless otherwise specified. $d(i,j)$ is the closest distance from $p_i$ to $f_j$'s edges. A natural choice for $d(i,j)$ is the Euclidean distance. However, other metrics, such as barycentric or $l_1$ distance, can be used in our approach.

Intuitively, by using the *sigmoid* function, Equation 2 normalizes the output to $(0, 1)$, which is a faithful continuous approximation of binary mask with boundary landed on 0.5. In addition, the sign indicator maps pixels inside and outside $f_j$ to the range of $(0.5, 1)$ and $(0, 0.5)$ respectively. Figure 4 shows $\mathcal{D}_j$ of a triangle with varying $\sigma$ using Euclidean distance. Smaller $\sigma$ leads to sharper probability distribution while larger $\sigma$ tends to blur the outcome. This design allows controllable influence for triangles on image plane. As $\sigma \to 0$, the resulting probability map converges to the exact shape of the triangle, enabling our probability map computation to be a generalized form of traditional rasterization.

In the following context of this subsection, we will provide details of the computation of $\frac{\partial \mathcal{D}}{\partial \mathbf{U}}$ in terms of different metric choices. In particular, we introduce two candidate metrics for $d(i,j)$, namely *signed Euclidean distance* and *barycentric metric*. To correlate $p_i$ with $f_j$, we represent $p_i$ using *barycentric coordinate* $\mathbf{b}_j^i \in \mathbb{R}^3$ defined by $f_j$:

$$\mathbf{b}_j^i = \mathbf{U}_j^{-1}\mathbf{p}_i, \tag{3}$$

where $\mathbf{U}_j = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}_{f_j}$ and $\mathbf{p}_i = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{p_i}$.

#### 3.2.1 Euclidean Distance

Let $\mathbf{t}_j^i \in \mathbb{R}^3$ be the barycentric coordinate of the point on the edge of $f_j$ that is closest to $p_i$. The signed Euclidean distance $D_E(i,j)$ from $p_i$ to the edges of $f_j$ can be computed as:

$$D_E(i,j) = \delta_j^i \left\| \mathbf{U}_j \mathbf{t}_j^i - \mathbf{p}_i \right\|_2^2, \tag{4}$$

where $\delta_j^i$ is a sign indicator defined as $\delta_j^i = \{+1, \text{if } p_i \in f_j; -1, \text{otherwise}\}$.

Then the partial gradient $\frac{\partial D_E(i,j)}{\partial \mathbf{U}_j}$ can be obtained via:

$$\frac{\partial D_E(i,j)}{\partial \mathbf{U}_j} = 2\delta_j^i \left(\mathbf{U}_j \mathbf{t}_j^i - \mathbf{p}_i\right) \left(\mathbf{t}_j^i\right)^T. \tag{5}$$

#### 3.2.2 Barycentric Distance

We define the barycentric metric $D_B(i,j)$ as the minimum of barycentric coordinate:

$$D_B(i,j) = \min\{\mathbf{b}_j^i\} \tag{6}$$

let $s = \underset{k}{\operatorname{argmin}} \left(\mathbf{b}_j^i\right)^{(k)}$, then the gradient from $D_B(i,j)$ to $\mathbf{U}_j$ can be obtained through:

$$\frac{\partial D_B(i,j)}{\partial \left(\mathbf{U}_j\right)^{(k,l)}} = -\sum_t \left(\mathbf{p}_i\right)^{(t)} \left(\mathbf{U}_j^{-1}\right)^{(s,k)} \left(\mathbf{U}_j^{-1}\right)^{(l,t)} \tag{7}$$

where $k$ and $l$ are the indices of $\mathbf{U}_j$'s element.

We show the probability maps of a triangle constructed using Euclidean and barycentric metrics under different parametric settings in Figure 4. In general, Euclidean metric is able to generate uniformly decaying influence as the distance increases regardless of the triangle size and shape. Hence, it is more robust to varying densities of triangular meshes. In contrast, the probabilistic distribution generated by barycentric metric is more sensitive to the shape of triangle. However, compared to Euclidean metric which can only influence the closest edge to the query point, barycentric metric is able to pass gradient to all the three vertices of the triangle. We provide more detailed performance comparison of these two metrics in the ablation study in Section 5.2.4.

### 3.3 Aggregation Function

For each mesh triangle $f_j$, we define its color map $C_j$ at pixel $p_i$ on the image plane by interpolating color using barycentric coordinates. We clip its barycentric coordinates to [0, 1] and normalize their sum amounts to 1, which prevents negative barycentric coordinate for color computation. We then propose to use an aggregation function $\mathcal{A}(\cdot)$ to merge color maps $\{C_j\}$ to obtain rendering output $I$ based on $\{\mathcal{D}_j\}$ and the relative depths $\{z_j\}$. Inspired by the softmax operator, we define an aggregation function $\mathcal{A}_S$ as follows:

$$I^i = \mathcal{A}_S(\{C_j\}) = \sum_j w_j^i C_j^i + w_b^i C_b, \tag{8}$$

where $C_b$ is the background color; the weights $\{w_j\}$ satisfy $\sum_j w_j^i + w_b^i = 1$ and are defined as:

$$w_j^i = \frac{\mathcal{D}_j^i \exp(z_j^i/\gamma)}{\sum_k \mathcal{D}_k^i \exp(z_k^i/\gamma) + \exp(\epsilon/\gamma)}. \tag{9}$$

In particular, $z_j^i$ denotes the normalized depth of the 3D point on $f_i$ whose 2D projection is $p_i$. We normalize the depth so that the closer triangle receives a larger $z_j^i$ by

$$z_j^i = \frac{Z_{far} - Z_j^i}{Z_{far} - Z_{near}}, \tag{10}$$

where $Z_j^i$ denotes the actual clipped depth of $f_j$ at $p_i$, while $Z_{near}$ and $Z_{far}$ denote the far and near cut-off distances of the viewing frustum respectively. $\epsilon$ is small constant that enables the background color while $\gamma$ (set as $1 \times 10^{-4}$ unless otherwise specified) controls the sharpness of the aggregation function. Note that $w_j$ is a function of two major variables: $\mathcal{D}_j$ and $z_j$. Specifically, $w_j$ assigns higher weight to closer triangles that have larger $z_j$. As $\gamma \to 0$, the color aggregation function only outputs the color of nearest triangle, which exactly matches the behavior of z-buffering. In addition, $w_j$ is robust to z-axis translations. $\mathcal{D}_j$ modulates the $w_j$ along the $x$, $y$ directions such that the triangles closer to $p_i$ on screen space will receive higher weight. Equation 8 also works for *shading images* when the intrinsic vertex colors are set as constant ones.

The gradient $\frac{\partial I}{\partial \mathcal{D}_j^i}$ and $\frac{\partial I}{\partial z_j^i}$ can be obtained as follows:

$$\frac{\partial I^i}{\partial \mathcal{D}_j^i} = \frac{w_j^i}{\mathcal{D}_j^i}(C_j^i - I^i) \tag{11}$$

$$\frac{\partial I^i}{\partial z_j^i} = \frac{w_j^i}{\gamma}(C_j^i - I^i) \tag{12}$$

Note that we only clamp the barycentric coordinates for color computation ($C$ in Equation 8, 11 and 12). The gradients with respect to spatial location and per-vertex normal still exist, i.e. through $w$ to $\mathcal{D}$, then to mesh vertices in Equation 11 and 12, where the barycentric weights are not clamped. In addition, our framework does not provide gradient for UV coordinates. However, we argue that this gradient may not be necessary as with given fixed mesh topology and UV mapping, we are able to optimize the color of UV texture map. Yet, directly deforming mesh in 3D spatial space has greater capability than deforming UV coordinates in 2D UV space. Further, since the texture itself might be changing during the learning process, forcing UV coordinates to be fixed can achieve more stable training of networks.

We explain details of gradient computation and texturing in the *supplemental materials*.

**Occupancy Aggregation Function.** While Equation 8 works for colors, it can also be extend to aggregate the alpha channel, i.e. the silhouette images. However, the continuous interpolation in Equation 8 may not fit the binary nature of silhouettes. In addition, the silhouette of object is independent from its color and depth map. Hence, we propose a dedicated aggregation function $\mathcal{A}_O$ for silhouettes based on the binary occupancy:

$$I_s^i = \mathcal{A}_O(\{\mathcal{D}_j\}) = 1 - \prod_j (1 - \mathcal{D}_j^i). \tag{13}$$

Intuitively, Equation 13 models silhouette as the probability of having *at least* one triangle cover the pixel $p_i$. The partial gradient $\frac{\partial I_{sil}^i}{\partial \mathcal{D}_j^i}$ can be computed as follows:

$$\frac{\partial I_{sil}^i}{\partial \mathcal{D}_j^i} = \frac{1 - I_{sil}^i}{1 - \mathcal{D}_j^i}. \tag{14}$$

Note that there might exist other forms of occupancy aggregate functions. One alternative option may be using a universal aggregate function $\mathcal{A}_N$ that is implemented as a neural network. We provide an ablation study on this regard in Section 5.2.4.

### 3.4 Comparisons with Prior Works

In this section, we compare our approach with the state-of-the-art rasterization-based differential renderers: OpenDR [35] and NMR [25], in terms of gradient flows as shown in Figure 5. In particular, NMR leverages a hand-crafted function to approximate the gradient for rasterization while directly using standard graphics renderer in the forward rendering. OpenDR approximates rendering using a function with respect to vertex locations, camera parameters and per-vertex brightness. When computing gradient from image intensity to 2D image coordinates, OpenDR applies image filtering with limited width to smooth image and pass gradient to boundary and interior pixels. Due to the filtering operation, pixels close to or inside the triangle can receive gradients. But the range is limited to the bandwidth of the filter, as demonstrated in Figure 5.

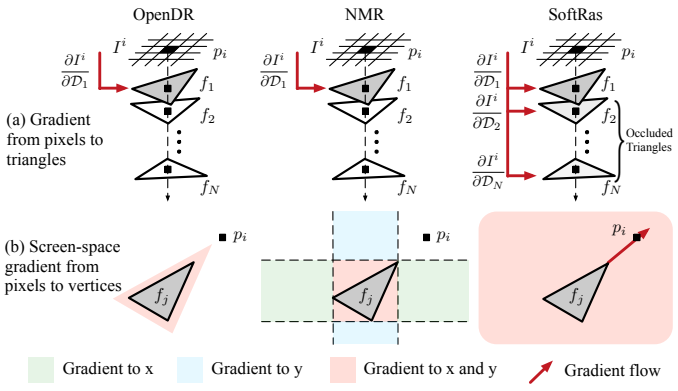



Fig. 5: Comparisons with prior differentiable renderers in terms of gradient flow.

**Gradient from pixels to triangles.** Since both OpenDR and NMR utilize standard graphics renderer in the forward pass, they have no control over the intermediate rendering process and thus cannot flow gradient into the triangles that are occluded in the final rendered image (Figure 5(a) left and middle). In addition, as their gradients only operate on the image plane, both OpenDR and NMR are not able to optimize the depth value $z$ of the triangles. In contrast, our approach has full control on the internal variables and is able to flow gradients to invisible triangles and the $z$ coordinates of all triangles through the aggregation function (Figure 5(a) right).

**Screen-space gradient from pixels to vertices.** Thanks to our continuous probabilistic formulation, in our approach, the gradient from pixel $p_j$ in screen space can flow gradient to all distant vertices (Figure 5(b) right). However, for OpenDR, a vertex can only receive gradients from neighboring pixels within a close distance due to the local filtering operation (Figure 5(b) left). Regarding NMR, there is no gradient defined from the pixels inside the white regions with respect to the triangle vertices ((Figure 5(b) middle). In contrast, our approach does not have such issue thanks to our orientation-invariant formulation.

**Comparison of time and space complexity.** In our implementation, we do not use PyTorch autograd functions considering its large memory footage and computation overhead. Instead, we argue that it is necessary to customize the CUDA kernel functions, and propose two strategies for optimization: 1) we clip the computation for pixels that are too far away from the triangle (negligible influence), hence the footprint of each triangle is not infinite; 2) we customize our aggregation function implementation so that the memory consumption for each pixel is $O(1)$. Our implemented complexity is the same as that of prior works (NMR and OpenDR), which are $O(HWN)$ and $O(HW)$ respectively. Theoretically, it is possible to further reduce the time complexity by introducing more advanced hierarchical data structure. However, due to the restriction on data management and workflow streaming, it is very challenging to implement techniques like hierarchical z-buffer in current deep learning frameworks, making it difficult to reach the lower bound of theoretical complexity.

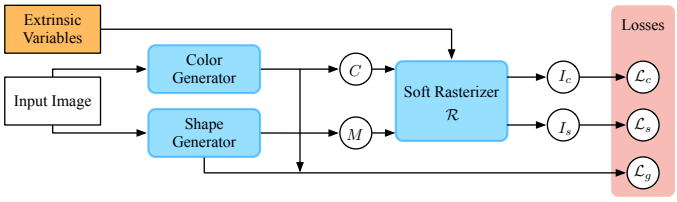



Fig. 6: The proposed framework for single-view mesh reconstruction.

## 4 Image-based 3D Reasoning

Our SoftRas can compute gradients to both extrinsic (e.g. camera and lighting) and intrinsic (e.g. geometry, texture, material, etc.) properties, enabling a variety of tasks on 3D reasoning. In Section 5.2, we evaluate SoftRas for single-view mesh reconstruction by fixing extrinsic parameters.

### 4.1 Single-view Mesh Reconstruction

Image-based 3D reconstruction plays a key role in a variety of tasks in computer vision and computer graphics, such as scene understanding, VR/AR, autonomous driving, etc. Reconstructing 3D objects either in mesh [47], [58] or voxel [62] representation from a single RGB image has been actively studied thanks to the advent of deep learning technologies. While most approaches on mesh reconstruction rely on supervised learning, methods working on voxel representation have strived to leverage rendering loss [8], [28], [57] to mitigate the lack of 3D data. However, the reconstruction quality of voxel-based approaches are limited primarily due to the high computational expense and its discrete nature. Nonetheless, unlike voxels, which can be easily rendered via differentiable projection, rendering a mesh in a differentiable fashion is non-trivial as discussed in the previous context. By introducing a naturally differentiable mesh renderer, SoftRas combines the merits of both worlds – the ability to harness abundant resources of multi-view images and the high reconstruction quality of mesh representation.

To demonstrate the effectiveness of soft rasterizer, we fix the extrinsic variables and evaluate its performance on single-view 3D reconstruction by incorporating it with a mesh generator. The direct gradient from image pixels to shape and color generators enables us to achieve *3D unsupervised* mesh reconstruction. Our framework is demonstrated in Figure 6. Given an input image, our shape and color generators generate a triangle mesh $M$ and its corresponding colors $C$, which are then fed into the soft rasterizer. The SoftRas layer renders both the silhouette $I_s$ and color image $I_c$ and provide rendering-based error signal by comparing with the ground truths. Inspired by the latest advances in mesh learning [25], [58], we leverage a similar idea of synthesizing 3D model by deforming a template mesh. To validate the performance of soft rasterizer, the shape generator employ an encoder-decoder architecture identical to that of [25], [62]. The details of the shape and generators are described in the *supplemental materials*.

**Losses.** The reconstruction networks are supervised by three losses: silhouette loss $\mathcal{L}_s$, color loss $\mathcal{L}_c$ and geometry loss $\mathcal{L}_g$. Let $\hat{I}_s$ and $I_s$ denote the predicted and the ground-truth silhouette respectively. The silhouette loss is defined as $\mathcal{L}_s = 1 - \frac{||\hat{I}_s \otimes I_s||_1}{||\hat{I}_s \oplus I_s - \hat{I}_s \otimes I_s||_1}$, where $\otimes$ and $\oplus$ are the element-wise product and sum operators respectively. The color loss is measured as the $l_1$ norm between the rendered and input image: $\mathcal{L}_c = ||\hat{I}_c - I_c||_1$.

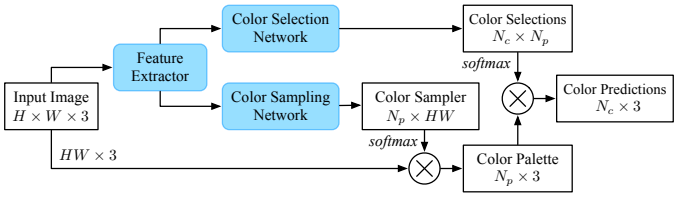



Fig. 7: Network structure for color reconstruction.

To achieve appealing visual quality, we further impose a geometry loss $\mathcal{L}_g$ that regularizes the Laplacian of both shape and color predictions. The final loss is a weighted sum of the three losses:

$$\mathcal{L} = \mathcal{L}_s + \lambda \mathcal{L}_c + \mu \mathcal{L}_g. \tag{15}$$

#### 4.1.1 Color Reconstruction

The recent advances in novel view synthesis [65] has demonstrated that instead of learning to synthesize pixels from scratch, learning to *copy* from the input image can achieve results with even higher fidelity. Though directly regressing colors is conceptually simpler, training a reliable regression model is inherently challenging and prone to over-fitting due to the difficulty of predicting a continuous variable lying in a high-dimensional space. Hence, we propose to formulate color reconstruction as a classification problem that learns to reuse the pixel colors in the input image for each sampling point. Let $N_c$ denote the number of sampling points on $M$ and $H, W$ be the height and width of the input image respectively. However, the computational cost of a naive color selection approach is prohibitive, i.e. $O(HWN_c)$. To address this challenge, we propose to colorize mesh using a color palette, as shown in Figure 7. Specifically, after passing input image to a neural network, the extracted features are fed into (1) a sampling network that samples the representative colors for building the palette; and (2) a selection network that combines colors from the palette for texturing the sampling points. The color prediction is obtained by multiplying the color selections with the learned color palette. Our approach reduces the computation complexity to $O(N_p(HW + N_c))$, where $N_p$ is the size of color palette. With a proper setting of $N_p$, one can significantly reduce the computational cost while achieving sharp and accurate color recovery.

### 4.2 Image-based Shape Fitting

Image-based shape fitting has a fundamental impact in various tasks, such as pose estimation, shape alignment, model-based reconstruction, *etc*. Yet without direct correlation between image and 3D parameters, conventional approaches have to rely on coarse correspondences, e.g. 2D joints [4] or feature points [48], to obtain supervision signals for optimization. In contrast, SoftRas can directly back-propagate pixel-level errors to 3D properties, enabling dense image-to-3D correspondence for high-quality shape fitting. However, a differentiable renderer has to resolve two challenges in order to be readily applicable. (1) *occlusion awareness*: the occluded portion of 3D model should be able to receive gradients in order to handle large pose changes. (2) *far-range impact*: the loss at a pixel should have influence on distant mesh vertices, which is critical to dealing with local minima during optimization. While prior differentiable renderers [25], [35] fail to satisfy these two criteria, our approach handles these challenges simultaneously. (1) Our aggregate function fuses the probability maps from all triangles, enabling the gradients to be flowed to all vertices including the occluded ones. (2) Our soft approximation based on probability distribution allows the gradient to be propagated to the far end while the size of receptive field can be well controlled (Figure 4). To this end, our approach can faithfully solve the image-based shape fitting problem by minimizing the following energy objective:

$$\underset{\rho,\theta,t}{\operatorname{argmin}} ||R(M(\rho,\theta,t)) - I_t||_2, \tag{16}$$

where $R(\cdot)$ is the rendering function that generates a rendered image $I$ from mesh $M$, which is parametrized by its pose $\theta$, translation $t$ and non-rigid deformation parameters $\rho$. The difference between $I$ and the target image $I_t$ provides strong supervision to solve the unknowns $\{\rho, \theta, t\}$.

## 5 Experiments

### 5.1 Forward Rendering Results

Our proposed SoftRas can directly render a given mesh using differentiable functions, while previous rasterization-based differentiable renderers [25], [35] have to rely the off-the-shelf renders for forward rendering. In addition, compared to standard graphics renderer, SoftRas can achieve different rendering effects in a continuous manner thanks to its probabilistic formulation.

By increasing $\sigma$, the key parameter that controls the sharpness of the screen-space probability distribution, we are able to generate more blurry rendering results. Furthermore, with increased $\gamma$, one can assign more weights to the triangles on the far end, naturally achieving more transparency in the rendered image. We demonstrate rendering effects in the *supplemental materials*. We will show in Section 5.3 that the blurring and transparent effects are the key to reshaping the energy landscape in order to avoid local minima.

### 5.2 Single-view Mesh Reconstruction

#### 5.2.1 Experimental Setup

**Datasets and Evaluation Metrics.** We use the dataset provided by [25], which contains 13 categories of objects from ShapeNet [7]. Each object is rendered in 24 different views with image resolution of 64 $\times$ 64. For fair comparison, we employ the same train/validate/test split on the same dataset as in [25], [62]. For quantitative evaluation, we adopt the standard reconstruction metric, 3D intersection over union (IoU), to compare with baseline methods. Specifically, we voxelize our mesh to 32 $\times$ 32 $\times$ 32 and compare it with the ground truth.

**Implementation Details.** We use the same structure as [25], [62] for mesh generation. Our network is optimized using Adam [27] with $\alpha = 1 \times 10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The training of our model takes 12 hours per category on a single NVIDIA 1080Ti GPU. Specifically, we set $\lambda = 1$ and $\mu = 1 \times 10^{-3}$ across all experiments unless otherwise specified. We train the network with multi-view images of batch size 64 and implement it using PyTorch. The template mesh used in single-view reconstruction has 642 vertices and 1280 triangles. We use Phong model with 0.5 ambient rate rate and 0.5 diffusive rate. We use intrinsic colors for materials where ambient reflection rate is 1.0 and diffuse reflection rate is 1.0. Note that we do not use the AutoGrad function provided by PyTorch. The practical problem of PyTorch AutoGrad function is that it explicitly stores all the intermediate variables in CUDA

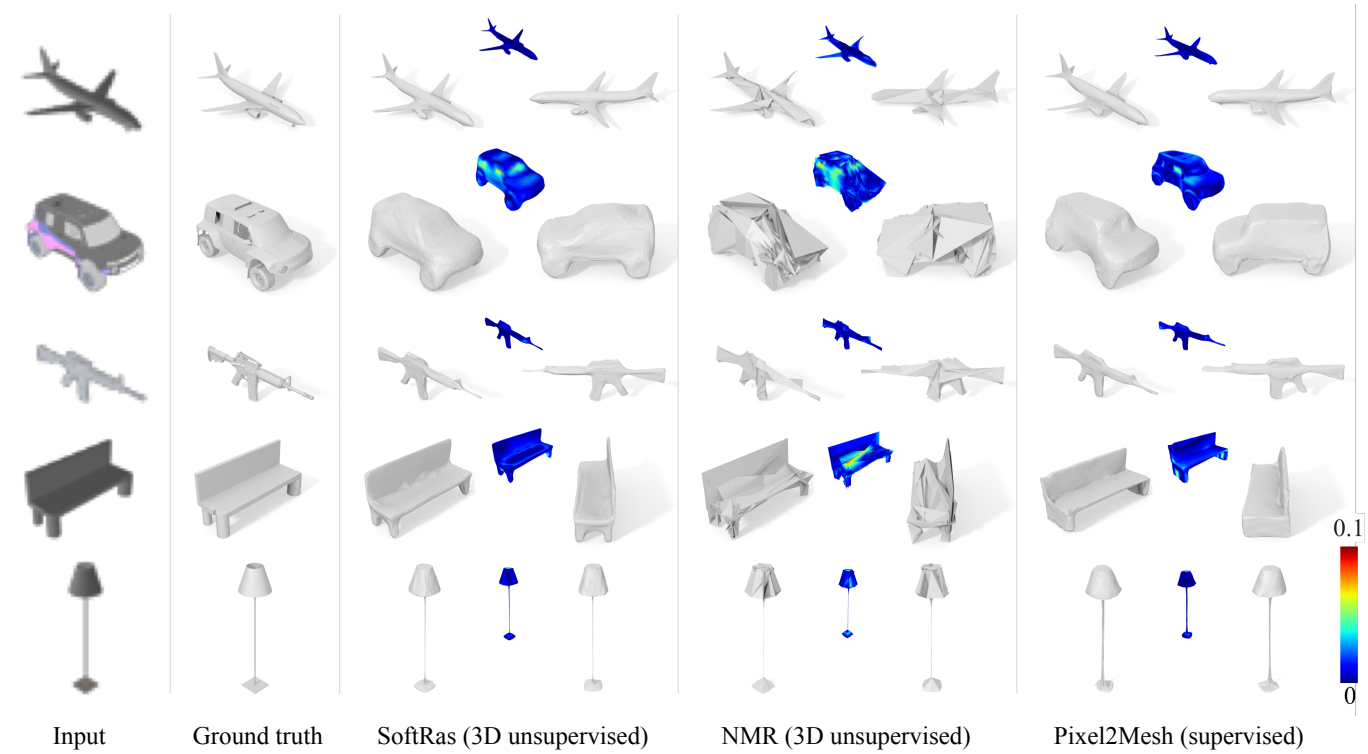



Fig. 8: 3D mesh reconstruction from a single image. From left to right, we show input image, ground truth, the results of our method (SoftRas), Neural Mesh Renderer [25] and Pixel2mesh [58] – all visualized from 2 different views. Along with the results, we also visualize mesh-to-scan distances measured from reconstructed mesh to ground truth.

the memory. This leads to a prohibitive complexity of $O(HWN)$, where $H$, $W$ are the image height and width, respectively; $N$ is number of triangles. Instead, we implement customized CUDA kernel functions that reduce the memory complexity to $O(HW)$ by aggregating all triangles in a single pixel on-the-fly. The gradient is crucial to implement these kernels. Therefore, we provide the details of all the gradients in Section 3.

#### 5.2.2 Qualitative Results

**Comparisons with the state-of-the-arts.** We compare the qualitative results of our approach with that of the state-of-the-art supervised [58] and 3D unsupervised [25] mesh reconstruction approaches in Figure 8. Though NMR [25] can recover the rough shape, the mesh surface is discontinuous and suffers from a considerable amount of self intersections. In contrast, our method can faithfully reconstruct fine details of the object, such as the airplane tail and the rifle barrel, while ensuring smoothness of the surface. Though trained without 3D supervision, our approach achieves results on par with the supervised method Pixel2Mesh [58]. In some cases, our approach can generate even more appealing details than that of [58], e.g. the bench legs, the airplane engine and the side of the car. Mesh-to-scan distance visualization also shows our results achieve much higher accuracy than [25] and comparable accuracy with that of [58].

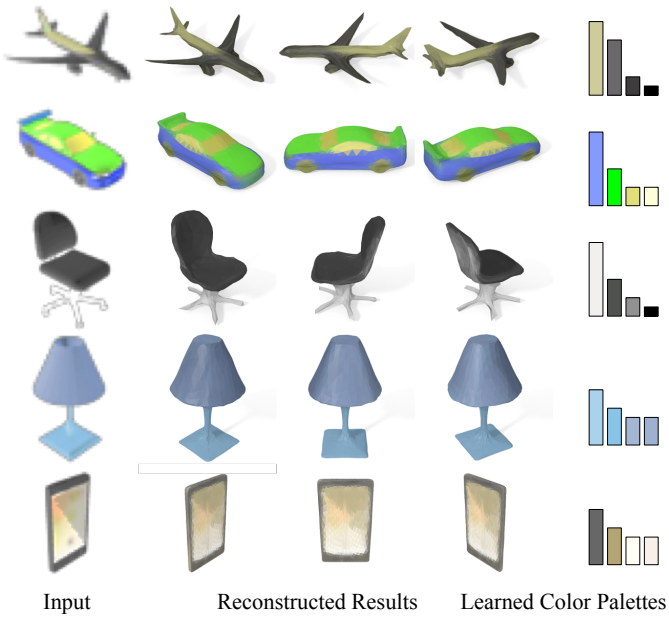



Fig. 9: Results of colorized mesh reconstruction. The learned principal colors and their usage histogram are visualize on the right.

**Color Reconstruction.** Our method is able to faithfully recover the mesh color based on the input image. Figure 9 presents the colorized reconstruction from a single image and the learned color palettes. Though the resolution of the input image is rather low ($64 \times 64$), our approach is still able to achieve sharp color recovery and accurately restore the fine details, e.g. the subtle color transition on the body of airplane and the shadow on the phone screen.

**Single-view Reconstruction from Real Images.** We further

| Category | Airplane | Bench | Dresser | Car | Chair | Display | Lamp | Speaker | Rifle | Sofa | Table | Phone | Vessel | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| retrieval [62] | 0.5564 | 0.4875 | 0.5713 | 0.6519 | 0.3512 | 0.3958 | 0.2905 | 0.4600 | 0.5133 | 0.5314 | 0.3097 | 0.6696 | 0.4078 | 0.4766 |
| voxel [62] | 0.5556 | 0.4924 | 0.6823 | 0.7123 | 0.4494 | 0.5395 | 0.4223 | 0.5868 | 0.5987 | 0.6221 | 0.4938 | 0.7504 | 0.5507 | 0.5736 |
| NMR [25] | 0.6172 | 0.4998 | 0.7143 | 0.7095 | 0.4990 | 0.5831 | 0.4126 | 0.6536 | 0.6322 | 0.6735 | 0.4829 | 0.7777 | 0.5645 | 0.6015 |
| Ours (sil.) | 0.6419 | 0.5080 | 0.7116 | 0.7697 | 0.5270 | 0.6156 | **0.4628** | 0.6654 | **0.6811** | 0.6878 | 0.4487 | 0.7895 | 0.5953 | 0.6234 |
| Ours (full) | **0.6670** | **0.5429** | **0.7382** | **0.7876** | **0.5470** | **0.6298** | 0.4580 | **0.6807** | 0.6702 | **0.7220** | **0.5325** | **0.8127** | **0.6145** | **0.6464** |

TABLE 1: Comparison of mean IoU with other 3D unsupervised reconstruction methods on 13 categories of ShapeNet datasets.

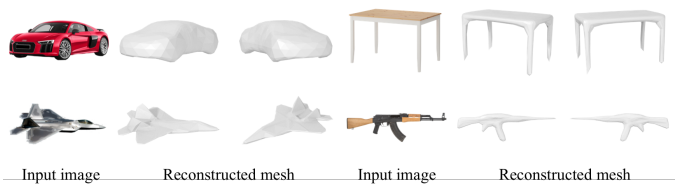



Fig. 10: Single-view reconstruction results on real images.

evaluate our approach on real images. As demonstrated in Figure 10, though only trained on synthetic data, our model generalizes well to real images and novel views with faithful reconstructions and fine-scale details, e.g. the tail fins of the fighter aircraft and thin structures in the rifle and table legs.

#### 5.2.3 Quantitative Evaluations

We show the comparisons on 3D IoU score with the state-of-the-art approaches in Table 1. We test our approach under two settings: one trained with silhouette loss only (sil.) and the other with both silhouette and shading supervisions (full). Our approach has significantly outperformed all the other 3D unsupervised methods on all categories. In addition, the mean score of our best setting has surpassed the state-of-the-art NMR [25] by more than 4.5 points. As we use the identical mesh generator and same training settings with [25], it indicates that it is the proposed SoftRas renderer that leads to the superior performance.

| SoftRas settings | | | $\mathcal{L}_{lap}$ | mIoU (%) |
|---|---|---|---|---|
| distance func. | aggregate func. ($\alpha$) | aggregate func. (color) | | |
| Barycentric | $\mathcal{A}_O$ | - | | 60.8 |
| Euclidean | $\mathcal{A}_O$ | - | | 62.0 |
| Euclidean | $\mathcal{A}_O$ | - | ✓ | 62.4 |
| Euclidean | $\mathcal{A}_N$ | - | ✓ | 63.2 |
| Euclidean | $\mathcal{A}_O$ | $\mathcal{A}_S$ | ✓ | **64.6** |

TABLE 2: Ablation study of the regularizer and various forms of distance and aggregate functions. $\mathcal{A}_N$ is the aggregation function implemented as a neural network. $\mathcal{A}_S$ and $\mathcal{A}_O$ are defined in Equation 8 and 13 respectively.

#### 5.2.4 Ablation Study

**Loss Terms and Alternative Functions.** In Table 2, we investigate the impact of Laplacian regularizer and various forms of the distance function (Section 3.2) and the aggregate function. As the RGB color channel and the $\alpha$ channel (silhouette) have different candidate aggregate functions, we separate their lists in Table 2. First, by adding Laplacian constraint, our performance is increased by 0.4 point (62.4 *v.s.* 62.0). In contrast, NMR [25] has reported a negative effect of geometry regularizer on its quantitative results. The performance drop may be due to the fact that the ad-hoc gradient is not compatible with the regularizer. It is optional to have color supervision on the mesh generation. However, we show that adding a color loss can significantly improve the performance (64.6 *v.s.* 62.4) as more information is leveraged for reducing the ambiguity of using silhouette loss only. In addition, we also show that Euclidean metric usually outperforms the barycentric distance while the aggregate function based on neural network $\mathcal{A}_N$ performs slightly better than the non-parametric counterpart $\mathcal{A}_O$ at the cost of more computations.

### 5.3 Image-based Shape Fitting

**Rigid Pose Fitting.** We compare our approach with NMR in the task of rigid pose fitting. In particular, given a colorized cube and a target image, the pose of the cube needs to be optimized so that its rendered result matches the target image. Despite the simple geometry, the discontinuity of face colors, the non-linearity of rotation and the large occlusions make it particularly difficult to optimize. As shown in Figure 11, NMR is stuck in a local minimum while our approach succeeds to obtain the correct pose. The key is that our method produces smooth and partially transparent renderings which "soften" the loss landscape. Such smoothness can be controlled by $\sigma$ and $\gamma$, which allows us to avoid the local minimum. We also demonstrate the intermediate process of how the proposed SoftRas renderer managed to fit the color cube to the target image in Figure 12. By rendering the cube with stronger blurring at the earlier stage, our approach is able to avoid local minima, and gradually reduce the rendering loss until an accurate pose can be fitted.

Further, we evaluate the rotation estimation accuracy on synthetic data given 100 randomly sampled initializations and targets. We compare methods w/ and w/o scheduling schemes, and summarize mean relative angle error in Table 3. Without optimization scheduling, our method outperforms the best baseline by 10.60°, demonstrating the effectiveness of the gradient flows provided by our method and the benefit of handling largely occluded triangles. Scheduling is a commonly used technique for solving non-linear optimization problems. For other methods, we solve with multi-resolution images in 5 levels; while for our method, we set schedules to decay $\sigma$ and $\gamma$ in 5 steps. Please refer to the *supplemental materials* for details of scheduling. While scheduling improves all methods, our approach still achieves better accuracy than the best baseline by 14.99°, indicating our consistent superiority.

**Non-rigid Shape Fitting.** In Figure 13, we show that SoftRas can provide stronger supervision for non-rigid shape fitting even in the presence of part occlusions. We optimize the human body parametrized by SMPL model [34]. As the right hand (textured as red) is completely occluded in the initial view, it is extremely challenging to fit the body pose to the target image. To obtain correct parameters, the optimization should be able to (1) consider the impact of the occluded part on the rendered image and

*The expectation of uniform-sampled SO3 rotation angle is $\pi/2 + 2/\pi$

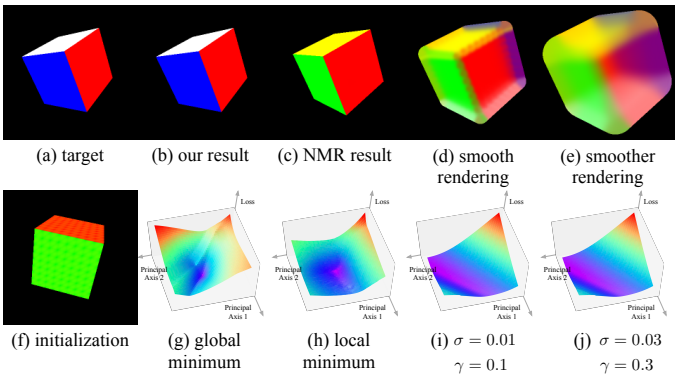



Fig. 11: Visualization of loss function landscapes of NMR and SoftRas for pose optimization given target image (a) and initialization (f). SoftRas achieves global minimum (b) with loss landscape (g). NMR is stuck in local minimum (c) with loss landscape (h). At this local minimum, SoftRas produces the smooth and partially transparent rendering (d)(e), which smoothens the loss landscape (i)(j) with larger $\sigma$ and $\gamma$, and consequently leads to better minimum.

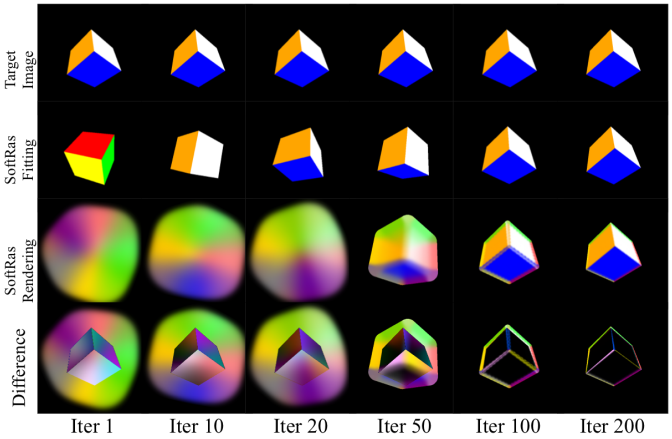



Fig. 12: Intermediate process of fitting a color cube (second row) to a target pose shown in the input image (first row). The smoothened rendering (third row) that is used to escape local minimum, as well as the colorized fitting errors (fourth row), are also demonstrated.

(2) back-propagate the error signals to the occluded vertices. NMR [25] fails to move the hand to the right position due to its incapability to handle occlusions. In comparison, our approach can faithfully complete the task as our probabilistic formulation and aggregating mechanism can take all triangles into account while being able to optimize the $z$ coordinates (depth) of the mesh vertices. In the *supplemental materials*, we further compare the intermediate fitting processes of NMR [25] and SoftRas.

NMR fails to provide efficient signal to advance the optimization of the hand pose. In contrast, our approach is able to obtain

| Method | w/o scheduling | w/ scheduling |
|---|---|---|
| random guess | 126.48°* | 126.48° |
| NMR [25] | 93.40° | 80.94° |
| Li et al. [30] | 95.02° | 78.56° |
| SoftRas | **82.80°** | **63.57°** |

TABLE 3: Comparison of cube rotation estimation error with NMR, measured in mean relative angular error.

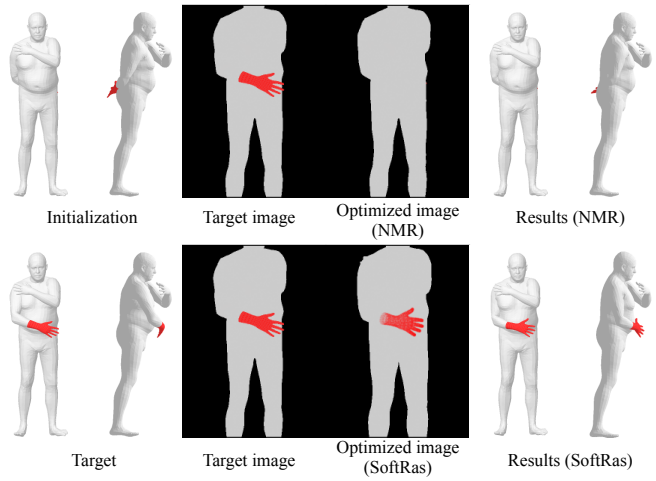



Fig. 13: Results for optimizing human pose given single image.

the correct pose within 320 iterations thanks to the occlusion-aware technique. We wish to emphasize that a proper handling of transparency is the key to passing gradient to occluded vertices. As Rhodin et al. [51] share the similar idea of viewing points as Gaussian blobs with tunable visibility, their method is also able to handle pose estimation even in the presence of occlusion. However, it is not trivial to approximate an arbitrary mesh using a large number of small Gaussians and accurately produce the fine-scale mesh deformations using this approach.

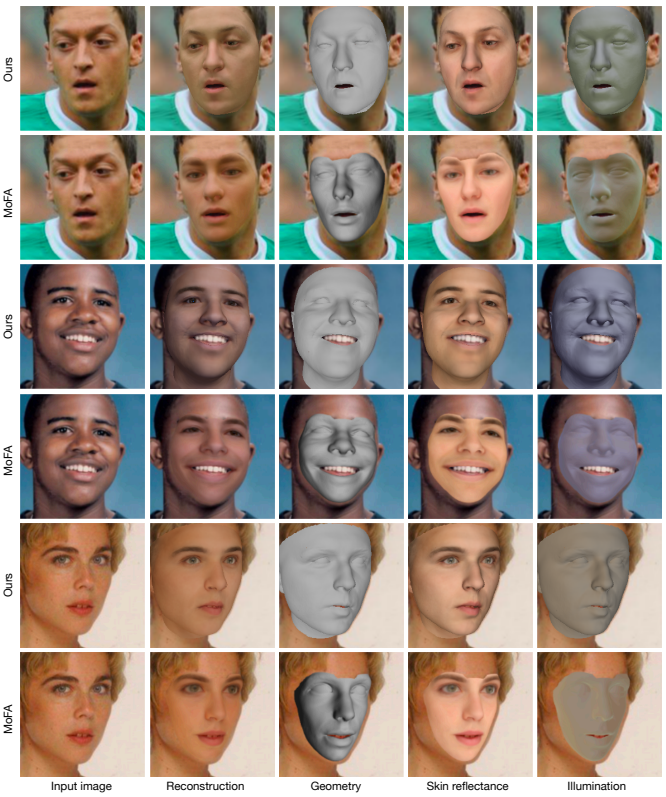



Fig. 14: Results for optimizing facial identity, expression, skin reflectance, lighting and rigid pose given single 2D image along with 2D landmarks.

**Face Reconstruction.** To demonstrate SoftRas is able to provide efficient gradients for multiple properties in inverse rendering problem, we show further experiments to fit face shape, expression, skin reflectance, lighting and rigid pose. Given a single

2D image and facial landmarks, we optimize the coefficients of shape, expression and skin reflectance model that is similar to the one used in MoFA [54], along with lighting modeled by spherical harmonics and rigid pose parameters. The optimization is scheduled in two stages: (1) fitting to 2D landmarks by solving for shape, expression and rigid pose; (2) fitting to photometric loss and 2D landmarks by solving for all parameters (lowering learning rate for rigid pose). We show results in comparison to [54] in Figure 14. We can see our method is able to achieve comparable or better results compare to [54]. We show more results on single-view hand fitting and analysis on smoothing in the *supplemental materials*.

## 6 Conclusions and Discussions

In this paper, we have presented a truly differentiable rendering framework (SoftRas) that is able to directly render a given mesh in a fully differentiable manner. SoftRas can consider both extrinsic and intrinsic variables in a unified rendering framework and generate efficient gradients flowing from pixels to mesh vertices and their attributes (color, normal, *etc.*). We achieve this goal by re-formulating the discrete operations including rasterization and z-buffering as differentiable probabilistic processes. Such ~~novel~~ formulation enables our renderer to flow gradients to unseen vertices and optimize the $z$ coordinates of mesh triangles, leading to significant improvements in the tasks of single-view mesh reconstruction and image-based shape fitting. However, our approach, in current form, cannot handle shadows and topology changes, which are worth investigation in the future. In addition, our approach do not provide gradients to the UV coordinates. However, given a fixed UV mapping, we are able to pass gradient to the colors in texture map. It would be interesting to explore simultaneous optimization of UV mapping and texture map in the future work.

## References


[1] L. Bavoil and K. Myers. Order independent transparency with dual depth peeling. *NVIDIA OpenGL SDK*, pages 1–12, 2008.

[2] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[3] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on pattern analysis and machine intelligence*, 25(9):1063–1074, 2003.

[4] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *European Conference on Computer Vision*, pages 561–578. Springer, 2016.

[5] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.

[6] L. Carpenter. The a-buffer, an antialiased hidden surface method. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 103–108, 1984.

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[8] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[9] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):681–685, 2001.

[10] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (TOG)*, 37(4):128, 2018.

[11] E. Enderton, E. Sintorn, P. Shirley, and D. Luebke. Stochastic transparency. *IEEE transactions on visualization and computer graphics*, 17(8):1036–1047, 2010.

[12] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2010.

[13] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8377–8386, 2018.

[14] I. Gkioulekas, A. Levin, and T. Zickler. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *European Conference on Computer Vision*, pages 685–701. Springer, 2016.

[15] I. Gkioulekas, S. Zhao, K. Bala, T. Zickler, and A. Levin. Inverse volume rendering with material dictionaries. *ACM Transactions on Graphics (TOG)*, 32(6):162, 2013.

[16] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Atlasnet: A papier-mch approach to learning 3d surface generation. *computer vision and pattern recognition*, 2018.

[17] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] P. Henderson and V. Ferrari. Learning to generate and reconstruct 3d meshes with only 2d supervision. In *British Machine Vision Conference (BMVC)*, 2018.

[19] Z. Huang, T. Li, W. Chen, Y. Zhao, J. Xing, C. LeGendre, L. Luo, C. Ma, and H. Li. Deep volumetric video from very sparse multi-view performance capture. In *European Conference on Computer Vision*, pages 351–369. Springer, 2018.

[20] L. Huynh, W. Chen, S. Saito, J. Xing, K. Nagano, A. Jones, P. Debevec, and H. Li. Mesoscopic facial geometry inference using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8407–8416, 2018.

[21] E. Insafutdinov and A. Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems*, pages 2802–2812, 2018.

[22] J. Jansen and L. Bavoil. Fourier opacity mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 165–172, 2010.

[23] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018.

[24] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. *arXiv preprint arXiv:1803.07549*, 2018.

[25] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[26] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] A. Kundu, Y. Li, and J. M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3559–3568, 2018.

[29] H. Lensch, J. Kautz, M. Goesele, W. Heidrich, and H.-P. Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Transactions on Graphics (TOG)*, 22(2):234–257, 2003.

[30] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[31] F. Liu, C. Shen, G. Lin, and I. D. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10):2024–2039, 2016.

[32] F. Liu, D. Zeng, Q. Zhao, and X. Liu. Joint face alignment and 3d face reconstruction. In *European Conference on Computer Vision*, pages 545–560. Springer, 2016.

[33] G. Liu, D. Ceylan, E. Yumer, J. Yang, and J.-M. Lien. Material editing using a physically based rendering network. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2280–2288. IEEE, 2017.

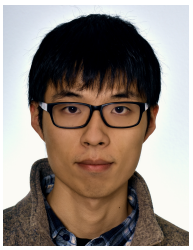
[34] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):248, 2015.

[35] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014.

[36] G. Loubet, N. Holzschuch, and W. Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Transactions on*

*Graphics (TOG)*, 38(6):1–14, 2019.
[37] V. K. Mansinghka, T. D. Kulkarni, Y. N. Perov, and J. Tenenbaum. Approximate bayesian image interpretation using generative probabilistic graphics programs. In *Advances in Neural Information Processing Systems*, pages 1520–1528, 2013.
[38] I. Masi, S. Rawls, G. Medioni, and P. Natarajan. Pose-aware face recognition in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4838–4846, 2016.
[39] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374. ACM Press/Addison-Wesley Publishing Co., 2000.
[40] M. Maule, J. Comba, R. Torchelsen, and R. Bastos. Hybrid transparency. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 103–118, 2013.
[41] M. McGuire and L. Bavoil. Weighted blended order-independent transparency. *Journal of Computer Graphics Techniques*, 2013.
[42] H. Meshkin. Sort-independent alpha blending. *GDC Talk*, 2007.
[43] K. Myers and L. Bavoil. Stencil routed a-buffer. In *SIGGRAPH Sketches*, page 21, 2007.
[44] O. Nalbach, E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel. Deep shading: convolutional neural networks for screen space shading. In *Computer graphics forum*, volume 36, pages 65–78. Wiley Online Library, 2017.
[45] T. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. *arXiv preprint arXiv:1806.06575*, 2018.
[46] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob. Mitsuba 2: a retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6):203, 2019.
[47] J. Pan, X. Han, W. Chen, J. Tang, and K. Jia. Deep mesh reconstruction from single rgb images via topology modification networks. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.
[48] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7025–7034, 2017.
[49] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia. Geonet: Geometric neural network for joint depth and surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 283–291, 2018.
[50] D. J. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems*, pages 4996–5004, 2016.
[51] H. Rhodin, N. Robertini, C. Richardt, H.-P. Seidel, and C. Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proceedings of the 2015 International Conference on Computer Vision (ICCV 2015)*, 2015.
[52] E. Richardson, M. Sela, R. Or-El, and R. Kimmel. Learning detailed face reconstruction from a single image. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5553–5562. IEEE, 2017.
[53] A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, and C. Theobalt. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2549–2559, 2018.
[54] A. Tewari, M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Pérez, and C. Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 2, page 5, 2017.
[55] L. Tran and X. Liu. Nonlinear 3d face morphable model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7346–7355, 2018.
[56] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015.
[57] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, volume 1, page 3, 2017.
[58] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018.
[59] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–547, 2015.
[60] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.
[61] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018.
[62] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 1696–1704, 2016.
[63] C. Zhang, L. Wu, C. Zheng, I. Gkioulekas, R. Ramamoorthi, and S. Zhao. A differential theory of radiative transfer. *ACM Trans. Graph.*, 38(6), 2019.
[64] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706, 1999.
[65] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European conference on computer vision*, pages 286–301. Springer, 2016.
[66] J. Zienkiewicz, A. Davison, and S. Leutenegger. Real-time height map fusion using differentiable rendering. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4280–4287. IEEE, 2016.

**Shichen Liu** received his B.E degree in software engineering from Tsinghua University in 2018. He is currently a Ph.D. student in University of Southern California. His research interests include computer vision and graphics, 3D deep learning and representation learning.



**Tianye Li** is currently a Ph.D. student in computer science at University of Southern California (USC). He received his B.Eng. in electronic and information engineering from Xidian University in 2013, and M.S. in electrical engineering from USC in 2015. His research interests include computer vision and computer graphics, especially in capturing, modeling and understanding human and environment.

**Weikai Chen** is a senior research scientist at Tencent America. Previously, he was a postdoc and then research associate at the Vision and Graphics Lab (VGL), USC ICT. He obtained his PhD degree from Department of Computer Science, the University of Hong Kong in 2017. His research lies in the interplay among computer graphics, computer vision and deep learning. In particular, he is interested in 3D deep vision, image-based 3D content reasoning, human digitization and geometric modeling.

**Hao Li** is the CEO/Co-Founder of Pinscreen. He was previously an Associate Professor of Computer Science at the University of Southern California, the Director of the Vision and Graphics Lab at the USC Institute for Creative Technologies, a research lead at Industrial Light & Magic / Lucasfilm, and a postdoctoral fellow at Columbia and Princeton Universities. He was named top 35 innovator under 35 by MIT Technology Review in 2013 and was also awarded the Google Faculty Award, the Okawa Foundation Research Grant, as well as the Andrew and Erna Viterbi Early Career Chair. He won the Office of Naval Research (ONR) Young Investigator Award in 2018 and was named to the DARPA ISAT Study Group in 2019. Hao obtained his PhD at ETH Zurich and his MSc at the University of Karlsruhe (TH).