Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Full length article

Analytical derivatives for differentiable renderer: 3D pose estimation by silhouette consistency[☆]

Zaiqiang Wu[a], Wei Jiang[a,*], Hongyan Yu[b]

[a] The State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China
[b] Beijing Electro-mechanical Engineering Institute, Beijing 100074, China

ARTICLE INFO

Keywords:
Inverse graphics
Differentiable renderer
3D pose estimation

ABSTRACT

Differentiable renderer is widely used in optimization-based 3D reconstruction which requires gradients for optimization. The existing differentiable renderers obtain gradients via numerical techniques. However, these methods are inaccurate and inefficient. Motivated by this fact, we propose a differentiable renderer with analytical gradients. The main obstacle of traditional renderer being differentiable is the discrete sampling operation of rasterization. To obtain a differentiable rasterization renderer, we define pixel intensity as a double integral over the pixel grid, and then derive the analytical gradients with respect to vertices. 3D pose estimation by multi-viewpoint silhouettes is conducted to reveal the effectiveness and efficiency of the proposed method. Experimental results show that 3D pose estimation without 3D and 2D joint supervision can produce competitive results. The findings also indicate that the proposed method has higher accuracy and efficiency than previous differentiable renderers.

1. Introduction

In recent years, deep learning techniques have achieved impressive success in computer vision. Researchers can design various network structures according to their specific intentions. All operations in the network must be differentiable to enable the training of network parameters by back-propagation algorithm. However, some operations cannot obtain derivatives trivially. For instance, tasks of shape-from-silhouette require differentiable rendering operation to construct the loss function for supervision. However, the traditional rendering algorithms (e.g., rasterization and ray tracing [1]) are not differentiable and cannot be directly applied to deep learning framework due to the discrete sampling operation.

To address this issue, many researchers attempted to develop a differentiable renderer to incorporate the rendering operation into gradient-based optimization framework. To the best of our knowledge, de La Gorce et al. [2] were the earliest to apply differentiable renderer in 3D hand pose estimation. Loper et al. [3] proposed a general-purpose differentiable renderer named OpenDR which can render triangular meshes into images and automatically acquire derivatives with respect to the model parameters. However, the derivatives of OpenDR are computed by numerical methods which lack accuracy. Moreover, OpenDR is incompatible with the existing deep learning framework. Kato et al. [4] proposed a differentiable renderer for neural networks, but this method still relies on numerical methods to compute derivatives. Liu et al. [5] proposed a differentiable renderer named SoftRas which only focuses on the rendering of silhouette. This method generates probability maps for each triangle in the mesh, causing high memory consumption and blurry rendering results.

We present a differentiable silhouette renderer with analytical derivatives to address these problems. Our work differs from other works [2–4] in that our renderer only focuses on synthesizing silhouettes and obtains derivatives by analytical approach. In most cases, only silhouette information is available in shape-form-silhouette tasks. Therefore, a differentiable renderer, which only focusing on synthesizing silhouettes, is significant and sufficient for 3D reconstruction. The forward pass of our renderer, just like other differentiable renderers, is similar to rasterization with anti-aliasing. However, the backward pass of our differentiable renderer is different from those of prior methods that depend on the access to frame buffers and the derivatives obtained by numerical methods. The highlight of our work is that the derivatives of pixel intensities with respect to the coordinates of vertices can be obtained by our proposed analytical method without the need to access the frame buffers repeatedly and apply any numerical method.

Our strategy for avoiding discrete sampling is to define pixel intensity as the average value of the area within the pixel region. Pixel

☆ This paper has been recommended for acceptance by Dr Zicheng Liu.
* Corresponding author.
E-mail address: jiangwei_zju@zju.edu.cn (W. Jiang).

https://doi.org/10.1016/j.jvcir.2020.102960
Received 1 November 2019; Received in revised form 27 August 2020; Accepted 24 October 2020
Available online 27 October 2020

intensity can be computed by a double integral over the pixel region of the pixel intensity function. On the basis of the integral expression of pixel intensity, the expression of derivatives can be derived and simplified to an analytical form without integral operation. Only silhouette is considered in this study, and thus we do not need to deal with self-occlusion. The proposed analytical derivatives can facilitate the backward pass implementation and improve its efficiency. Our main contributions are summarized as follows:

- We present a novel non-numerical method to obtain the analytical derivatives with respect to vertex location for silhouette renderer. In addition, our algorithm can be implemented on GPU and offer significant speedup in high resolution setups.
- Our experimental results of 3D pose estimation reveal that our proposed method has higher accuracy and efficiency than prior unsupervised methods.
- The potential of 3D pose estimation by silhouette consistency without 2D or 3D joints is shown in the conducted experiments.

2. Related work

2.1. Differentiable renderer

Computer vision problems have long been regarded as inverse graphics in the literature. In computer graphics, we aim to render an image from object shape, texture, and illumination. On the contrary, inverse graphics aims to estimate object shape, texture, and illumination from observed images. Differentiable rendering offers a straightforward and practical technique to infer the parameters of 3D models by gradient-based methods.

Many model-based approaches have been proposed for differentiable rendering. Gkioulekas et al. [6] developed an algorithmic framework to infer internal scattering parameters for heterogeneous materials. Gradients are leveraged for optimization to solve this inverse problem, but this approach is limited to specific illumination problems. Mansinghka et al. [7] introduced a probabilistic graphic model to estimate scene parameters from observations. de La Gorce et al. [2] proposed to differentiate the graphics rendering pipeline and use anti-aliasing to obtain derivatives at object boundaries. We apply similar techniques, but we present an analytical method for computing boundary derivatives. Loper and Black [3] introduced an approximate differentiable renderer called OpenDR, which is used to render a 3D model and automatically obtain derivatives with respect to the model parameters. However, OpenDR has no interfaces to popular deep learning library; thus, it is difficult to incorporate into deep learning frameworks. Kato et al. [4] introduced a differentiable rendering pipeline that approximates the rasterization gradient with a hand-designed function. Li et al. [8] recently presented a differentiable ray tracer that can compute derivatives of scalar function over the rendered image with respect to arbitrary scene parameters. Nevertheless, the forward and backward passes of this method are performed by Monte Carlo ray tracing. Thus, this technique is time consuming and impractical to be incorporated into learning-based frameworks.

With the development of deep learning, the forward and backward passes of differentiable rendering have been achieved in a deep learning framework [9–16]. Nguyen-Phuoc et al. [17] presented RenderNet, a convolutional network that learns a direct map from scene parameters to corresponding rendered images. However, RenderNet is computationally expensive because it is composed of convolutional networks.

In the present study, we explore a rasterization-based differentiable renderer with analytical derivatives. The main difference between our method and Neural 3D Mesh Render [4] (N3MR) is that instead of approximating the derivatives with hand-designed functions, we derive an analytical expression to obtain derivatives with considerably high efficiency and accuracy.

2.2. Image-based 3D reconstruction

Inferring 3D shape from images is a traditional and challenging problem in computer vision. With the surge of deep learning, 3D reconstruction from monocular images has become an active research topic.

Most learning-based approaches learn the mapping from 2D images to 3D shapes with 3D supervision. Several methods predict a depth map to reconstruct 3D shapes [18,19], whereas others directly predict 3D shapes [4,20–25]. However, those methods lack details in the generated results.

To generate detailed 3D shape, deformable models have been incorporated into 3D reconstruction framework. For example, statistical body shape models, such as SMPL [26] and SCAPE [27], are frequently used in 3D pose estimation to generate 3D body meshes. Bogo et al. [28] proposed an iteratively optimization-based approach to reconstruct 3D human poses and shapes from single image by minimizing the reprojection error between 2D images and statistical body shape models. Pavlakos et al. [29] presented an end-to-end framework to predict the parameters of a statistical body shape model by training neural networks with monocular images and 3D ground truth.

The 3D ground truth shapes are difficult to obtain. Therefore, 3D reconstruction in unsupervised manner also attracts increasing attention. Yan et al. [30] proposed perspective transformer nets to infer 3D voxels from silhouette images in multiple viewpoints. Kato et al. [4] proposed to reconstruct 3D shapes by training a mesh deforming network with only 2D silhouette supervision. We follow these works in supervision, but we use a statistical body shape model named SMPL to represent the 3D shape of human body and optimize the 3D pose with the gradients obtained by our proposed differentiable renderer.

3. Analytical derivatives for rasterization

In computer graphics, rasterization is the task of computing the mapping from scene geometry described in vector graphic format to raster images. The main obstacle that impedes rasterization from being differentiable is the discrete sampling operation, in which pixel intensity is sampled only at the central position of each pixel grid. In addition, aliasing effect often appears in the rendered images due to discrete sampling operation and limited sampling frequency.

One of the most commonly used anti-aliasing techniques is supersampling, which can reduce the magnitude of the discontinuous intensity changes. Inspired by this approach, we can reasonably conclude that if the sampling frequency is increased to infinity, then the sampling operation becomes continuous and differentiable. However, because infinite sampling frequency cannot be achieved in practice, the sampling rate is set to a relatively high value to approximate the ideal situation. The forward pass of our renderer works similarly to the standard graphics pipeline with anti-aliasing. On the contrary, the backward derivatives are derived under the hypothesis that the image is sampled with infinite sampling rate and then down-sampled by an average filter.

3.1. Forward rendering

The forward pass of our proposed differentiable renderer follows the standard graphics method [31]. We apply anti-aliasing to smooth the output images to ensure the consistency between the forward and backward propagation.

Considering the rendering process of the Stanford Bunny model, let $p(x,y)$ be the corresponding continuous distribution of intensity in the screen space, as shown in Fig. 1. Then rasterization can be reinterpreted as sampling each pixel intensity value within the pixel grid.

We only focus on synthesizing silhouettes; thus, only two possible values exist for $p(x,y)$, namely, the foreground intensity p_1 and the background intensity p_0. We suppose that the output image has H rows

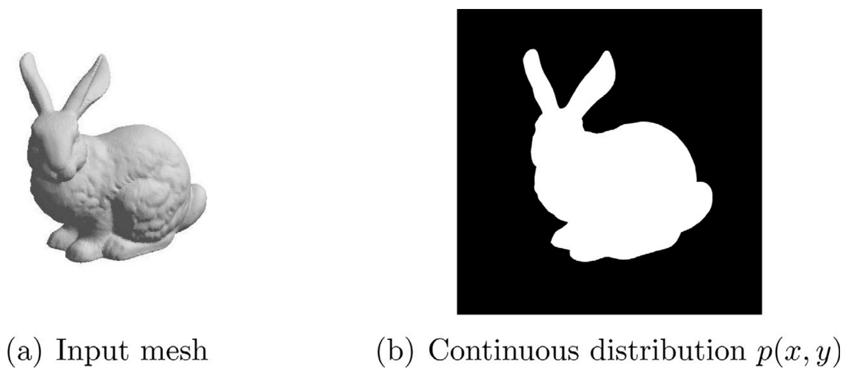

Fig. 1. (a) The input Stanford Bunny model and (b) the corresponding continuous intensity distribution in screen space.

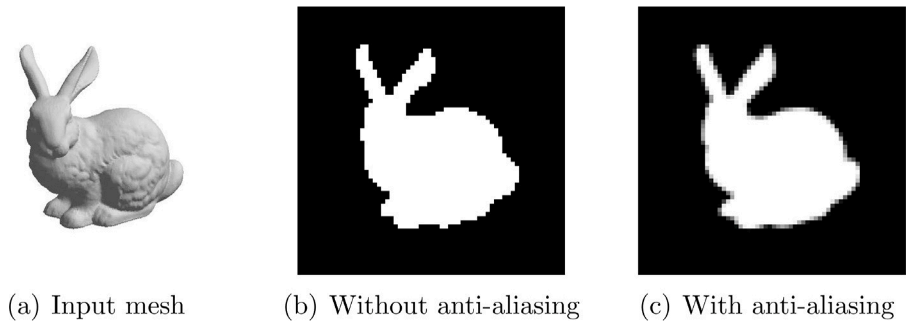

Fig. 2. The forward rendering process of our differentiable renderer. To make the output silhouette image more smooth, we first render a silhouette image with higher resolution then down-sample it to get the final output image.

and W columns, and the pixel intensity $I(i,j)$ in the ith row and jth column can be presented as follows:

$$I(i,j) = \frac{1}{S} \iint_{\Omega_{i,j}} p(x,y)\,dx\,dy \tag{1}$$

where $\Omega_{i,j}$ represents the pixel grid in the ith row and jth column, and S denotes the area of the pixel grid.

However, computing the integral expression in Eq. (1) analytically by programming is difficult. Therefore, we apply anti-aliasing to evaluate the integral numerically, as shown in Fig. 2.

Let F be the multiple of super-sampling, then the pixel intensity of the ith row and jth column can be expressed as follows:

$$I(i,j) = \frac{1}{F^2} \sum_{k=1}^{F^2} p(x_k, y_k) \tag{2}$$

where (x_k, y_k) denotes the coordinate of the kth sampling point in screen space.

Evidently, we can obtain the following:

$$\lim_{F\to\infty} \frac{1}{F^2} \sum_{k=1}^{F^2} p(x_k, y_k) = \frac{1}{S} \iint_{\Omega_{i,j}} p(x,y)\,dx\,dy \tag{3}$$

In the forward rendering pass, high accuracy can be achieved by setting F to a large value, but at the cost of additional performance overhead. For the trade-off between accuracy and efficiency, we set F to 4 in our implementation.

3.2. Derivative computation

The following derivation is based on the assumption that all pixel intensities are obtained by Eq. (1). Consider an edge consisted of vertices v_a and v_b located at silhouette boundary, the coordinates of v_a and v_b are denoted as (x_0, y_0) and (x_1, y_1), respectively. This edge is assumed to intersect with the pixel grid in the ith row and jth column. The partial derivative $\frac{\partial I(i,j)}{\partial x_0}$ can be written as follows:

$$\begin{aligned}\frac{\partial I(i,j)}{\partial x_0} &= \frac{\partial \frac{1}{S} \iint_{\Omega_{i,j}} p(x,y)\,dx\,dy}{\partial x_0} \\ &= \frac{1}{S} \iint_{\Omega_{i,j}} \frac{\partial p(x,y)}{\partial x_0}\,dx\,dy\end{aligned} \tag{4}$$

For notational convenience, we denote that $A = y_1 - y_0$, $B = x_0 - x_1$, and $C = x_1 y_0 - x_0 y_1$. Then the edge equation can be presented as follows:

$$\alpha(x,y) = Ax + By + C \tag{5}$$

We suppose that if $\alpha(x,y) < 0$, then point (x,y) is located in the foreground region, and vice-versa. Let Ω_0 be an appropriate subregion of $\Omega_{i,j}$ such that Ω_0 only covers the edge connecting v_a and v_b. Thus, the intensity distribution function $p(x,y)$ can be written as follows:

$$p(x,y) = \begin{cases} p_1, & \text{if } \alpha(x,y) < 0 \text{ and } (x,y) \in \Omega_0 \\ p_0, & \text{if } \alpha(x,y) > 0 \text{ and } (x,y) \in \Omega_0 \end{cases} \tag{6}$$

The equation above can be simplified with the Heaviside step function h, as follows:

$$p(x,y) = p_0 h(\alpha(x,y)) + p_1 h(-\alpha(x,y)), (x,y) \in \Omega_0 \tag{7}$$

Given that $\frac{\partial I(i,j)}{\partial x_0} \neq 0$ only if $(x,y) \in \Omega_0$, thus the partial derivative $\frac{\partial I(i,j)}{\partial x_0}$ can be further rewritten as follows:

$$\begin{aligned}\frac{\partial I(i,j)}{\partial x_0} &= \frac{1}{S} \iint_{\Omega_0} \frac{\partial p(x,y)}{\partial x_0}\,dx\,dy + \frac{1}{S} \iint_{\Omega_{i,j}-\Omega_0} \frac{\partial p(x,y)}{\partial x_0}\,dx\,dy \\ &= \frac{1}{S} \iint_{\Omega_0} \frac{\partial p(x,y)}{\partial x_0}\,dx\,dy\end{aligned} \tag{8}$$

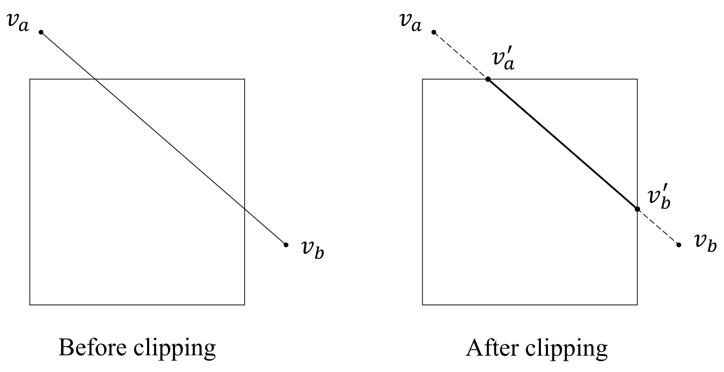

Fig. 3. Illustration of how to determine the lower and upper limits of the integral by Liang–Barsky algorithm. After clipping, two new endpoints v'_a and v'_b are obtained. Following the variable substitution, the coordinates of v'_a and v'_b can be transformed to the lower limit k_0 and upper limit k_1.

Eqs. (7) and (8) are used to obtain the partial derivative $\frac{\partial I(i,j)}{\partial x_0}$ as follows:

$$\begin{aligned}\frac{\partial I(i,j)}{\partial x_0} &= \frac{1}{S}\iint_{\Omega_0} p_0\delta(\alpha(x,y))\frac{\partial\alpha(x,y)}{\partial x_0} - p_1\delta(\alpha(x,y))\frac{\partial\alpha(x,y)}{\partial x_0}\,dx\,dy \\ &= \frac{p_1-p_0}{S}\iint_{\Omega_0}\delta(\alpha(x,y))(-\frac{\partial\alpha(x,y)}{\partial x_0})\,dx\,dy\end{aligned} \tag{9}$$

where δ denotes the Dirac delta function.

Substituting Eq. (5) into Eq. (9), the partial derivative $\frac{\partial I(i,j)}{\partial x_0}$ can be presented as follows:

$$\frac{\partial I(i,j)}{\partial x_0} = \frac{p_1-p_0}{S}\iint_{\Omega_0}\delta(Ax+By+C)(y_1-y)\,dx\,dy \tag{10}$$

To eliminate the Dirac delta function δ, we perform the following variable substitution:

$$\begin{cases} t = Ax+By \\ k = -Bx+Ay \end{cases} \tag{11}$$

After variable substitution, Eq. (10) can be rewritten as follows:

$$\begin{aligned}\frac{\partial I(i,j)}{\partial x_0} &= \frac{p_1-p_0}{S(A^2+B^2)}\iint\delta(t+C)(y_1-\frac{Bt+Ak}{A^2+B^2})\,dt\,dk \\ &= \frac{p_1-p_0}{S(A^2+B^2)}\int_{k_0}^{k_1}(y_1-\frac{Ak-BC}{A^2+B^2})\,dk \\ &= \frac{p_1-p_0}{S(A^2+B^2)}((y_1+\frac{BC}{A^2+B^2})(k_1-k_0)-\frac{A(k_1^2-k_0^2)}{2(A^2+B^2)})\end{aligned} \tag{12}$$

where A^2+B^2 is the L^2 length of the edge, which considers the Jacobian of the variable substitution. k_0 and k_1 are the lower and upper limits of integral, respectively.

To specify k_0 and k_1, we apply Liang–Barsky algorithm [32] to remove the portion of segment v_av_b outside the pixel grid. The two new endpoints after clipping are denoted as v'_a and v'_b, and their coordinates are denoted as (x'_0, y'_0) and (x'_1, y'_1) respectively. Fig. 3 illustrates the procedure of obtaining the lower and upper limits as follows:

$$\begin{cases} k_0 &= -Bx'_0 + Ay'_0 \\ k_1 &= -Bx'_1 + Ay'_1 \end{cases} \tag{13}$$

Similarly, we can obtain the partial derivatives $\frac{\partial I(i,j)}{\partial y_0}$, $\frac{\partial I(i,j)}{\partial x_1}$ and $\frac{\partial I(i,j)}{\partial y_1}$ as follows:

$$\frac{\partial I(i,j)}{\partial y_0} = -\frac{p_1-p_0}{S(A^2+B^2)}((x_1+\frac{AC}{A^2+B^2})(k_1-k_0)+\frac{B(k_1^2-k_0^2)}{2(A^2+B^2)}) \tag{14}$$

$$\frac{\partial I(i,j)}{\partial x_1} = \frac{p_1-p_0}{S(A^2+B^2)}(-(y_0+\frac{BC}{A^2+B^2})(k_1-k_0)+\frac{A(k_1^2-k_0^2)}{2(A^2+B^2)}) \tag{15}$$

$$\frac{\partial I(i,j)}{\partial y_1} = \frac{p_1-p_0}{S(A^2+B^2)}((x_0+\frac{AC}{A^2+B^2})(k_1-k_0)+\frac{B(k_1^2-k_0^2)}{2(A^2+B^2)}) \tag{16}$$

The derivatives can be obtained without any numerical method by using these analytical expressions. Thus, further improvement in accuracy and efficiency can be achieved.

3.3. Backward gradient flow

We consider a 3D mesh consisting of a set of vertices $\{v_1{}^\circ, v_2{}^\circ, \ldots, v^0_{N_v}\}$ and faces $\{f_1, f_2, \ldots, f_{N_f}\}$. $v^o_k \in \mathbb{R}^3$ represents the position of the kth vertex in the 3D object space, and $f_k \in \mathbb{N}^3$ represents the indices of the three vertices corresponding to the kth triangle face. To render this 3D mesh, the vertices $\{v^o_k\}$ in the object space are projected into screen space as vertices $\{v_k\}, v_k \in \mathbb{R}^2$.

The scalar loss function over the rendered image for optimization is denoted as L. The partial derivatives $\{\frac{\partial L}{\partial I(i,j)} | i = 1,\ldots,H, j = 1,\ldots,W\}$ can be computed through automatic differentiable libraries, such as PyTorch and TensorFlow.

Our task is as follows: given the partial derivatives of loss function L with respect to pixel intensities $\{\frac{\partial L}{\partial I(i,j)}\}$, we compute the derivatives of pixel intensities with respect to vertices $\{\frac{\partial I(i,j)}{\partial v_k}\}$, and then obtain the derivatives $\{\frac{\partial L}{\partial v_k}\}$ using chain rule. The remaining of the work is preformed by automatic differentiable libraries.

A key observation is that the gradient flow is sparse, because $\frac{\partial I(i,j)}{\partial v_k} \neq 0$ only if at least one edge consist of v_k intersected with the pixel grid of $I(i,j)$. This condition allows us to focus on specific i, j, and k, such that $\frac{\partial I(i,j)}{\partial v_k} \neq 0$. Therefore we can skip the pixels with zero gradient contribution to the current triangle when traversing pixel grids, and thus improve the efficiency.

In addition, pixels out of the bounding box of the current triangle can be excluded because they do not have nonzero gradient contributions. Furthermore, we adopt the Liang–Barsky clipping algorithm [32] to determine whether a pixel is intersected with current triangle. As shown in Fig. 4, a pixel intersects with the triangle only if at least one edge of the triangle is intersected with the pixel grid.

To further improve the efficiency of our method, we exploit the fact that only pixels at the object boundary have nonzero gradient contribution. Therefore, edge detection is performed on the rendered image and computation is only required at those boundary pixels, as shown in Fig. 5.

Consider a pixel at the boundary in the ith row and jth column, the partial derivative of pixel intensity with respect to the location of kth vertices v_k, denoted as $\frac{\partial I(i,j)}{\partial v_k}$, is to be computed. We assume that there are N_e edges consisted of v_k intersecting with the pixel in the ith row and jth column. The derivative of the pixel intensity $I(i,j)$ with respect to the location of v_k can be expressed as follows:

$$\frac{\partial I(i,j)}{\partial v_k} = \begin{cases} \sum_{n=1}^{N_e}\frac{\partial I(i,j)}{\partial v^n_k}, & \text{if } N_e > 0 \\ 0, & \text{if } N_e = 0 \end{cases} \tag{17}$$

where $\frac{\partial I(i,j)}{\partial v^n_k}$ represents the derivative computed by the nth edge.

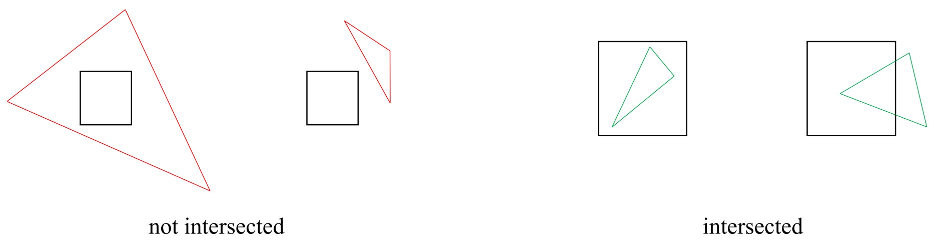

Fig. 4. Several intuitive examples illustrating how to determine whether a triangle is intersected with a pixel grid.

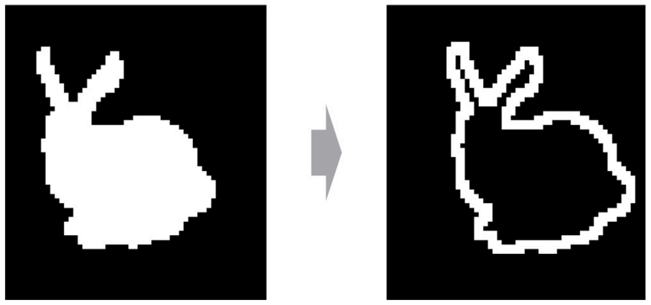

Fig. 5. The boundary pixels are extracted to reduce computational efforts.

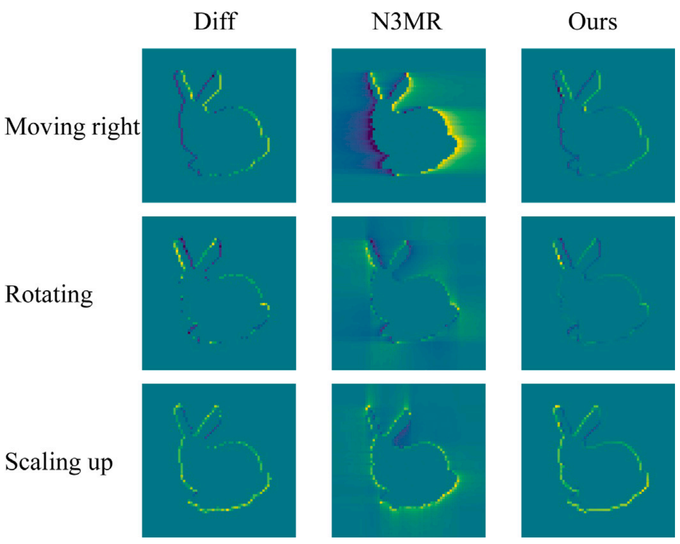

Fig. 6. Comparison of per-pixel gradient among our method, N3MR, and finite difference (Diff). From the first row to the third row, the gradients are with respect to moving right, rotating anti-clockwise, and scaling up respectively.

3.4. Per-pixel gradient visualization

We visualize per-pixel gradients generated by our differentiable renderer, N3MR and finite difference to demonstrate the effectiveness of our method. In finite difference method, gradients are approximated by subtracting two rendered images and then dividing the result by the small change in object parameter, thereby resulting a good approximation of per-pixel gradients.

As shown in Fig. 6, our per-pixel gradients are roughly the same as the finite difference. In addition, our method produces smoother and more reasonable results than finite difference because finite difference introduces errors from numerical approximation in the forward rendering pass. In the visualized results of N3MR, nonzero gradients are found outside the object silhouette, which is unreasonable because gradients should be zero except for the boundary of the object.

In summary, our differentiable renderer can generate accurate gradients with respect to vertex location, and thus enable the gradient-based optimization for 3D pose estimation.

4. 3D pose estimation

We perform experiments of 3D pose estimation based on a statistical body shape model by using our proposed differentiable silhouette renderer to show the effectiveness of our method. Following the work of [28], an iteratively optimization-based method is presented to estimate the pose parameters of the statistical body shape model by minimizing the error between reprojected and ground truth silhouettes. The images and 3D ground truth leveraged in the experiments are obtained from a 3D pose dataset named UP-3D [33]. Different from previous works, this study does not necessitate 2D and 3D joint ground truth for experiments of 3D pose estimation.

4.1. Statistical body shape model

We use a statistical body shape model named SMPL [26] as our representation of 3D poses. Essential notations of the SMPL model are provided here. The SMPL model can be viewed as a function $\mathcal{M}(\beta,\theta;\Phi)$, where β denotes the shape parameters, θ denotes the pose parameters, and Φ denotes the fixed parameters learned from a dataset with body scans [34]. The outputs of the SMPL function are vertices $P \in \mathbb{R}^{N\times 3}$ with $N = 6890$ of a body mesh. The shape parameters $\beta \in \mathbb{R}^{10}$ are the linear coefficients of the principal body shapes. The pose parameters $\theta \in \mathbb{R}^{24\times 3}$ are expressed in axis and angle representations; they define the relative rotation among parts of the skeleton. The 3D joints $J \in \mathbb{R}^{24\times 3}$ are obtained conveniently by a sparse linear combination of mesh vertices.

The shape parameters β are fixed in our experiments, and our task is to optimize the pose parameters θ to minimize the errors between the ground truth and reprojected silhouettes.

4.2. Data preparation

We assume that only images and multi-viewpoint silhouettes are available in the 3D pose estimation task. The ground truth silhouettes are generated by rendering the 3D ground truth meshes of UP-3D [33] from four azimuth angles (with step of 90°) with fixed elevation angles (0°) under the same camera setup illustrated in Fig. 7. The resolution of the silhouettes is set to 64×64.

4.3. Method

Given a single image I and its multi-viewpoint 2D silhouettes $\{S_i\}$, we optimize the pose parameters of SMPL by minimizing a weighted sum of error terms. The differentiable silhouette rendering process is

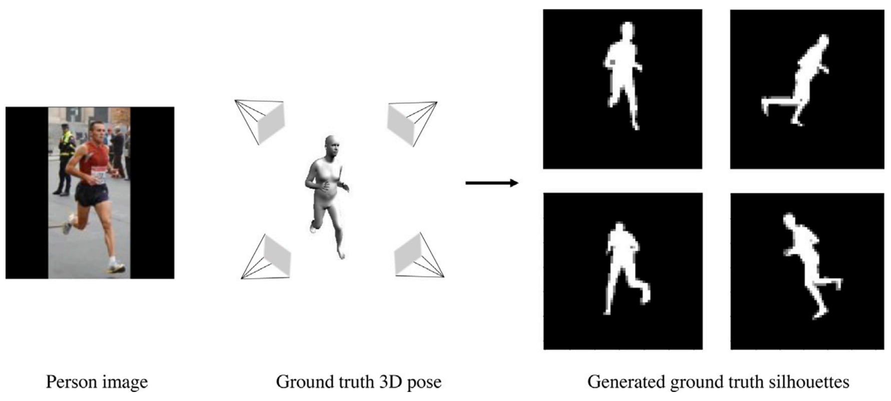

Fig. 7. The ground truth silhouettes for supervision are generated by projecting the ground truth 3D model to the image plane by cameras in different viewpoints.

denoted as $\mathcal{R}$, and the silhouette error term E_{sl} can be presented as follows:

$$\begin{aligned} E_{sl} &= \sum_{i=1}^{N_s} \|\mathcal{R}_i(\hat{P}) - \mathcal{R}_i(P)\|_2^2 \\ &= \sum_{i=1}^{N_s} \|\mathcal{R}_i(\hat{P}) - S_i\|_2^2 \\ &= \sum_{i=1}^{N_s} \|\mathcal{R}_i(\mathcal{M}(\beta,\theta;\Phi)) - S_i\|_2^2 \end{aligned} \tag{18}$$

where P and $\hat{P}$ denote the ground truth and estimated vertices, respectively. N_s denotes the total number of silhouettes, $\mathcal{R}_i$ denotes the camera in the ith position, and S_i denotes the ith ground truth silhouette.

In addition, a self-intersection penalty term (SPT) E_{spt} from [35] is adopted to prevent the body model from self-intersection. The SPT term can be presented as follows:

$$E_{spt} = \frac{N_{sec}}{N_v} \tag{19}$$

where N_{sec} denotes the number of vertices in the self-intersection region, and N_v denotes the total number of vertices.

The backward gradients of E_{spt} are obtained by a hand-designed algorithm which can produce gradients to pull vertices out of the self-intersection region. The details of this algorithm are beyond the scope of this study, we refer the interested readers to [35] for additional details.

Overall, the objective function can be written as the weighted sum of the two error terms above, namely

$$E = E_{sl} + \lambda E_{spt} \tag{20}$$

where λ is a scalar weight.

5. Experiments

This section provides the details of our experimental setup. Moreover, the results of qualitative and quantitative comparison are presented to demonstrate the effectiveness of our method.

5.1. Experimental setup

5.1.1. Dataset

Our proposed method is tested on UP-3D [33] for evaluation. This dataset contains color images and corresponding ground truth 3D pose represented as pose parameters of the SMPL model. The results on the subset of UP-3D selected by Tan et al. [36] to limit the range of global rotation of SMPL models are reported in the present work given that our iterative optimization-based method is sensitive to the initial pose.

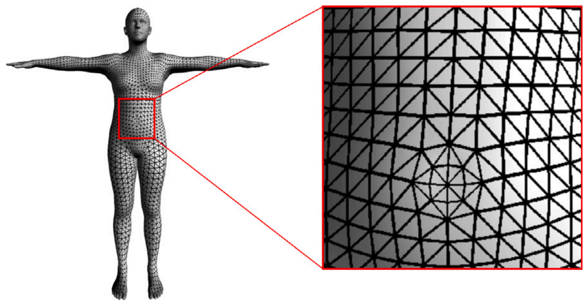

Fig. 8. Detail of SMPL mesh model. The SMPL mesh model is a vertex-based model that accurately represents body shapes by vertices and triangles.

5.1.2. Evaluation metric

For quantitative evaluation, the per-vertex error from [29] is used as a metric to evaluate the accuracy of 3D poses generated by different methods. The surface of body mesh is represented as vertices and triangles in Fig. 8. The accuracy of pose estimation can be effectively evaluated by measuring the error of each vertex, and the per-vertex error E_p can be presented as follows:

$$E_p = \frac{1}{N_v} \sum_{i=1}^{N_v} \|\hat{P}_i - P_i\|_2 \tag{21}$$

where N_v denotes the total number of vertices, $\hat{P}_i$ denotes the estimated location of vertices, and P_i denotes the ground truth location of vertices.

5.1.3. Implementation details

The resolution of the output images of the differentiable renderer is set to 64×64, and the multiple of anti-aliasing F is set to 4. The number of silhouettes N_s is set to 4. Our code is implemented in C++ and CUDA with interfaces to the automatic differentiation library PyTorch [37] to allow the use of their built-in optimizers and easily optimize the pose parameters of SMPL model. The objective function is minimized with Adam optimizer [38] with $\alpha = 1.5 \times 10^{-4}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. λ in Eq. (20) is set to 0.001 across all experiments.

5.2. Qualitative comparison

We compare our differentiable renderer with N3MR [4] by conducting 3D pose estimation in the same experimental setup. We also compare our results with that of direct prediction method named *learning to estimate 3D human pose and shape from a single color image* (L2EPS) by Pavlakos et al. [29] to demonstrate the effectiveness of our approach.

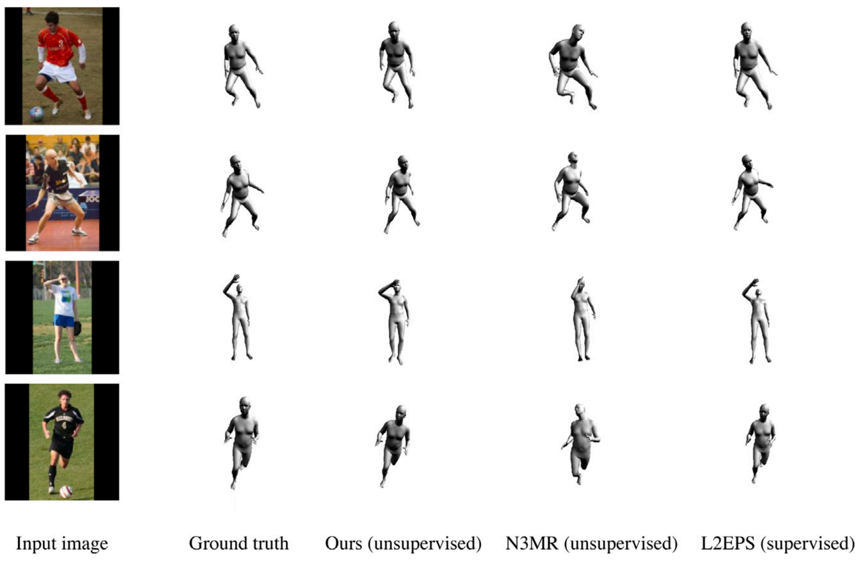

Fig. 9. Visualized results of 3D pose estimation by different methods. From left to right, we show the input images, ground truth, the results obtained by our method, the results obtained by N3MR [4] and results of L2EPS [29].

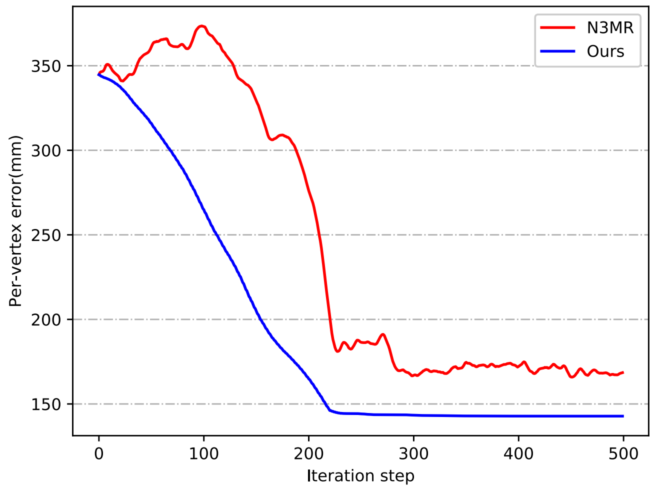

Fig. 10. Per-vertex error curves of our method and N3MR in iterative optimization.

Table 1
Quantitative comparison of accuracy and running time with other prior methods.

Method	Per-vertex error	Runtime	Training time
L2EPS [29](supervised)	117.7 mm	233.5 ms	about 3 days
IDSL [36]	189.0 mm	196.3 ms	about 2 days
N3MR [4](iterative)	172.2 mm	5.4 s	None
Ours(iterative)	142.8 mm	3.8 s	None

Table 2
Quantitative results of different rendering resolution and whether anti-aliasing is applied.

Resolution	Anti-aliasing	Per-vertex error (mm)
32 × 32	No	181.9
32 × 32	Yes	173.7
64 × 64	No	153.4
64 × 64	Yes	142.8

The results in Fig. 9 indicate that the N3MR suffers from local minima, which often results in prediction failure. The optimization process of N3MR is unstable and tends to fall in local minima due to the inconsistency between forward and backward passes. By contrast, the results of our approach appear more appealing and are comparable to that of supervised method [29]. The consistency between forward and backward passes of our differentiable renderer guarantees the optimization stability.

In practice, 3D and 2D joint ground truth are tedious and difficult to obtain. Hence, our method is significant because it facilitates 3D pose estimation without any 2D joint location and 3D ground truth.

5.3. Quantitative comparison

The quantitative evaluation on per-vertex error with different approaches is shown in Table 1. Our differentiable renderer is superior to N3MR [4] in 3D pose estimation; however, it is not as good as L2EPS [29]. The L2EPS method leverages 3D ground truth, whereas our approach only leverages 2D silhouettes and predicts 3D pose in an unsupervised manner. Our method outperforms the *Indirect Deep Structured Learning* (IDSL) for 3D pose estimation by Tan et al. [36], whose differentiable renderer was obtained by training a neural network.

We also report the running time of those methods in Table 1. Although L2EPS and IDSL run significantly faster than optimization-based methods, they require long-time training; thus, they cannot output acceptable results immediately. For a distinct comparison of running time between our method and N3MR, we present the error curves of iterative optimization in Fig. 10. Evidently, our approach converges faster than N3MR, whereas the curve of N3MR tends to oscillate and converge to a larger error than our method.

5.4. Ablation analysis

In this section, we conduct controlled experiments to validate the necessity of different components.

5.4.1. SPT

We conduct an experiment with and without SPT to investigate its influence on 3D pose estimation. We visually compare the results of 3D pose estimation with and without SPT in Fig. 11. These results indicate that the estimated poses without SPT suffers from self-intersection, whereas those with SPT obtains more reasonable poses.

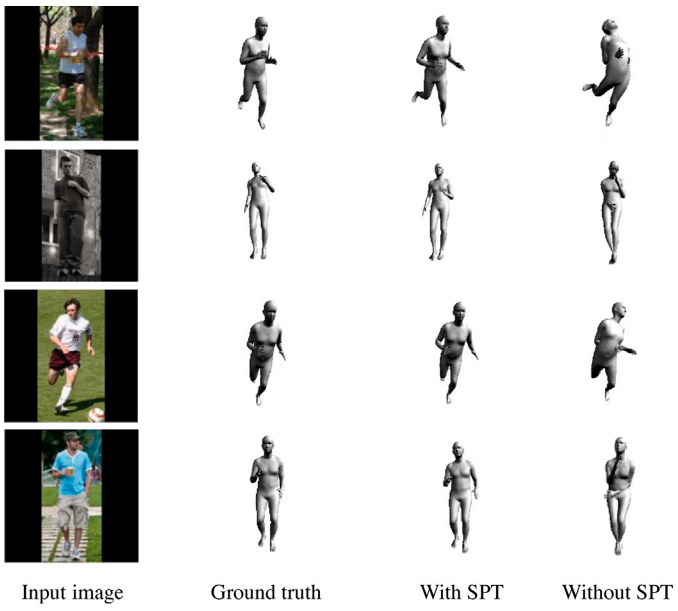

Fig. 11. Results of 3D pose estimation with and without SPT term. From left to right: input image, ground truth, prediction with SPT term and prediction without SPT term.

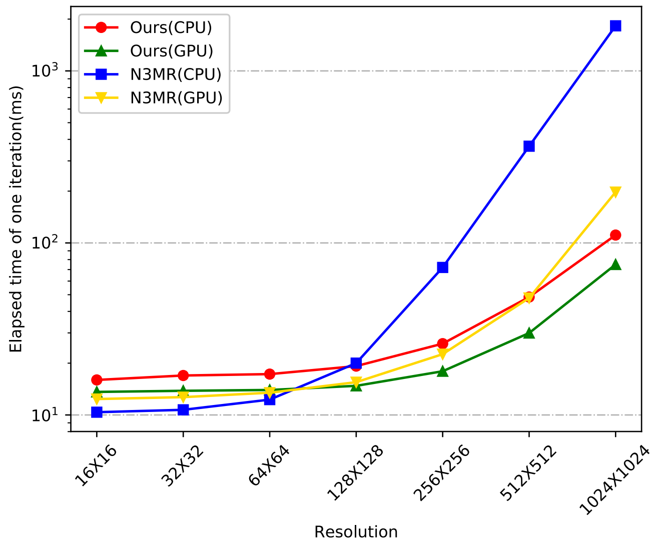

Fig. 12. Elapsed time of one iteration of our method and N3MR under different resolution setups.

5.4.2. *Anti-aliasing*

We conduct a quantitative comparison of 3D pose estimation by differentiable renderers with and without anti-aliasing to reveal the importance of anti-aliasing in the forward pass of our differentiable renderer. The results are shown in Table 2. These results reveal that anti-aliasing can effectively improve the accuracy of 3D pose estimation.

5.5. *Running time analysis*

We conduct experiments using our method with different resolutions and compare it with N3MR [4] to reveal the efficiency of our differentiable renderer. We conduct running time comparison between our method and N3MR on both CPU and GPU to show the efficiency of our method on CPU and GPU. However, N3MR does not have CPU version. For fair comparison, we implement the CPU version of N3MR from their released GPU version. All experiments in this section are performed on a laptop with Intel(R) Core(TM) i7-8750H processor and NVIDIA GeForce GTX 1060 graphic card. We use the Stanford Bunny as input mesh. Table 3 and Fig. 12 present the elapsed time of a single forward and backward passes of the two methods.

Table 3
Elapsed time (ms) of one iteration of our method and N3MR under different resolution setups.

Resolution	Ours(CPU)	Ours(GPU)	N3MR(CPU)	N3MR(GPU)
16 × 16	15.98	13.61	10.38	12.38
32 × 32	16.96	13.82	10.71	12.70
64 × 64	17.28	13.97	12.28	13.47
128 × 128	19.23	14.77	20.01	15.51
256 × 256	25.97	17.95	72.02	22.56
512 × 512	48.97	29.94	365.00	47.85
1024 × 1024	111.10	74.94	1828.69	197.46

In resolution setups lower than 128 × 128, no significant difference is found between our method and N3MR on GPU. In addition, our approach is slightly slower than N3MR on CPU. This is attributed to two reasons, as follows: first, our method requires rendering an image with 16 times higher resolution than that of N3MR, resulting in large computation and long elapsed time in the forward pass. Second, the implementation of N3MR and our method is similar. Both methods traverse all the triangles in the mesh to compute derivatives for each edge. Our method achieves high efficiency by using an analytical method to avoid repeatedly accessing the frame buffers. However, this advantage is insignificant under low resolution setup.

In resolution setup higher than 128 × 128, we observe up to 16X and 2X speedup over N3MR on CPU and GPU respectively. These results demonstrate that our method is more efficient than N3MR under high-resolution setup.

6. Conclusion

We propose a novel method to obtain analytical derivatives for differentiable silhouette renderer. Experiments of 3D pose estimation by silhouette consistency are conducted to show the effectiveness and efficiency of our proposed method. Different from N3MR [4] that uses a hand-design approach to obtain derivatives, our proposed method derives the analytical derivatives on the basis of the continuous definition of pixel intensities. Furthermore, we adopt anti-aliasing to guarantee the consistency between forward and backward passes, thereby achieving optimization accuracy and stability. The efficiency can be improved by skipping irrelevant pixels because we only focus on synthesizing silhouettes. Experimental results indicate that our method has efficiency and accuracy superiority to N3MR [4] in 3D pose estimation.

One of the main limitations of our method is that our algorithm is derived on the basis of the assumption that pixel intensities have only two possible values, and thus only works for silhouette renderer. This characteristic restricts its application in inverse graphics. Another limitation is that our method cannot offer significant speedup under low-resolution setups.

Future direction of this work may include generalizing our method to obtain analytical derivatives with a continuous definition of pixel intensities. Furthermore, extending this approach to obtain analytical derivatives for texture and lighting parameters is interesting.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by the National Natural Science Foundation of China under Grant 61633019, the Science Foundation of Chinese Aerospace Industry under Grant JCKY2018204B053 and the Autonomous Research Project of the State Key Laboratory of Industrial Control Technology, China (Grant No. ICT1917).

References

[1] T. Whitted, An improved illumination model for shaded display, in: ACM Siggraph 2005 Courses, ACM, 2005, p. 4.
[2] M. de La Gorce, D.J. Fleet, N. Paragios, Model-based 3d hand pose estimation from monocular video, IEEE Trans. Pattern Anal. Mach. Intell. 33 (9) (2011) 1793–1805.
[3] M.M. Loper, M.J. Black, OpenDR: An approximate differentiable renderer, in: European Conference on Computer Vision, Springer, 2014, pp. 154–169.
[4] H. Kato, Y. Ushiku, T. Harada, Neural 3d mesh renderer, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3907–3916.
[5] S. Liu, W. Chen, T. Li, H. Li, Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction, 2019, arXiv preprint arXiv:1901.05567.
[6] I. Gkioulekas, A. Levin, T. Zickler, An evaluation of computational imaging techniques for heterogeneous inverse scattering, in: European Conference on Computer Vision, Springer, 2016, pp. 685–701.
[7] V.K. Mansinghka, T.D. Kulkarni, Y.N. Perov, J. Tenenbaum, Approximate bayesian image interpretation using generative probabilistic graphics programs, in: Advances in Neural Information Processing Systems, 2013, pp. 1520–1528.
[8] T.M. Li, M. Aittala, F. Durand, J. Lehtinen, Differentiable monte carlo ray tracing through edge sampling, in: SIGGRAPH Asia 2018 Technical Papers, ACM, 2018, p. 222.
[9] J. Zienkiewicz, A. Davison, S. Leutenegger, Real-time height map fusion using differentiable rendering, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2016, pp. 4280–4287.
[10] G. Liu, D. Ceylan, E. Yumer, J. Yang, J.M. Lien, Material editing using a physically based rendering network, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2261–2269.
[11] E. Richardson, M. Sela, R. Or-El, R. Kimmel, Learning detailed face reconstruction from a single image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1259–1268.
[12] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, C. Theobalt, Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1274–1283.
[13] A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, 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, 2018, pp. 2549–2559.
[14] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, A. Bousseau, Single-image svbrdf capture with a rendering-aware deep network, ACM Trans. Graph. 37 (4) (2018) 128.
[15] A. Kundu, Y. Li, 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, 2018, pp. 3559–3568.
[16] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, W.T. Freeman, Unsupervised training for 3d morphable model regression, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8377–8386.
[17] T.H. Nguyen-Phuoc, C. Li, S. Balaban, Y. Yang, RenderNet: A deep convolutional network for differentiable rendering from 3D shapes, in: Advances in Neural Information Processing Systems, 2018, pp. 7891–7901.
[18] D. Eigen, C. Puhrsch, R. Fergus, Depth map prediction from a single image using a multi-scale deep network, in: Advances in Neural Information Processing Systems, 2014, pp. 2366–2374.
[19] A. Saxena, S.H. Chung, A.Y. Ng, 3-d depth reconstruction from a single still image, Int. J. Comput. Vis. 76 (1) (2008) 53–69.
[20] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, Y.G. Jiang, Pixel2mesh: Generating 3d mesh models from single rgb images, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 52–67.
[21] C.B. Choy, D. Xu, J. Gwak, K. Chen, S. Savarese, 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction, in: European Conference on Computer Vision, Springer, 2016, pp. 628–644.
[22] H. Fan, H. Su, L.J. Guibas, A point set generation network for 3d object reconstruction from a single image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 605–613.
[23] M. Tatarchenko, A. Dosovitskiy, T. Brox, Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2088–2096.
[24] S. Tulsiani, T. Zhou, A.A. Efros, J. Malik, Multi-view supervision for single-view reconstruction via differentiable ray consistency, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2626–2634.
[25] J. Wu, C. Zhang, T. Xue, B. Freeman, J. Tenenbaum, Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling, in: Advances in Neural Information Processing Systems, 2016, pp. 82–90.
[26] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, M.J. Black, SMPL: A skinned multi-person linear model, ACM Trans. Graph. 34 (6) (2015) 248.
[27] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, J. Davis, SCAPE: shape completion and animation of people, ACM Trans. Graph. 24 (3) (2005) 408–416.
[28] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, M.J. Black, Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image, in: European Conference on Computer Vision, Springer, 2016, pp. 561–578.
[29] G. Pavlakos, L. Zhu, X. Zhou, K. Daniilidis, Learning to estimate 3D human pose and shape from a single color image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 459–468.
[30] X. Yan, J. Yang, E. Yumer, Y. Guo, H. Lee, Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision, in: Advances in Neural Information Processing Systems, 2016, pp. 1696–1704.
[31] S. Marschner, P. Shirley, Fundamentals of Computer Graphics, CRC Press, 2015.
[32] Y.D. Liang, B.A. Barsky, A New Concept and Method for Line Clipping, CUMINCAD, 1984.
[33] C. Lassner, J. Romero, M. Kiefel, F. Bogo, M.J. Black, P.V. Gehler, Unite the people: Closing the loop between 3d and 2d human representations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6050–6059.
[34] K.M. Robinette, S. Blackwell, H. Daanen, M. Boehmer, S. Fleming, Civilian American and European Surface Anthropometry Resource (CAESAR), Final Report. Volume 1. Summary, Technical Report, Sytronics Inc Dayton Oh, 2002.
[35] Z. Wu, W. Jiang, H. Luo, L. Cheng, A novel self-intersection penalty term for statistical body shape models and its applications in 3D pose estimation, Appl. Sci. 9 (3) (2019) 400.
[36] V. Tan, I. Budvytis, R. Cipolla, Indirect deep structured learning for 3d human body shape and pose prediction, 2018.
[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.
[38] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.