A Self-occlusion Aware Lighting Model for Real-time Dynamic Reconstruction

Chengwei Zheng, Wenbin Lin, and Feng Xu

Abstract—In real-time dynamic reconstruction, geometry and motion are the major focuses while appearance is not fully explored, leading to the low-quality appearance of the reconstructed surfaces. In this paper, we propose a lightweight lighting model that considers spatially varying lighting conditions caused by self-occlusion. This model estimates per-vertex masks on top of a single Spherical Harmonic (SH) lighting to represent spatially varying lighting conditions without adding too much computation cost. The mask is estimated based on the local geometry of a vertex to model the self-occlusion effect, which is the major reason leading to the spatial variation of lighting. Furthermore, to use this model in dynamic reconstruction, we also improve the motion estimation quality by adding a real-time per-vertex displacement estimation step. Experiments demonstrate that both the reconstructed appearance and the motion are largely improved compared with the current state-of-the-art techniques.

Index Terms—albedo reconstruction, 3D dynamic reconstruction, spatially varying lighting, real-time reconstruction.

1 Introduction

Dynamic reconstruction, aiming to reconstruct the shape, motion, and appearance of the objects in the scene, is an important task in computer vision and graphics. Recently, with the development of depth sensing and parallel computing techniques, this task can be achieved by a single RGB-D sensor in real time [1], [2], [3], which may enable applications like telecommunications, VR/AR content generation, relighting, appearance editing, and so on.

However, the accuracy of the current techniques is not satisfactory due to the low-quality input and the strong assumptions to achieve real-time performance. For example, existing methods [1], [2] use motion graphs to represent the nonrigid surface motions, which cannot model detail motions. Per-vertex displacements could help to handle this; however, the input depth data is quite noisy, and thus strong spatial and temporal regularization is required to estimate robust displacements for the vertices. This leads to a large linear system that is difficult to be solved in real time. Besides the geometry, to solve the appearance of the reconstructed surfaces in real time, a single Spherical Harmonic (SH) is used to model the lighting of all vertices on the surfaces [3]. However, this is not true, especially for complex surface geometries containing self-occlusions. As a consequence, the appearance can not be correctly solved.

To deal with these problems, we propose a novel technique that estimates more detailed surface motions and more accurate surface albedos, still using a single consumer RGB-D sensor and achieving real-time performance. On the motion aspect, we propose a technique that uses the local motions of the previous time step to build the regularization terms in solving the per-vertex displacements of the current time step. Since the motions of vertices in the previous time step are already known, the displacements in this time step are all disentangled in the energy function. And thus, the large linear system is replaced by many small linear systems, which can be solved in parallel with high computation efficiency. On the appearance aspect, we propose a lightweight masking technique to model the spatially varying lighting condition of the reconstructed surface. This method still uses a single SH to model the environment lighting, but each vertex has an implicit mask to model the lighting changes caused by self-occlusions. Combining with a novel optimization framework, which uses the lighting model to construct the data terms by considering the self-occlusion effects, the albedos of all vertices can also be solved in parallel without affecting the real-time performance.

The contribution of this paper lies in three aspects:

- For single view real-time dynamic reconstruction, we propose a system that estimates more accurate surface albedos as well as more detailed surface motions.
- Per-vertex displacements are solved in a very efficient manner by building regularization with the historical data.
- Spatially varying lighting is modeled and used in albedo solving by a masking technique with the ability to model self-occlusions.

2 Related Work

In this paper, we focus on reconstructing the geometry and albedo of an object considering the self-occlusion effects, and we discuss related techniques in this section.

2.1 Dynamic Reconstruction

Many methods have been proposed to reconstruct the geometry and appearance of dynamic objects. High-quality appearance can be generated using physical models [4], [5], [6]; however, these methods usually require a large multi-view setup as well as controlled lighting. Guo et al. [3] proposed a real-time method that took a single-view RGB-D input to reconstruct geometry, surface albedo, non-rigid motion, and low-frequency lighting in real time. This method first uses an optimization-based framework to jointly optimize motion and lighting, then updates the geometry and

• *Chengwei Zheng, Wenbin Lin, and Feng Xu were with school of software and BNRist, Tsinghua University, Beijing, China.
E-mail: zhengcw18@gmail.com, lwb20@mails.tsinghua.edu.cn, xufeng2003@gmail.com*

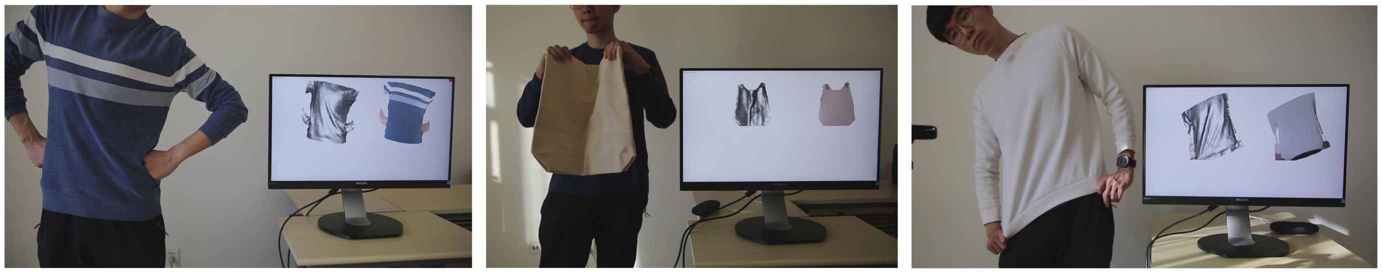


Fig. 1. Live demos of our method for real-time albedo and geometry reconstruction. On the screen, the left shows the current geometry for the currently recorded object pose, and the right shows the corresponding surface albedo.

albedo model by fusing the current color and depth maps into the model. Our method share the same setup as [3], while our method outperforms [3] in both geometry and albedo reconstructions as shown in Section 4.3. Dou et al. [7] presented a 360 performance capture system that enabled the real-time reconstruction of non-rigid scenes. Du et al. [8] proposed a solution toward real-time seamless texture montage build by leveraging geodesics-guided diffusion and temporal texture fields. Guo et al. [9] presented a volumetric capture system for photo-realistic and high-quality relightable full-body performance capture. Zheng and Xu [10] reconstructed the high-quality dynamic texture of general dynamic objects. Other methods also achieved reconstruction with dynamic texture [11], [12], [13]. Besides, some works focused on geometry and reflectance reconstruction of human faces based on parametric models [6], [14], [15], [16].

Recently, deep learning techniques provide new opportunities to capture the appearance of dynamic objects, especially for human faces [17], [18], [19], [20], [21] and human bodies [22], [23]. For instance, Saito et al. [17] represented fine-scale texture details of human faces based on mid-layer feature correlations from a convolutional neural network. Martin-Brualla et al. [22] produced high-resolution and high-quality images of the human bodies using a deep architecture in real time. Pandey et al. [23] proposed an end-to-end framework to synthesize renderings of humans in free-viewpoints using a single RGB-D camera. Although deep learning reconstruction techniques are able to generate high-quality results, they are restricted to specific objects, and a wide variety of objects remains unexplored. Some networks are also too heavy to run in real time. Our proposed method aims to reconstruct the detailed geometries and high-quality albedos of general dynamic objects using only a single RGB-D camera and in real time, which can be a difficult task for the methods above.

2.2 Intrinsic Decomposition

To decompose a color image into reflectance and shading, optimization-based methods usually build energy terms with assumptions and priors on reflectance [24], [25], [26]. Methods based on the Retinex algorithm dealt with this problem by assuming that large image gradients and small gradients were respectively caused by reflectance and shading [27], [28]. Barron and Malik [29], [30] recovered shape, albedo, and illumination from a single image using a combination of priors. Bi et al. [31] introduced an image transform based on the L_1 norm for piece-wise image flattening and further for complex scene-level intrinsic image decomposition. Cheng et al. [32] regularized intrinsic decomposition with the aid of near-infrared imagery and proposed priors.

In addition to RGB images, Some methods also made use of depth maps from an RGB-D camera to decompose the color images [33], [34], [35], [36]. Jeon et al. [34] combined a texture-aware image model and a surface normal based constraint from an RGB-D image to improve the results. Hachama et al. [35] reconstructed the surface from a single or multiple RGB-D images of a static scene with a data term, which was related to the image formation process and expressed the relation between different components, and a regularity term, which contained an efficient combination of two regularizers on the illumination vector field and albedo. Wei et al. [36] further took advantage of physical principles from inverse rendering and achieved high accuracy with real-time performance. Besides, intrinsic video decomposition methods were able to separate a video stream into reflectance and shading layers [37], [38], [39], [40].

Based on provided data sets [27], [41], [42], [43], [44], [45], some deep learning intrinsic image decomposition methods were proposed to estimate reflectance [46], [47], [48], [49], [50]. Cheng et al. [51] treated image decomposition as an image-to-image transformation problem and developed a multi-channel architecture that learned the transformation function in successive frequency bands in parallel. Liu et al. [52] proposed an unsupervised intrinsic image decomposition framework, which directly learned the latent feature of reflectance and shading from unsupervised and uncorrelated data. As reflectance is related to the surface normal, several methods also estimated surface normal to improve the performance. Sengupta et al. [53] presented an end-to-end learning framework for producing an accurate decomposition of an unconstrained human face image into shape, reflectance, and illuminance. Kanamori and Endo [54] inferred albedo, shape, and illumination from a human portrait with light occlusion. Yu and Smith [55] trained a fully convolutional neural network to regress albedo and normal maps from a single image. Luo et al. [56] proposed a novel learning-based framework that adapted surface normal knowledge to decompose a natural image into a reflectance image and a shading image. Many intrinsic decomposition methods assume piece-wise constant reflectance [39], [42], [45], [52] and we also use this assumption in our method. However, some challenging scenes, such as wrinkles, are still difficult to handle. In contrast, our system takes an RGB-D sequence as input and can output high-quality albedo even in wrinkle regions with the help of geometry reconstruction and self-occlusion model.

2.3 Shape-from-shading and lighting estimation

Intrinsic decomposition methods majorly aim to recover the albedo and the shading components, while there are also some methods that additionally focus on recovering the shape or estimating the environment lighting from inputs. Shape-from-shading

(SfS) problem [57] focuses on estimating the shape, especially the surface normal from a single image. As shading is strongly related to lighting, many of these methods also estimate the environment lighting to improve the results. Barron and Malik [58] proposed to apply a mixture of shapes and a mixture of several illuminations that were embedded in a “soft” segmentation of the input RGB-D image. Yu et al. [59] made use of a noisy depth map from an RGB-D camera to resolve ambiguities in shape-from-shading, by using edges from the RGB image to guide a hole filling process and create a reliable depth map proxy. In [60], Wu et al. presented the first real-time method for refinement of depth data using shape-from-shading. Han et al. [61] applied a shape-from-shading approach with a general lighting model to estimate detailed shape from a single RGB-D image. In their lighting model, a local lighting parameter for each pixel was multiplied with global lighting and was solved with the help of smoothness terms and a uniform albedo assumption. Some methods also use an RGB-D sequence as input for geometry and albedo recovery in a static scene [62], [63]. Spatially-varying spherical harmonics were used in [63] by partitioning the object volume into subvolumes and estimating SH coefficients for each subvolume. Spatially varying lighting of indoor scenes can also be recovered from a single RGB-D image [64], or a single RGB image [65]. The spatially varying lighting was modeled by light sources and indirect illumination in [64], and an environment map for each pixel in [65], which made these methods time-consuming. Yu et al. [66] further proposed a self-supervised approach to decompose an outdoor image into its albedo, geometry, and illumination, then to achieve relighting. However, these methods take a single image or a sequence of a static scene as input and do not take dynamic objects into account. Unlike previous works [59], [60], [62], [66] that only consider a single global environment lighting, we use different masks on a global environment lighting for different surface points to model the spatially varying lighting accurately and keep real-time performance.

2.4 Ambient Occlusion

Ambient occlusion (AO) [67] is a shading method that takes the light occluded by geometries into account. Ambient occlusion was introduced by Zhukov et al. [68] and yields the percentage of light blocked by the geometry close to a surface point. Since then, many algorithms for computing ambient occlusion have been proposed [69], [70], [71], [72]. Kontkanen and Laine [73] presented a real-time technique for computing inter-object ambient occlusion. Bavoil et al. [74] proposed a real-time ambient occlusion computation based on a depth image from the eye’s point of view. Díaz et al. [75] presented two methods for the fast generation of ambient occlusion on volumetric models. Laine and Karras. [76] proposed efficient methods for calculating ambient occlusion so that the results could match those produced by a ray tracer. Hauagge et al. [77] presented a method for computing ambient occlusion for a stack of images of a scene from a fixed viewpoint and further used it for intrinsic image decomposition. Although both AO and our self-occlusion model take the occlusion into account, our method is a different method from AO. AO methods provide the percentage of light that is blocked by the geometry, which will not change under different environment lightings. In contrast, our method calculates the percentage of the blocked light according to the environment lighting and can obtain more accurate results, which will be detailed in Section 4.3.

3 Method

Our method runs frame-by-frame in real time to reconstruct the geometry, non-rigid motions, environment lighting, and surface albedo of the object. For each input frame, we first solve for non-rigid surface motion and update the base geometry similarly to [3]. The base geometry is in the pose of the first frame, called the canonical frame. The non-rigid motions are driven by nodes that are normally distributed on the surface. Based on the reconstructed 3D mesh, we run geometry detail fitting to get the displacement of each vertex (Section 3.3). Then the environment lighting will be solved (Section 3.1) and the surface albedo will be updated (Section 3.4) using our proposed self-occlusion model (Section 3.2). The pipeline is demonstrated in Fig. 2.

3.1 Lighting

For a surface point with surface normal $\mathbf{n}$, its appearance color is determined by its albedo and received irradiance $i(\mathbf{n})$. Without considering self-occlusion, irradiance $i(\mathbf{n})$ can be calculated by an integral of incoming radiance $L(\omega)$ over the upper hemisphere $\Omega(\mathbf{n})$, as in [78]:

$$i(\mathbf{n}) = \int_{\Omega(\mathbf{n})} L(\omega)(\mathbf{n}\cdot\omega)d\omega. \tag{1}$$

We use a cubemap to represent the incoming radiance environment map, and each pixel in the cubemap corresponds to a light source. Thus (1) can be formulated as:

$$i(\mathbf{n}) = \sum_{\omega\in\Omega(\mathbf{n})} L(\omega)(\mathbf{n}\cdot\omega)\delta(\omega), \tag{2}$$

where $\delta(\omega)$ denotes the solid angle for the light source ω. The radiance of each light source is represented by spherical harmonics (SH) as follows:

$$L(\omega) = \sum_{l,m} L_{l,m}H_{l,m}(\omega). \tag{3}$$

Here, $\{H_{l,m}\}$ represent the SH basis functions and $\{L_{l,m}\}$ are the SH coefficients that define the environment lighting. Up to second-order SH basis functions are used with $0 \leqslant l \leqslant 2$, and $-l \leqslant m \leqslant l$. We assume that the environment lightings are low-frequency and can be represented by up to the second order of spherical harmonics.

Based on the lighting representation above, we reconstruct the environment lighting defined by 9 SH coefficients $\{L_{l,m}\}$ by minimizing the following energy function containing *multi-frame* data terms:

$$E_{light} = \sum_{t\in T}\sum_{s\in S(t)} \left\|a_s \cdot i(\mathbf{n}_s^t) - c_s^t\right\|^2, \tag{4}$$

where a_s is the albedo of surface point s, $\mathbf{n}_s^t$ is its normal in frame t, and c_s^t is the input pixel color in frame t that corresponds to surface point s. S and T are the sets of surface points and frames, respectively. Different from [3], we use *multi-frame* data terms, which means we also construct data terms from previous frames, not only the current frame. Multi-frame data terms are commonly used in offline reconstruction methods [62], [63], while other real-time methods usually only focus on the current frame. Here we apply multi-frame data terms in an incremental mode to make these terms suitable for real-time reconstruction as follows.

To construct data terms from the previous frames, we need to record some information of these frames first. Note that according

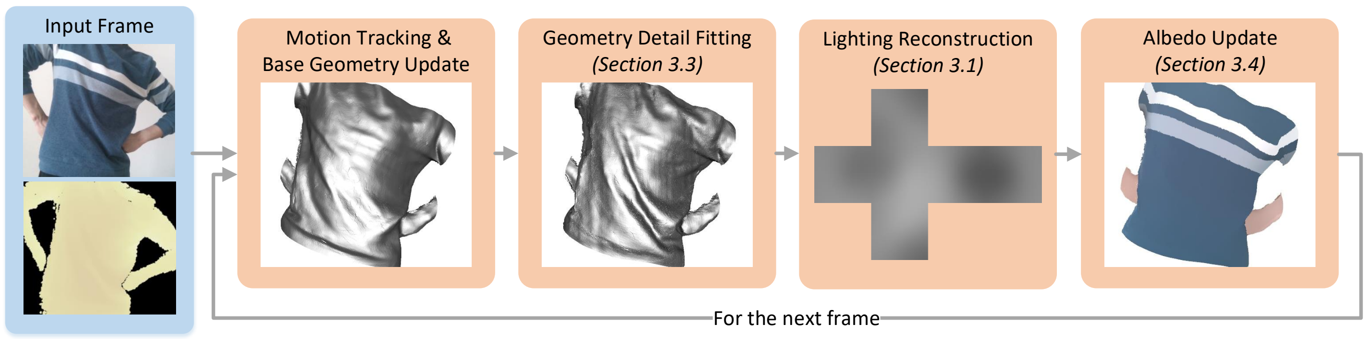

Input Frame
Motion Tracking & Base Geometry Update
Geometry Detail Fitting (Section 3.3)
Lighting Reconstruction (Section 3.1)
Albedo Update (Section 3.4)
For the next frame

Fig. 2. The pipeline of our framework. Using input RGB-D images, our method first solves for non-rigid surface motion and updates the base geometry, and geometry detail displacements are obtained by fitting the input depth image. Then the environment lighting is solved with multi-frame data terms, and the albedo is updated using our self-occlusion model. Some of the reconstruction results and the intermediate results will be used for the next frame.

to (2) and (3), $i(\mathbf{n}_s^t)$ is a linear combination of 9 SH coefficients $\{L_{l,m}\}$:

$$i(\mathbf{n}_s^t) = \sum_{l,m} b_{l,m}(s,t) L_{l,m}. \tag{5}$$

Here $\{b_{l,m}(s,t)\}$ are the coefficients of this linear combination and model how each SH coefficient contributes to the received irradiance. For each pixel in the current frame, we record the corresponding 3D canonical position, the input pixel color, and the vector b, and these will be used later. When constructing a previous data term in (4), the recorded 3D canonical position will be used to find the current albedo of the surface point, while the input pixel color and the vector b will be used directly. We apply a queue with a fixed maximum size for storing the recorded data above. We further improve the lighting reconstruction with the following self-occlusion model.

3.2 Self-occlusion Model

Taking self-occlusion into account, a new visibility term $V(\omega)$ should be added into (2) as:

$$I(\mathbf{n}) = \sum_{\omega \in \Omega(\mathbf{n})} L(\omega)(\mathbf{n} \cdot \omega) V(\omega) \delta(\omega). \tag{6}$$

Here we use $I(\mathbf{n})$ instead of $i(\mathbf{n})$ in (2) to identify that the incoming radiance is computed using our self-occlusion model, and also in the following equations. The environment lighting $L(\omega)$ is represented by spherical harmonics solved in Section 3.1, and then is converted into cubemap. The value of $V(\omega)$ is 1 if the surface point is visible in the direction of light source ω (i.e., the light is not occluded); otherwise, its value turns to 0. Our novel self-occlusion model is able to calculate whether the light from light source ω to a surface point A is occluded by its neighboring surface points or not.

First, as only the light in the upper hemisphere can reach the surface point A, we need to find out the neighboring surface points in the upper hemisphere as candidate neighbors. We traverse all the vertices in a certain region to judge if it is a candidate neighbor by:

$$\overrightarrow{AB} \cdot \mathbf{n}_A > 0, \tag{7}$$

where B is a neighboring vertex. An index voxel volume is built for searching the neighboring vertices, in which each voxel stores the inside vertex indices. Thus the neighboring vertices can be obtained by searching the neighboring voxels.

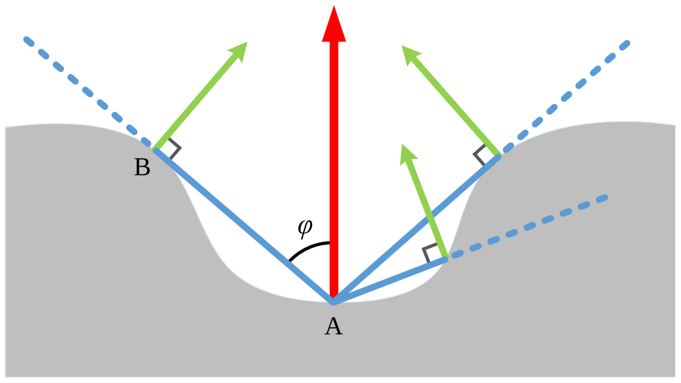


Fig. 3. Our self-occlusion model. Red for the surface normal; green for the *link-normal*.

We then propose a vector named *link-normal*, to help calculate self-occlusion. For each candidate neighbor, a link-normal can be calculated as follows:

$$\mathbf{m} = \mathbf{n}_A - \left(\frac{\overrightarrow{AB}}{|\overrightarrow{AB}|} \cdot \mathbf{n}_A \right) \frac{\overrightarrow{AB}}{|\overrightarrow{AB}|}, \tag{8}$$

$$\mathbf{M} = \frac{\mathbf{m}}{|\mathbf{m}|}, \tag{9}$$

where $\mathbf{M}$ is the link-normal corresponding to candidate neighbor B, while $\mathbf{m}$ is the link-normal before normalization. According to (8) and shown in Fig. 3, $\mathbf{m}$ is actually the difference between $\mathbf{n}_A$ and the projection of $\mathbf{n}_A$ on $\overrightarrow{AB}$. Thus $\mathbf{m}$ is in the direction of the perpendicular line of $\mathbf{n}_A$ on $\overrightarrow{AB}$.

We can obtain one link-normal from each candidate neighbor of the surface point A, and these link-normals compose the link-normal set of A, denoted as $\mathcal{M}_A$.

Light source ω is not occluded if and only if all the angles between ω and link-normals are acute angles, formulated as:

$$\forall \mathbf{M} \in \mathcal{M}_A, \mathbf{M} \cdot \omega > 0. \tag{10}$$

That is, the direction of light source ω is in the intersection of all hemispheres centered on link-normals. Otherwise, the light source ω will be masked by setting $V(\omega)$ in (6) to zero.

Acceleration. We further apply a pruning algorithm to speed up the calculation in (10). According to (10), we may need to traverse all the link-normals in $\mathcal{M}_A$ for each light source to determine whether it is occluded or not, which is a time-consuming step. Instead, when building up link-normal set $\mathcal{M}_A$, we calculate each angle φ between each connecting line (e.g., $\overrightarrow{AB}$ for candidate neighbor B as shown in Fig. 3) and the surface normal $\mathbf{n}_A$, and record the smallest angle as φ_{min}. Then if the angle between a light source ω and the surface normal $\mathbf{n}_A$ is smaller than φ_{min}, this light source will be directly set as visible (not occluded) without traversing all link-normals. This is because that the light source is in the intersection of all hemispheres centered on link-normals. This pruning algorithm contributes a lot to our real-time performance, as many of the light sources in the upper hemisphere are not occluded and will be set as visible directly.

3.3 Geometry Detail Fitting

The base non-rigid motions are driven by nodes in the motion graph as in [3]. To model detail non-rigid motions, we add a displacement to each vertex after non-rigid motion driven by nodes to fit the input depth image. The displacement d added to a vertex is obtained by minimizing the following energy function consisting of a data term and a regularization term:

$$E_{geo}(d) = E_{fit}(d) + w_{reg}E_{reg}(d), \tag{11}$$

where w_{reg} is a predefined weight of E_{reg}.

The data term E_{fit} is computed using the input depth image:

$$E_{fit}(d) = \left(D_{base} + d - D_{in}\right)^2. \tag{12}$$

Here D_{base} is the distance from the vertex after the base non-rigid motion driven by nodes to the camera, while D_{in} is the input depth value corresponding to the vertex. The added displacement is in the viewing direction.

And the regularization term E_{reg} is formulated as:

$$E_{reg}(d) = \left(d - \frac{1}{\|N\|}\sum_{j\in N} d_j^{(t-1)}\right)^2, \tag{13}$$

where N indicates the set of the neighborhood. The regularization term constrains the displacement d to be close to the average of the neighboring displacements in the previous frame, contributing to smooth results.

As the topology of the base mesh may change during the geometry updating process, we use a voxel volume in the canonical frame to store the average displacement of vertices in each voxel in the previous frame. The displacements of the neighbors in N in the previous frame can be obtained directly from this volume, with each neighbor corresponding to a voxel. Then the volume will get updated for the next frame.

With these designs, the proposed detail fitting method can calculate the displacement of each vertex independently and fully in parallel, which helps to obtain high-quality geometry in real time without solving large linear problems. Also, this method can be easily integrated into other real-time systems but still keep the real-time performance.

It should be noted that the data term in (11) to fit the input depth is effective only when the vertex corresponds to a valid depth value in the input depth map. If a vertex corresponds to a pixel with no depth data, its displacement is computed only using the regularization term.

3.4 Albedo Update

The last step for each frame is to update the albedo. The albedo for each vertex x is updated by minimizing the following energy function:

$$E_{albedo}(a_x) = E_{data} + w_tE_t + w_sE_s, \tag{14}$$

which contains three terms: data term, time consistency term, and spatial smooth term. w_t and w_s are the weights of E_t and E_s, respectively.

The first term E_{data} is a data term based on the refined geometry, and is defined as:

$$E_{data}(a_x) = \left\|a_x \cdot I(\mathbf{n}_x) - C(x)\right\|^2. \tag{15}$$

Here $I(\mathbf{n}_x)$ is the incoming irradiance obtained using our self-occlusion model in Section 3.2, and $C(x)$ is the corresponding pixel color in the input color image.

The second term E_t is to keep the time consistency of the albedo to the previous frame:

$$E_t(a_x) = \left\|a_x - a_x^{(t-1)}\right\|^2. \tag{16}$$

And its weight w_t is a predefined parameter.

The spatial smooth term E_s is formulated similarly to [79], based on the assumption of piece-wise constant reflectance:

$$E_s(a_x) = \sum_{j\in N}\phi(\Gamma(x) - \Gamma(j))\left\|a_x - a_j^{(t-1)}\right\|^2, \tag{17}$$

where N indicates the set of neighborhood. Different from [79], we use a certain region instead of the one-ring neighborhood for a larger search range. $\Gamma(x)$ indicates the chromaticity of the vertex x, and is computed by $\Gamma(x) = C(x)/G(x)$, where C is the color and G is the intensity. The spatial smooth term weight w_s is a predefined coefficient.

Similarly to geometry detail fitting, we use an albedo volume to store the solved albedo, and each voxel saves the average albedo of the inside vertices. The previous albedo values in (16) and (17) can be found from this volume, and each neighbor in N corresponds to a voxel. The albedo of each vertex is also calculated independently and in parallel, contributing to our system's real-time performance.

4 Experiments

In this section, we first present the performance and the parameter settings of our system. Then, we evaluate several parts of our pipeline. Besides, we present our results on various dynamic objects, also with qualitative and quantitative comparisons with other methods. Sequence results can be found in our accompanying video.

4.1 Performance and Parameters

Our system is implemented on a computer with a 3.40-GHz four-core CPU, 16 GB RAM, and an NVIDIA GTX GeForce 2080Ti graphics card. We use Intel RealSense SR300 to record RGB-D sequences with the resolution of 1280×720 at 30 fps. Our pipeline runs at 34 ms for each frame, and the detailed running time of each process can be found in Table 1. Parallel computing is implemented using CUDA on the graphics card. Note that because the environment lighting is usually not changed, we run lighting reconstruction every two frames, which takes 6 ms, and on average it takes 3 ms for each frame.

TABLE 1
Running time of our system.

Process	Running time
Motion tracking	18 ms
Base geometry update	6 ms
Geometry detail fitting	1 ms
Lighting reconstruction	3 ms
Albedo update	4 ms
Everything else	2 ms
Total	34 ms

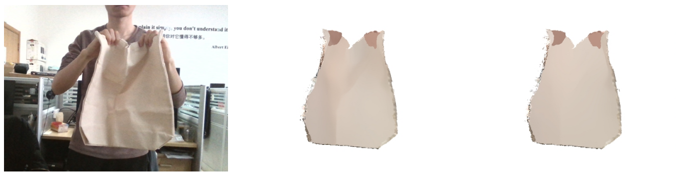


Fig. 4. Evaluation of self-occlusion model. Left: input color image; middle: reconstructed albedo without self-occlusion model; right: our albedo.

We use a cubemap with 8×8 light sources on each face (totally 384 light sources) to represent the environment lighting. The voxel volume resolution is $320 \times 320 \times 320$. The search radius for the candidate neighbor in our self-occlusion model is set to around 8 cm. For geometry reconstruction, we set w_{reg} in (11) to 2 and the radius of neighborhood in (13) to 1 cm. For albedo update, we set w_t and w_s in (14) to 30 and 10 respectively, and the radius of neighborhood in (17) to 2 cm.

4.2 Evaluations

Evaluation of self-occlusion model. We propose a novel self-occlusion model to calculate the incident light by taking the occlusion of neighboring surface points into account. We run our method with and without the self-occlusion model, and the results are demonstrated in Fig. 4. Without our self-occlusion model, the surfaces with significant self-occlusions will get poor results.

Evaluation of geometry detail fitting. To improve the reconstructed geometry and albedo, we add a displacement to each vertex on the mesh. Since we get self-occlusion information from geometry and surface normal is important in albedo update, high-quality albedo also benefits from accurate geometry. We compare the results using geometry displacements and the results without,

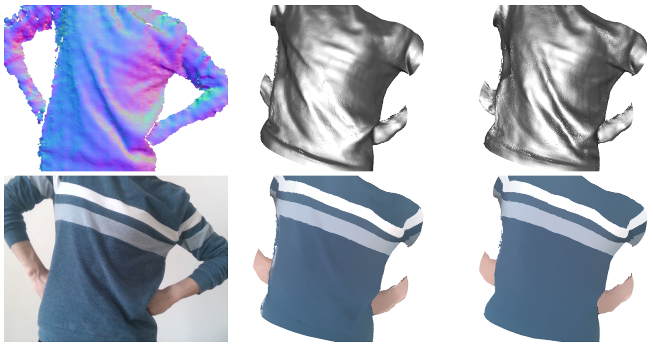


Fig. 5. Evaluation of geometry detail fitting. Left: normal map from depth input and input color image; middle: reconstructed geometry and albedo without geometry detail fitting; right: our geometry and albedo.

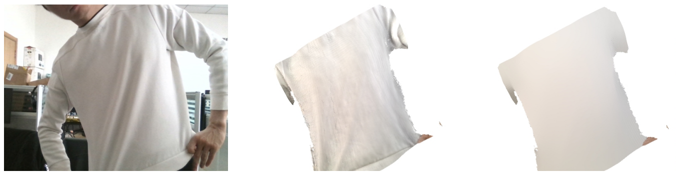


Fig. 6. Evaluation of albedo spatial smooth term. Left: input color image; middle: reconstructed albedo without albedo spatial smooth term; right: our albedo.

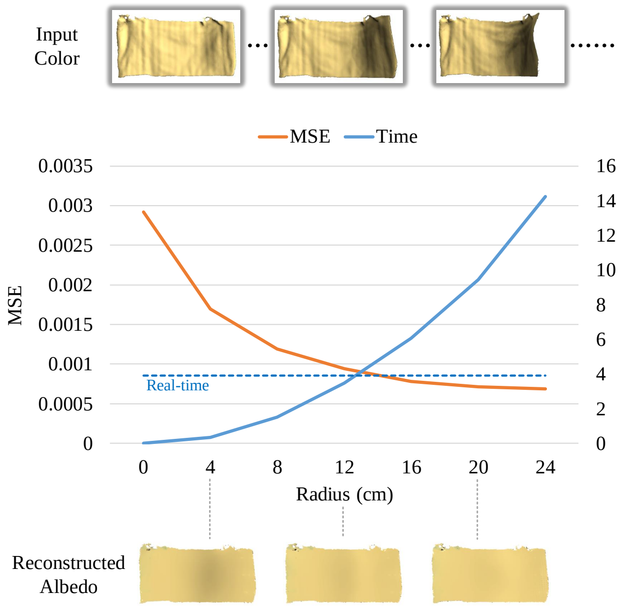


Fig. 7. Ablation study on the search radius of the candidate neighbor for self-occlusion. The length of the towel is 75 cm, which leads to long-range self-occlusion. The dotted line identifies the time limit for real-time performance.

as shown in Fig. 5. We can see that geometry detail fitting improves geometry and further contributes to better albedo.

Evaluation of albedo spatial smooth term. In our albedo update method, we add a spatial smooth term in the energy function (14). This term forces the albedo of a surface point to be close to the albedos of the neighboring points in a certain region, based on chromaticity differences. We remove this term, and the reconstructed results can be found in Fig. 6. The results show that this term helps us get smooth albedo despite input noises and reconstruction inaccuracies.

Ablation study on the search radius of self-occlusion. In our self-occlusion model, a major parameter is the search radius for the candidate neighbor. We select different values of this radius and run our method using a challenging input sequence with long-range self-occlusion. In order to better evaluate different parameters, we use synthetic data generated by an offline render of which the ground truth can be obtained. To synthesize this sequence, we first capture a sequence of real data with long-range self-occlusion. Then we reconstruct the geometry and motions using our method, and set the albedo and environment lighting to given ones. Finally, we use an offline render to render the object in the original camera views and get the synthetic color images. When running using the synthetic data, the original depth images from the captured real data and the synthetic color images are inputted. The environment lighting is initialized to be the

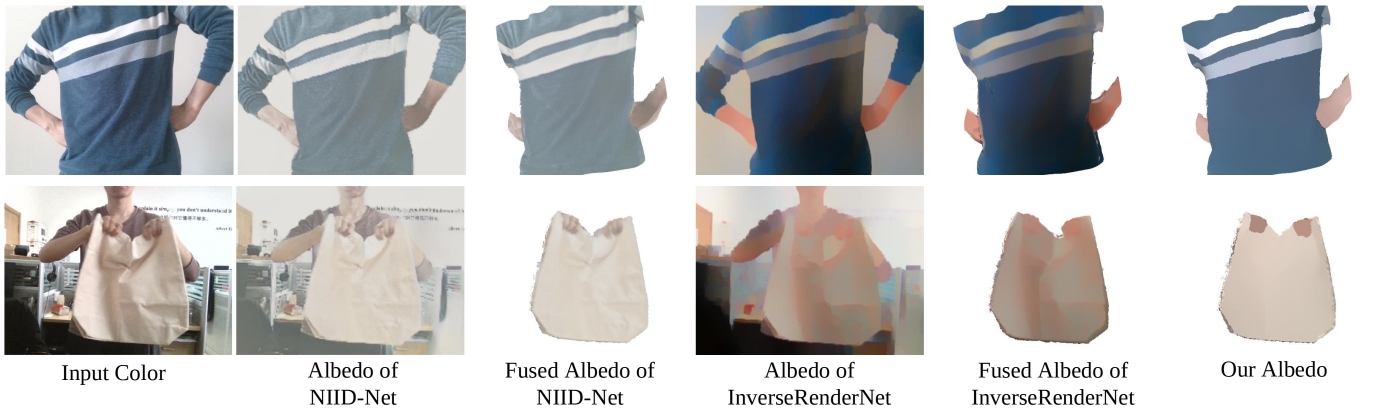


Fig. 8. Comparison with intrinsic fusions using NIID-Net [56] and InverseRenderNet [55]. We show both the direct output albedo and the fused albedo in data sequences of the two compared methods.

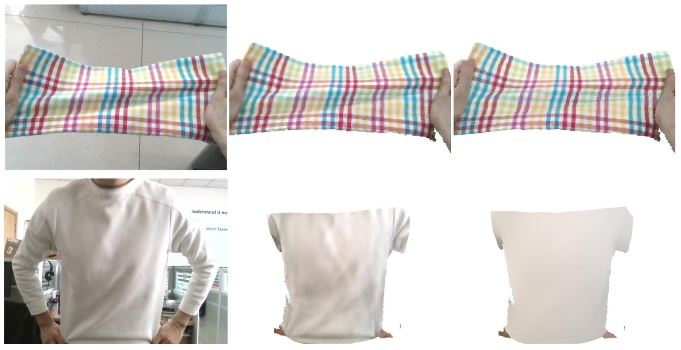


Fig. 9. Comparison with [3]. Left: input color images; middle: reconstructed albedo of [3]; right: our albedo.

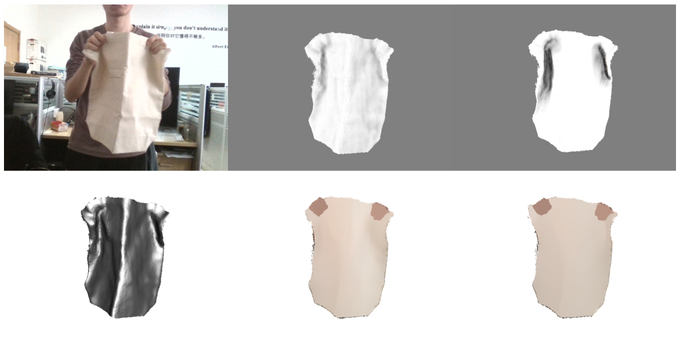


Fig. 10. Comparison with AO reconstruction. Left: input color image and reconstructed geometry; middle: percentage of light not occluded and reconstructed albedo in AO reconstruction; right: our percentage of light not occluded and our albedo. (Gray scale [0, 255] corresponds to percentage [0, 100%])

ground truth. In Fig. 7, we show the standard mean-squared errors (MSE), running times, and reconstructed albedos on different radius parameters, as well as several input frames. From Fig. 7, we can see that increasing the radius will lead to better reconstruction results but will take more time costs.

4.3 Results and Comparisons

We show our reconstructed results on different objects in Fig. 15, as well as in our accompanying video. Our method is able to reconstruct different objects, such as garments, clothes, toys, paper, cushions, and bags. Besides, multiple objects can also be handled. The live demos running in real time can also be found in Fig. 1 and in our video.

Comparison with [3]. We compare our method with [3], which reconstructs the geometry, albedo, non-rigid motions, and lighting in real time. As shown in Fig. 9, our method outperforms the method in [3], especially in the areas with wrinkles, where the self-occlusions are obvious.

Comparison with intrinsic fusions. Based on a state-of-the-art intrinsic decomposition method NIID-Net [56], we fuse the output albedo map of each frame into a model. The geometry and motions of the model are reconstructed by [3]. We also compare with a self-supervision method InverseRenderNet [55] in the same way. Both intrinsic decomposition methods are implemented by the original authors and use the models trained by the original authors. The albedo maps and the fused results are shown in Fig. 8. From the results, we can see that our method is able to reconstruct high-quality albedo, whereas the compared methods can not fully eliminate the effects of wrinkles. There are also some color distortions in the output albedos of NIID-Net [56] and InverseRenderNet [55], such as lower color saturation, which can lead to great RGB errors. Besides, the method in NIID-Net [56] and InverseRenderNet [55] can not reach real-time performance.

Comparison with ambient occlusion (AO) reconstruction. Ambient occlusion is widely used in computer graphics, which takes the geometry as input and approximates the percentage of light reaching a point based on occlusion. Although both AO and our method take the occlusion into account, our method is a different method from AO. AO methods provide the percentage of light that is blocked by the geometry, which will not change according to environment lightings. In contrast, our method masks the light sources that are occluded, so the percentage of the blocked light changes under different environment lightings. We replace our self-occlusion model with the image-space horizon-based ambient occlusion (HBAO) method [74] and run our reconstruction pipeline, and the results are shown in Fig. 10. We also show the percentage of light not occluded in Fig. 10, which is the direct output of HBAO [74] for the AO reconstruction; while for our method, we compute the ratio of the received light with and without our self-occlusion model as this percentage. From the results, we can see that our self-occlusion model can provide more accurate lighting and albedo than the reconstruction using AO,

TABLE 2
MSE, LMSE, DSSIM, average RGB errors, and variances on *synthetic* data with 840 frames. Lower is better for all metrics.

Method	MSE	LMSE	DSSIM	Error	Variance
Guo *et al.* [3]	0.00528	0.00501	0.0262	20.24	39.28
Fusion of NIID-Net [56]	0.00653	0.00551	0.0216	44.23	6.93
Fusion of InverseRenderNet [55]	0.01608	0.00984	0.0329	57.76	390.13
AO reconstruction	0.00247	0.00215	0.0143	12.56	9.07
Ours w/o self-occ	0.00342	0.00284	0.0162	19.79	12.66
Ours w/o geo detail	0.00314	0.00306	0.0186	10.86	4.45
Ours w/o albedo smooth	0.00310	0.00307	0.0240	9.76	13.27
Ours final	**0.00210**	**0.00194**	**0.0142**	**8.31**	**2.42**

TABLE 3
WHDR and two regions' variances on *real* data with 840 frames. Lower is better for all metrics.

Method	WHDR(%)	Var. 1	Var. 2
Guo *et al.* [3]	16.32	70.41	100.28
Fusion of NIID-Net [56]	14.34	7.62	7.37
Fusion of InverseRenderNet [55]	29.92	73.27	270.86
AO reconstruction	8.78	8.84	12.25
Ours w/o self-occ	9.70	11.79	19.76
Ours w/o geo detail	8.18	6.03	8.25
Ours w/o albedo smooth	12.19	24.76	44.07
Ours final	**5.30**	**4.69**	**5.99**

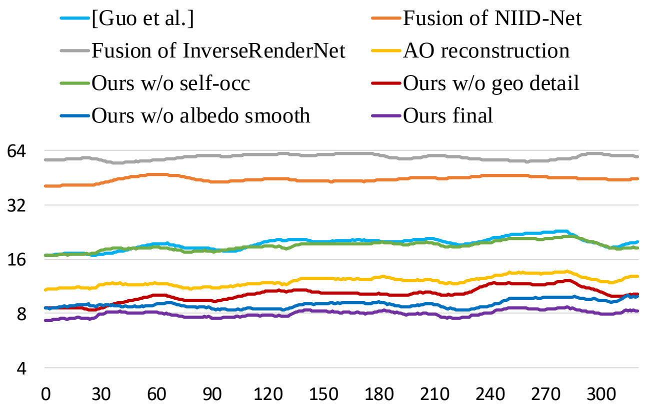


Fig. 11. RGB errors on a synthetic data sequence. Vertical logarithmic coordinates for the errors and horizontal coordinates for the frame indices. (320 frames shown)

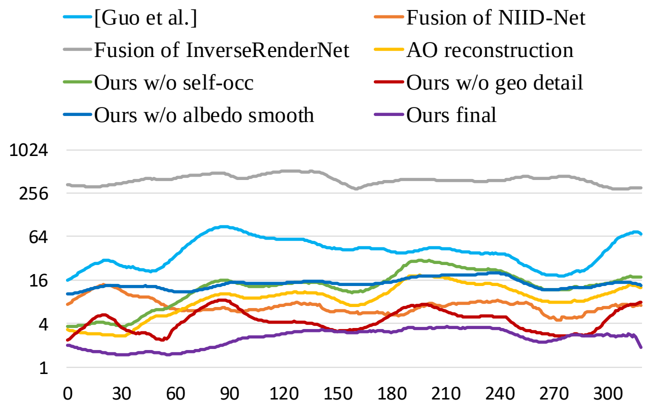


Fig. 12. Variances on a synthetic data sequence. Vertical logarithmic coordinates for the variances and horizontal coordinates for the frame indices. (320 frames shown)

which uses fixed percentages that do not change under different lightings, especially in the reconstruction under strong directional light. More comparisons can be found in Fig. 16 and in our accompanying video.

We also performed quantitative comparisons of reconstructed albedo on both synthetic data and real data. The synthetic data is generated using an offline render. For synthetic data with dense ground truth, following previous works [33], [47], we report the standard mean-squared error (MSE), local mean-squared error (LMSE), and dissimilarity version of the structural similarity index measure (DSSIM). We also calculate the average RGB differences from the ground truth albedo as well as albedo variances on the regions with the same ground truth albedo, and the results can be found in Table 2. These results demonstrate that our method outperforms other methods in all metrics. The average RGB errors are computed by

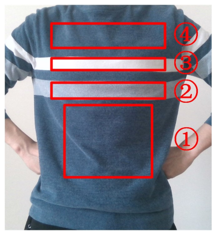
$$\frac{1}{\|P\|}\sum_{x\in P}\|rgb(x)-rgb'(x)\|_2^2, \tag{18}$$

where x is a pixel and P is the set of pixels corresponding to the foreground. Function rgb and rgb' return the three-channel RGB values of the reconstructed albedo and the ground truth albedo, respectively. The variances are computed by

$$\frac{1}{\|Q\|}\sum_{x\in Q}\left\|G(x)-\overline{G}\right\|^2, \tag{19}$$

where Q is the set of pixels in the selected region. G returns the intensity, and $\overline{G}$ is the average of intensity in Q. We also demonstrate the curves of errors and variances of a sequence in Fig. 11 and Fig. 12 with logarithmic coordinates. The synthetic data is generated in the same way as the one in Sec. 4.2. An example of the synthetic sequence is shown in the last row of Fig. 16. It should be noted that there is scale ambiguity between lighting and albedo in intrinsic fusion methods when computing the albedo RGB errors. So we multiplied a uniform constant to all the output albedo values of [56] to make sure its average intensity is the same as the ground truth, and also for [55].

For quantitative comparisons on real data, as we cannot obtain the dense ground truth albedo, we compare the weighted human disagreement rate (WHDR) [42] and the variances on manually selected regions with consistent albedos, and the results are shown in Table 3. The selected four regions are shown in the figure here. These regions are selected in the canonical frame and will be deformed according to the motions. The WHDR metric is the average rate of how often the results and the ground truth



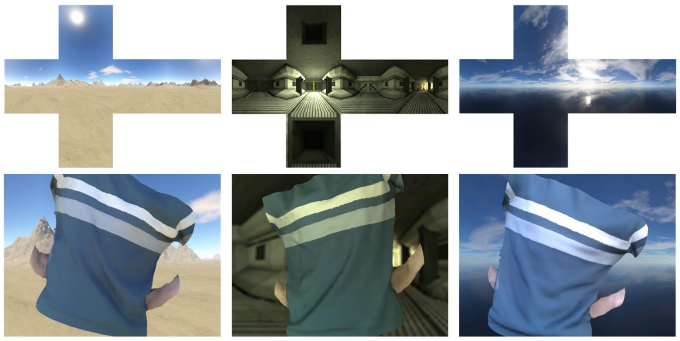


Fig. 13. Relighting. Top: environment lighting cubemaps; bottom: relighted results.

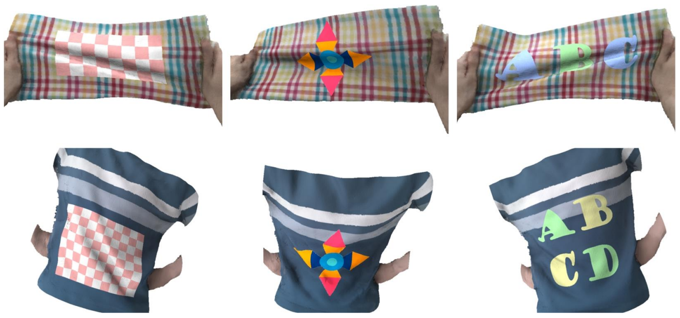


Fig. 14. Texture editing. We add three different patterns on two reconstructed objects.

relations between two points are inconsistent. Following [42], for each frame we first sample points in the four selected regions, and build edges between each pair of points. We obtain the ground truth by assuming that the points in region 1 and region 4 have the same albedo intensity, points in region 2 have brighter albedo than them, and points in region 3 are further brighter than those in region 2. We sample 20 points in region 1 and 10 points in each of the other regions. Totally $C_{50}^2 = 1225$ edges with ground truth relations are built on these 50 points. The final WHDR is computed by doing an average of all frames. All the compared methods share the same sampling points. δ is set to 10%, and all the edge weights are set to 1. Please refer to [42] for more details of WHDR. The variances in Table 3 are those of region 1 and region 2. Color values range from 0 to 255 in our quantitative comparisons.

4.4 Applications

As we reconstruct the albedo, geometry, and non-rigid motions, applications such as relighting and texture editing can be implemented with our method.

Relighting. Based on our reconstruction, the recorded object can be easily relighted by a given environment lighting. We use an offline render to generate the relighting sequences. Some results and the corresponding lighting cubemaps are shown in Fig. 13. Sequence results with rotating environment lightings are provided in our accompanying video.

Texture editing. It's also convenient to edit the reconstructed albedo in our method. The base geometry is reconstructed in the canonical frame, so we only need to edit the albedos of the selected surface points in the canonical frame. Then the solved non-rigid motions and displacements can be used to drive the object into different poses and to render the edited object. The rendered results are demonstrated in Fig. 14. Sequence results can also be found in our accompanying video.

5 Discussions

Our method can not handle the self-occlusion caused by surfaces in a long distance well, as we use a local search range when computing self-occlusion in real time. Also, cast shadows from other objects of which the geometries are not reconstructed can not be handled well. The motion tracking method that we use may fail if fast motions and topology changes occur, and this will further lead to failure in subsequent reconstructions. We use albedo to represent Lambertian surfaces; thus, highlights and specular reflectance cannot be well modeled. Deep learning may be used to improve performance. Please refer to our accompanying video for some examples of failure cases.

6 Conclusions

We propose a lightweight spatially varying lighting model that uses a masking strategy for each vertex to model the self-occlusion effect. Benefiting from the simple masking strategy, the model can be used in the solving of albedo without involving too much additional computation cost. A per-vertex displacement field is also reconstructed to improve the accuracy of motion estimation, and the real-time performance is maintained here by constructing regularization terms just from the historical data. Compared with the state-of-the-art techniques based on a single view, this method solves more accurate appearances as well as more detailed surface motions.

Acknowledgments

This work was supported by Beijing Natural Science Foundation (JQ19015), the NSFC (No.62021002, 61727808), the National Key R&D Program of China 2018YFA0704000. This work was supported by the Institute for Brain and Cognitive Science, Tsinghua University (THUIBCS) and Beijing Laboratory of Brain and Cognitive Intelligence, Beijing Municipal Education Commission (BLBCI). Feng Xu is the corresponding author.

References

[1] R. A. Newcombe, D. Fox, and S. M. Seitz, “Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 343–352.

[2] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, “Volumedeform: Real-time volumetric non-rigid reconstruction,” in *European Conference on Computer Vision*. Springer, 2016, pp. 362–379.

[3] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu, “Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera,” *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017.

[4] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar, “Acquiring the reflectance field of a human face,” in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 145–156.

[5] O. Alexander, M. Rogers, W. Lambeth, J.-Y. Chiang, W.-C. Ma, C.-C. Wang, and P. Debevec, “The digital emily project: Achieving a photorealistic digital actor,” *IEEE Computer Graphics and Applications*, vol. 30, no. 4, pp. 20–31, 2010.

[6] P. Gotardo, J. Riviere, D. Bradley, A. Ghosh, and T. Beeler, “Practical dynamic facial appearance modeling and acquisition,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–13, 2018.

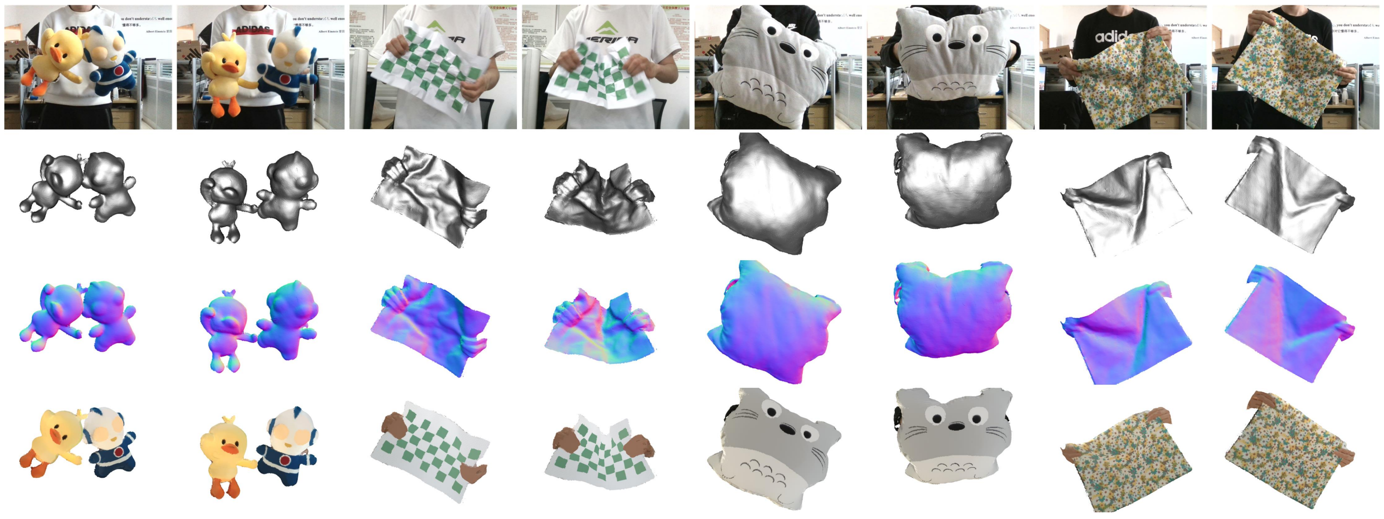


Fig. 15. Results of different dynamic objects. Row 1: input color images; row 2: reconstructed geometries; row 3: normal maps; row 4: albedo.

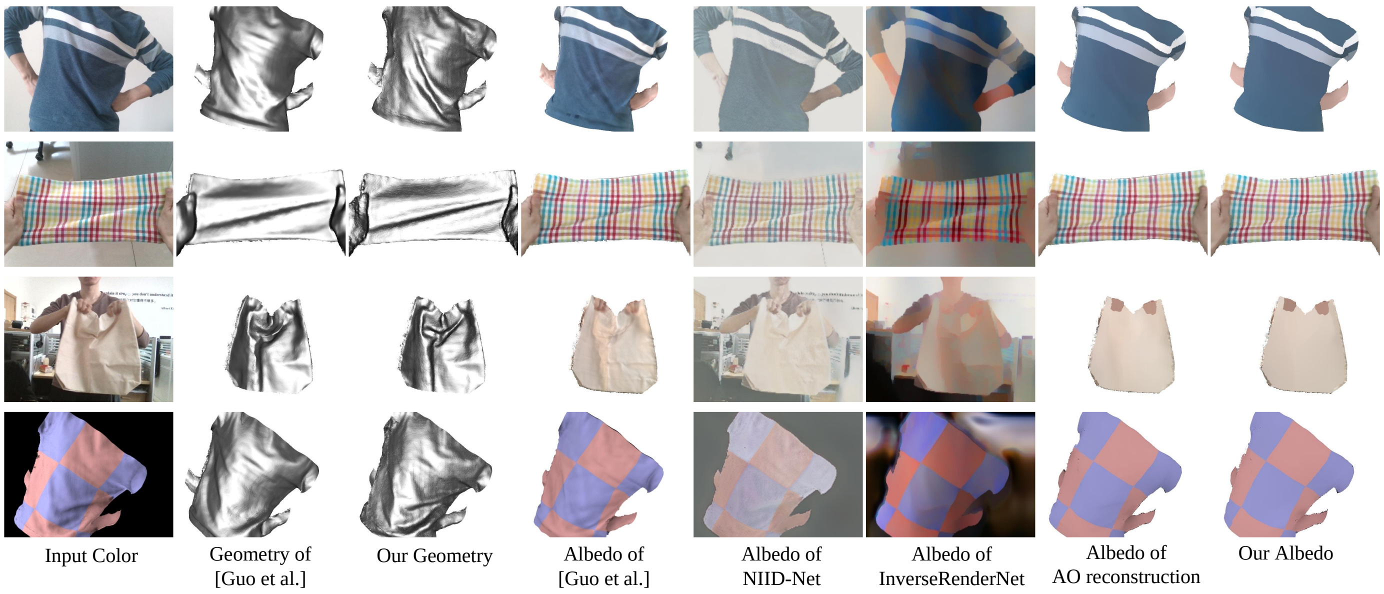


Fig. 16. Comparisons with [3], NIID-Net [56], InverseRenderNet [55], and AO reconstruction on different dynamic objects. The last row is from a synthetic sequence.

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

[8] R. Du, M. Chuang, W. Chang, H. Hoppe, and A. Varshney, "Montage4d: interactive seamless fusion of multiview video textures," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2018, pp. 1–11.

[9] K. Guo, P. Lincoln, P. Davidson, J. Busch, X. Yu, M. Whalen, G. Harvey, S. Orts-Escolano, R. Pandey, J. Dourgarian *et al.*, "The relightables: Volumetric performance capture of humans with realistic relighting," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–19, 2019.

[10] C. Zheng and F. Xu, "Dtexfusion: Dynamic texture fusion using a consumer rgbd sensor," *IEEE Transactions on Visualization & Computer Graphics*, no. 01, pp. 1–1, 2021.

[11] T. Tung, S. Nobuhara, and T. Matsuyama, "Simultaneous super-resolution and 3d video using graph-cuts," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[12] D. Casas, M. Volino, J. Collomosse, and A. Hilton, "4d video textures for interactive character appearance," in *Computer Graphics Forum*, vol. 33, no. 2. Wiley Online Library, 2014, pp. 371–380.

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

[14] A. E. Ichim, S. Bouaziz, and M. Pauly, "Dynamic 3d avatar creation from hand-held video input," *ACM Transactions on Graphics (ToG)*, vol. 34, no. 4, pp. 1–14, 2015.

[15] P. Garrido, M. Zollhöfer, D. Casas, L. Valgaerts, K. Varanasi, P. Pérez, and C. Theobalt, "Reconstruction of personalized 3d face rigs from monocular video," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 3, pp. 1–15, 2016.

[16] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2387–2395.

[17] S. Saito, L. Wei, L. Hu, K. Nagano, and H. Li, "Photorealistic facial texture inference using deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5144–5153.

[18] K. Nagano, J. Seo, J. Xing, L. Wei, Z. Li, S. Saito, A. Agarwal, J. Fursund, H. Li, R. Roberts *et al.*, "pagan: real-time avatars using dynamic textures," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 258.

[19] C. Wu, T. Shiratori, and Y. Sheikh, "Deep incremental learning for

efficient high-fidelity face tracking," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 234.

[20] S. Lombardi, J. Saragih, T. Simon, and Y. Sheikh, "Deep appearance models for face rendering," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–13, 2018.

[21] S.-E. Wei, J. Saragih, T. Simon, A. W. Harley, S. Lombardi, M. Perdoch, A. Hypes, D. Wang, H. Badino, and Y. Sheikh, "Vr facial animation via multiview image translation," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 67, 2019.

[22] R. Martin-Brualla, R. Pandey, S. Yang, P. Pidlypenskyi, J. Taylor, J. Valentin, S. Khamis, P. Davidson, A. Tkach, P. Lincoln *et al.*, "Lookingood: enhancing performance capture with real-time neural re-rendering," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–14, 2018.

[23] R. Pandey, A. Tkach, S. Yang, P. Pidlypenskyi, J. Taylor, R. Martin-Brualla, A. Tagliasacchi, G. Papandreou, P. Davidson, C. Keskin *et al.*, "Volumetric capture of humans with a single rgbd camera via semi-parametric learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9709–9718.

[24] C. Rother, M. Kiefel, L. Zhang, B. Schölkopf, and P. Gehler, "Recovering intrinsic images with a global sparsity prior on reflectance," *Advances in neural information processing systems*, vol. 24, pp. 765–773, 2011.

[25] L. Shen and C. Yeo, "Intrinsic images decomposition using a local and global sparse representation of reflectance," in *CVPR 2011*. IEEE, 2011, pp. 697–704.

[26] A. Bousseau, S. Paris, and F. Durand, "User assisted intrinsic images," *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 130–1, 2009.

[27] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman, "Ground truth dataset and baseline evaluations for intrinsic image algorithms," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 2335–2342.

[28] E. Garces, A. Munoz, J. Lopez-Moreno, and D. Gutierrez, "Intrinsic images by clustering," in *Computer graphics forum*, vol. 31, no. 4. Wiley Online Library, 2012, pp. 1415–1424.

[29] J. T. Barron and J. Malik, "Color constancy, intrinsic images, and shape estimation," in *European Conference on Computer Vision*. Springer, 2012, pp. 57–70.

[30] ——, "Shape, albedo, and illumination from a single image of an unknown object," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 334–341.

[31] S. Bi, X. Han, and Y. Yu, "An l1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–12, 2015.

[32] Z. Cheng, Y. Zheng, S. You, and I. Sato, "Non-local intrinsic decomposition with near-infrared priors," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2521–2530.

[33] Q. Chen and V. Koltun, "A simple model for intrinsic image decomposition with depth cues," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 241–248.

[34] J. Jeon, S. Cho, X. Tong, and S. Lee, "Intrinsic image decomposition using structure-texture separation and surface normals," in *European Conference on Computer Vision*. Springer, 2014, pp. 218–233.

[35] M. Hachama, B. Ghanem, and P. Wonka, "Intrinsic scene decomposition from rgb-d images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 810–818.

[36] X. Wei, G. Chen, Y. Dong, S. Lin, and X. Tong, "Object-based illumination estimation with rendering-aware neural networks," in *European Conference on Computer Vision*. Springer, 2020, pp. 380–396.

[37] G. Ye, E. Garces, Y. Liu, Q. Dai, and D. Gutierrez, "Intrinsic video and applications," *ACM Transactions on Graphics (ToG)*, vol. 33, no. 4, pp. 1–11, 2014.

[38] N. Bonneel, K. Sunkavalli, J. Tompkin, D. Sun, S. Paris, and H. Pfister, "Interactive intrinsic video editing," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, pp. 1–10, 2014.

[39] A. Meka, M. Zollhöfer, C. Richardt, and C. Theobalt, "Live intrinsic video," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–14, 2016.

[40] A. Meka, G. Fox, M. Zollhöfer, C. Richardt, and C. Theobalt, "Live user-guided intrinsic video for static scenes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 11, pp. 2447–2454, 2017.

[41] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European conference on computer vision*. Springer, 2012, pp. 611–625.

[42] S. Bell, K. Bala, and N. Snavely, "Intrinsic images in the wild," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–12, 2014.

[43] 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.

[44] B. Kovacs, S. Bell, N. Snavely, and K. Bala, "Shading annotations in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6998–7007.

[45] Z. Li and N. Snavely, "Cgintrinsics: Better intrinsic image decomposition through physically-based rendering," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 371–387.

[46] T. Zhou, P. Krahenbuhl, and A. A. Efros, "Learning data-driven reflectance priors for intrinsic image decomposition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3469–3477.

[47] T. Narihira, M. Maire, and S. X. Yu, "Direct intrinsics: Learning albedo-shading decomposition by convolutional regression," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2992–2992.

[48] J. Shi, Y. Dong, H. Su, and S. X. Yu, "Learning non-lambertian object intrinsics across shapenet categories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1685–1694.

[49] Q. Fan, J. Yang, G. Hua, B. Chen, and D. Wipf, "Revisiting deep intrinsic image decompositions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8944–8952.

[50] Z. Li and N. Snavely, "Learning intrinsic image decomposition from watching the world," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9039–9048.

[51] L. Cheng, C. Zhang, and Z. Liao, "Intrinsic image transformation via scale space decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 656–665.

[52] Y. Liu, Y. Li, S. You, and F. Lu, "Unsupervised learning for intrinsic image decomposition from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3248–3257.

[53] S. Sengupta, A. Kanazawa, C. D. Castillo, and D. W. Jacobs, "Sfsnet: Learning shape, reflectance and illuminance of facesin the wild'," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6296–6305.

[54] Y. Kanamori and Y. Endo, "Relighting humans: occlusion-aware inverse rendering for full-body human images," *arXiv preprint arXiv:1908.02714*, 2019.

[55] Y. Yu and W. A. Smith, "Inverserendernet: Learning single image inverse rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3155–3164.

[56] J. Luo, Z. Huang, Y. Li, X. Zhou, G. Zhang, and H. Bao, "Niid-net: Adapting surface normal knowledge for intrinsic image decomposition in indoor scenes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 12, pp. 3434–3445, 2020.

[57] B. K. Horn, "Shape from shading: A method for obtaining the shape of a smooth opaque object from one view," 1970.

[58] J. T. Barron and J. Malik, "Intrinsic scene properties from a single rgb-d image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 17–24.

[59] L.-F. Yu, S.-K. Yeung, Y.-W. Tai, and S. Lin, "Shading-based shape refinement of rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1415–1422.

[60] C. Wu, M. Zollhöfer, M. Nießner, M. Stamminger, S. Izadi, and C. Theobalt, "Real-time shading-based refinement for consumer depth cameras," *ACM Transactions on Graphics (ToG)*, vol. 33, no. 6, pp. 1–10, 2014.

[61] Y. Han, J.-Y. Lee, and I. So Kweon, "High quality shape from a single rgb-d image under uncalibrated natural illumination," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1617–1624.

[62] X. Zuo, S. Wang, J. Zheng, and R. Yang, "Detailed surface geometry and albedo recovery from rgb-d video under natural illumination," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3133–3142.

[63] R. Maier, K. Kim, D. Cremers, J. Kautz, and M. Nießner, "Intrinsic3d: High-quality 3d reconstruction by joint appearance and geometry optimization with spatially-varying lighting," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3114–3122.

[64] G. Xing, Y. Liu, H. Ling, X. Granier, and Y. Zhang, "Automatic spatially varying illumination recovery of indoor scenes based on a single rgb-d image," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 4, pp. 1672–1685, 2018.

[65] Z. Li, M. Shafiei, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker, "Inverse rendering for complex indoor scenes: Shape, spatially-

varying lighting and svbrdf from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2475–2484.

[66] Y. Yu, A. Meka, M. Elgharib, H.-P. Seidel, C. Theobalt, and W. A. Smith, "Self-supervised outdoor scene relighting," in *European Conference on Computer Vision*. Springer, 2020, pp. 84–101.

[67] M. Knecht, "State of the art report on ambient occlusion," *Vienna Institute of Technology, Technical Report*, 2007.

[68] S. Zhukov, A. Iones, and G. Kronin, "An ambient light illumination model," in *Eurographics Workshop on Rendering Techniques*. Springer, 1998, pp. 45–55.

[69] F. Hernell, P. Ljung, and A. Ynnerman, "Local ambient occlusion in direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 4, pp. 548–559, 2009.

[70] P. Shanmugam and O. Arikan, "Hardware accelerated ambient occlusion techniques on gpus," in *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, 2007, pp. 73–80.

[71] C. K. Reinbothe, T. Boubekeur, and M. Alexa, "Hybrid ambient occlusion." *Eurographics (Areas Papers)*, vol. 5, 2009.

[72] T. Ropinski, J. Meyer-Spradow, S. Diepenbrock, J. Mensmann, and K. Hinrichs, "Interactive volume rendering with dynamic ambient occlusion and color bleeding," in *Computer Graphics Forum*, vol. 27, no. 2. Wiley Online Library, 2008, pp. 567–576.

[73] J. Kontkanen and S. Laine, "Ambient occlusion fields," in *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 2005, pp. 41–48.

[74] L. Bavoil, M. Sainz, and R. Dimitrov, "Image-space horizon-based ambient occlusion," in *ACM SIGGRAPH 2008 talks*, 2008, pp. 1–1.

[75] J. Diaz, P.-P. Vazquez, I. Navazo, and F. Duguet, "Real-time ambient occlusion and halos with summed area tables," *Computers & Graphics*, vol. 34, no. 4, pp. 337–350, 2010.

[76] S. Laine and T. Karras, "Two methods for fast ray-cast ambient occlusion," in *Computer Graphics Forum*, vol. 29, no. 4. Wiley Online Library, 2010, pp. 1325–1333.

[77] D. Hauagge, S. Wehrwein, K. Bala, and N. Snavely, "Photometric ambient occlusion for intrinsic image decomposition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 639–651, 2015.

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

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

Chengwei Zheng received a B.S. degree in software engineering from Tsinghua University, Beijing, China, in 2018. He is currently working toward a Ph.D. degree in the School of Software, Tsinghua University. His research interests include dynamic reconstruction and 3D animation.

Wenbin Lin received a B.S. degree in Department of Automation, Tsinghua University, Beijing, China, in 2020. He is currently working toward a Ph.D. degree in the School of Software, Tsinghua University. His research interests include dynamic reconstruction and 3D animation.

Feng Xu received a B.S. degree in physics from Tsinghua University, Beijing, China, in 2007 and Ph.D. in automation from Tsinghua University, Beijing, China, in 2012. He is currently an associate professor in the School of Software, Tsinghua University. His research interests include face animation, performance capture, and 3D reconstruction.