Research Article

Hybrid mesh-neural representation for 3D transparent object reconstruction

Jiamin Xu[1,2], Zihan Zhu[2,3], Hujun Bao[2], and Weiwei Xu[2] (✉)

Abstract In this study, we propose a novel method to reconstruct the 3D shapes of transparent objects using images captured by handheld cameras under natural lighting conditions. It combines the advantages of an explicit mesh and multi-layer perceptron (MLP) network as a hybrid representation to simplify the capture settings used in recent studies. After obtaining an initial shape through multi-view silhouettes, we introduced surface-based local MLPs to encode the vertex displacement field (VDF) for reconstructing surface details. The design of local MLPs allowed representation of the VDF in a piecewise manner using two-layer MLP networks to support the optimization algorithm. Defining local MLPs on the surface instead of on the volume also reduced the search space. Such a hybrid representation enabled us to relax the ray–pixel correspondences that represent the light path constraint to our designed ray–cell correspondences, which significantly simplified the implementation of a single-image-based environment-matting algorithm. We evaluated our representation and reconstruction algorithm on several transparent objects based on ground truth models. The experimental results show that our method produces high-quality reconstructions that are superior to those of state-of-the-art methods using a simplified data-acquisition setup.

Keywords transparent object; 3D reconstruction; environment matting; neural rendering

1 College of Computer Science, Hangzhou Dianzi University, Hangzhou 310018, China. E-mail: superxjm@yeah.net.

2 State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China; E-mail: Z. Zhu, zhuzihan2000@gmail.com; H. Bao, bao@cad.zju.edu.cn; W. Xu, xww@cad.zju.edu.cn (✉).

3 College of Computer Science, ETH Zürich, Zürich 8092, Switzerland.

1 Introduction

The acquisition of 3D models is a frequent problem in computer graphics and vision. Most existing methods, such as laser scanning and multi-view reconstruction, are based on observations of surface color. Consequently, the surface is assumed to be opaque and approximately Lambertian. These methods cannot be directly applied to transparent objects because the appearance of a transparent object is indirectly observed owing to the complex refraction and reflection light paths at the interface between air and transparent materials.

A core technical challenge in 3D transparent object reconstruction is that of handling the dramatic changes in appearance that occur when observing an object in a multi-view setting. Slight changes in an object's shape can lead to nonlocal changes in appearance owing to the complexity of light paths. To address this issue, we utilized ray–pixel correspondence (i.e., the correspondence between a camera ray and a pixel on a static background pattern displayed on a monitor) and ray–ray correspondence (i.e., the correspondence between a camera ray and the incident ray from the background pattern) to provide light path constraints to facilitate 3D transparent object reconstruction [1–3]. A differentiable refraction-tracing technique can be applied to reduce the complexity of the capture setting, and the 3D shape can be recovered through ray–pixel correspondences as shown in Ref. [4]. However, in this method, a transparent object should be placed on a turntable under controlled lighting conditions. Li et al. [5] trained a physics-based neural network to handle complex light paths for 3D transparent objects. The network was trained on a synthetic dataset with a differentiable path tracing

rendering technique. This method optimizes surface normals in a latent space; thus, it can reconstruct 3D transparent objects under natural lighting conditions when receiving an environment map and a few images as input. However, this frequently produces overly smooth reconstruction results.

In this study, we consider how to combine the advantages of explicit meshes and multilayer perceptron (MLP) networks, a hybrid representation, to address the problem of reconstructing transparent objects under natural lighting conditions using images captured with a handheld camera. This representation can be reconstructed through optimization using a differentiable path-tracing rendering technique. The key idea is to use MLP to encode a vertex displacement field (VDF) defined on a base mesh to reconstruct surface details, wherein the base mesh is created using multi-view silhouette images. Our design is motivated by two observations. First, the representation of functions using MLP has been demonstrated to be efficient in optimization and robust to noise [6–8]. The MLP network parameterizes the VDF with weight parameters globally. Hence, it implicitly provides global constraints on changes in VDF. Second, defining the MLP-parameterized VDF on the base mesh reduces the search space during optimization [9]. This significantly accelerates the optimization process compared to MLP-based volumetric representation.

The advantage of our hybrid representation is that it allows for relaxation of the capture setting. Because the global smoothness constraints between vertex displacements are implied in MLP weights, the ray–pixel correspondence required in the optimization can be significantly relaxed to a ray–cell correspondence in our pipeline. Consequently, we can simplify the background pattern design and develop a robust single-image environment matting (EnvMatt) algorithm for handling images captured under natural lighting conditions. Compared to the capture settings used in Wu et al. [2] and Lyu et al. [4]'s work, our handheld capture setting is low-cost and simple. Moreover, we propose to represent VDF using a small number of local MLPs. Each MLP is responsible for encoding a local VDF. This strategy enables the design of small-scale MLPs to further accelerate the optimization process. A fusion module is designed to disperse the gradient information of the

displacement vectors of vertices to their neighboring local MLPs. This module helps maintain global constraints between local MLPs and produces high-quality reconstruction results.

The contributions of this study are summarized as follows.

- We present a hybrid representation that employs explicit mesh and local-MLP based functions to represent the detailed surface for transparent objects. This approach enables us to design small-scale MLPs to accelerate our optimization algorithm's convergence and achieve high-quality 3D reconstruction results for transparent objects.
- We propose a ray–cell correspondence as a relaxed representation of the light path constraint. The ray–cell correspondence is easier to capture, leading to a simplified capture setting under natural lighting conditions. Furthermore, it also eases the implementation of the EnvMatt algorithm.

The experimental results demonstrate that our method can produce 3D models with details for a variety of transparent objects, as illustrated in Fig. 1. With our simplified capture setting under natural light conditions, our reconstruction results were superior to those of state-of-the-art 3D reconstruction algorithms for transparent objects.

Fig. 1 Our reconstruction results paired with the associated renderings of three transparent objects. The fine surface details can be reconstructed well via our method using images captured with a handheld camera under natural lighting conditions.

2   Related work

Our algorithm is designed on the basis of considerable previous research. Here, we review the literature most related to the present work, including studies on transparent object reconstruction, differentiable rendering, and EnvMatt.

Transparent object reconstruction. Many transparent object reconstruction techniques utilize special hardware setups, including polarization [10–12], time-of-flight cameras [13], tomography [14], moving point light sources [15, 16], and light field

probes [17]. The proposed algorithm is most closely related to shape-from-distortion and light-path triangulation. Kutulakos and Steger [1] formulated the reconstruction of a refractive or mirror-like surface as a light path triangulation problem. Given a function that maps each point in an image onto a 3D "reference point" that is indirectly projected onto it, the authors characterized a set of reconstructible cases that depended only on the number of points along a light path. The mapping function can be estimated using the EnvMatt algorithm with a calibrated acquisition setup, denoted by ray–point correspondences. A ray–ray correspondence can be uniquely determined with two distinct reference points along the same ray.

In accordance with light path triangulation, one reconstructible case is that of single-refraction surfaces [18–20], particularly fluid surfaces [21–23]. Another tractable case is that of transparent objects when rays undergo refraction twice [24–26]. Wu et al. [2] recently reconstructed the full shape of a transparent object by first extracting ray–ray correspondences and then performing separate optimization and multi-view fusion. Lyu et al. [4] proposed the extraction of per-view ray–point correspondences using the EnvMatt algorithm in Ref. [27], and utilized differentiable rendering to progressively optimize an initial mesh.

In addition to optimization-based methods, deep learning techniques can also be incorporated to resolve depth-normal ambiguities [28, 29]. Li et al. [5] suggested performing optimization in the feature space to obtain surface normals. Subsequently, they performed multi-view feature mapping and 3D point-cloud reconstruction to obtain a 3D shape. Their method works on a simple acquisition setting with only one known environment map and approximately 10 captured images. However, their reconstructed transparent object may lose some details owing to the domain gap between the real-world images and synthetic training data.

Differentiable rendering. In accordance with the simulation level of light transport, differentiable rendering algorithms in computer graphics can be roughly divided into three categories: differentiable rasterization [30–34], differential volumetric rendering [7, 8, 35, 36], and differentiable ray-tracing [37–43]. Differentiable rasterization can be used to optimize

a mesh itself or its features, and the neural network parameters defined on the mesh. Differentiable volumetric rendering can be used to optimize implicit shape representations, such as the implicit occupancy function [44, 45], signed distance function (SDF) [7, 46], and unsigned distance function [47]. Differentiable rendering has also been used to optimize deep surface light fields [9]. This method represents per-vertex view-dependent reflections using an MLP. While we also utilize surface-based MLPs, our focus is different; our method employs local MLPs to represent VDF locally to reconstruct surface details and design a fusion layer to avoid discontinuities at the overlapped surface areas.

Considering that a light path with refraction is determined by the front and back surfaces of a transparent object, the geometry can be optimized in an iterative manner with forward ray tracing and backward gradient propagation. To this end, our algorithm exploits differential ray tracing to handle the light path of the reflected and refracted rays on the surface of transparent objects.

Environment matting. EnvMatt, which captures how an object refracts and reflects environment light, can be viewed as an extension of alpha matting [48, 49]. Image-based refraction and reflection are represented as pixel-texel (texture pixel) correspondences, in which environments are represented as texture maps. The seminal work of Zongker et al. [27] extracted EnvMatt from a series of 1D Gray codes, assuming that each pixel is only related to a rectangular background region. Chuang et al. [50] extended this work to recover a more accurate model at the expense of using more structured light backdrops. They also proposed a simplified EnvMatt algorithm that uses only a single backdrop. A pixel-texel correspondence search can also be performed in the wavelet [51] and frequency [52] domains. The number of required patterns can be reduced by combining them with a compressive sensing technique [53]. Chen et al. [54] recently presented a deep learning framework called TOM-Net to estimate EnvMatt as a refractive flow field. The aforementioned methods require images to be captured under controlled lighting conditions (e.g., in a dark room) to avoid the influence of ambient light. Wexler et al. [55] developed an EnvMatt algorithm for handling natural-scene backgrounds. However,

their method required capturing a set of images using a fixed camera and a moving background.

3   Overview

A transparent object reconstruction pipeline is shown in Fig. 2. This pipeline begins by reconstructing an object's rough shape (initial shape) from a collection of multi-view silhouettes. Instead of the space-carving method [56], we utilized the MLP-based signed distance function (SDF) in IDR [7] to obtain a smooth initial shape, as shown in Fig. 2. Subsequently, we employ the MLP network to represent the vertex displacement field (VDF) on the initial shape to reconstruct the surface details. This hybrid surface representation combines an explicit mesh and MLP-based neural network. In the following section, we detail the hybrid representation and optimization algorithm for reconstructing the representation from multi-view images.

Hybrid representation. We choose to encode the surface details with VDF as it is defined on a 2D manifold instead of the entire 3D space, reducing the search space of the optimization algorithm and producing high-quality reconstruction results. Moreover, we used to represent the displacement field defined on the vertices to simplify the optimization. Such hybrid representation can combine explicit vertex optimization to accelerate convergence and MLP-based neural representation, as in IDR [7] to enforce global constraints among vertices and improve the robustness of the optimization.

Rather than encoding the VDF using a single MLP, we found that representing the field with a couple of small local MLPs can achieve better

results. As shown in Fig. 2, each local MLP encodes the displacement vectors of the vertices within one cluster extracted from the mesh of the initial shape using the variational shape approximation (VSA) algorithm [57]. To avoid mesh discontinuities across local MLPs, we also added a fusion layer to blend the displacement vectors of neighboring vertices based on the geodesic distances on the mesh [58].

Optimization for VDF. The VDF is optimized based on the multi-bounce (up to two-bounces) light path constraints and the consistency between the rendering of our representation and the captured RGB images. The rendering procedure was performed using a recursive differentiable path-tracing algorithm [59].

The light path constraint due to multi-bounce refraction is approximated by a mapping function that maps each pixel in the input image onto a pixel of the background pattern image, which can be obtained using the EnvMatt algorithm. We store the background image as a texture. However, we found that traditional EnvMatt algorithms are either restricted to using multiple images with a fixed camera or are sensitive to natural light conditions. Consequently, we designed a grid-based background pattern to establish the correspondence between a foreground pixel p that covers a small part of the object surface and the cell of the grid. This correspondence is recorded as a tuple $\langle p, u \rangle$, where u is the pixel coordinate of the cell center on the background image. In this manner, the mapping function is simplified, but remains efficient in providing information on the light path constraint to facilitate optimization.
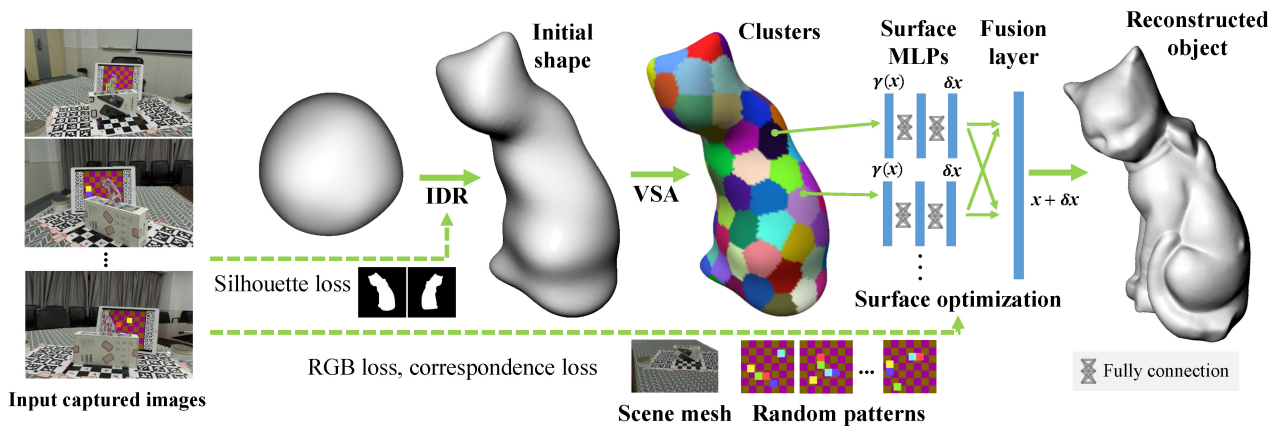
In the remainder of this paper, we first describe our

Fig. 2   Pipeline of our approach.

data pre-processing steps (Section 4.1), including the image acquisition setup and grid-based single-image EnvMatt algorithm. Then, we present the details of the initial shape reconstruction (Section 4.2) and surface optimization steps (Section 4.3).

4 Method

4.1 Pre-processing

Data acquisition. We captured images by using a Canon EOS 60D digital single-lens reflex camera. The transparent object to be captured was placed on a desk with preprinted AprilTags [60] underneath. AprilTags were used to facilitate the image registration. To capture the ray–cell correspondences for EnvMatt, we placed an iPad as a monitor behind the transparent object to display a grid-based background pattern. The pattern displayed P_i for the ith image was changed after capturing every four images. We manually moved the position of the iPad after capturing 60 images in our implementation. This setting is designed to incorporate more valid ray–cell correspondences to cover more surface areas. All patterns were pre-generated and used in the EnvMatt step. In addition to images with a background pattern, we captured more images without background patterns to provide further constraints in RGB space. During capture, the iPad was moved two to four times, and the total number of captured images ranged from 154 to 302, owing to the complexity of the object's surface shape.

Grid-based single image envmatt. Similar to Ref. [50], we can assume that a transparent object has no intrinsic color. Therefore, the correspondence between pixels that cover the object's surface and pixels on the background pattern can be calculated by searching a color space. However, we found that the color ramp pattern in Ref. [50] is sensitive to ambient light under natural light conditions because of the invertible and smooth properties of the pattern image in RGB space. Therefore, we designed grid-based background patterns in this study. The color of each grid cell was constant and was designed to create sharp boundaries between cells. As shown in Fig. 5, each pattern consists of a 7×7 checkboard with two different colors $\{c_h^i | i \in \{1, 2\}\}$. Then, we randomly sample five cells on the grid and move each cell with a random offset. These moved cells are assigned

one of five distinct salient colors $\{c_s^i | i \in \{1, \cdots, 5\}\}$ as shown in the rightmost column in Fig. 5. We chose these five salient colors because they have sufficient mutual distance in RGB space, which makes them more robust to the influence of environmental lighting. These salient cells are used to calculate the correspondence tuple, and the cell size is chosen as the balance between the surface coverage generated by refracted cells and EnvMatt precision.

The proposed EnvMatt algorithm calculates ray–cell correspondences for each pixel. Given a captured image, for each pixel p with color c_p, the corresponding cell center u can be calculated as Eq. (1).

$$u = \begin{cases} cr(\underset{i}{\arg\min} \|c_s^i - c_p\|), & \underset{i}{\min} \|c_s^i - c_p\| < \gamma_1 \\ \inf, & \underset{i,j}{\min}(\|c_h^i - c_p\|, \\ & \|c_s^j - c_p\|) > \gamma_2 \\ \text{none}, & \text{otherwise} \end{cases}$$

(1)

where $cr(\cdot)$ returns a pre-defined cell center. If c_p is similar to any salient color, u is a valid correspondence. u is marked as inf when no correspondence exists for p on the designed grid pattern, indicating that the light path terminates outside the grid. Otherwise, the pixel p corresponds to a pixel on the grid pattern with checkboard colors. In this case, a precise ray–cell correspondence cannot be obtained. Therefore, we set the correspondence between p and none. The parameters γ_1 and γ_2 are set as 0.3 and 0.4, respectively.

Once capturing has been performed, we use the 3D reconstruction software RealityCapture [61] to register the captured images, which enables us to trace rays in a unified coordinate frame during differentiable rendering.

Image registration and 3D reconstruction. After capturing the images (please see example images illustrated in Fig. 3), we used the 3D reconstruction software RealityCapture [61] to register the captured images, enabling us to trace rays in a unified coordinate frame during differentiable rendering. Given that the position of the iPad is changed every 60 images to provide more ray–cell correspondences, we reconstructed the 3D scene with textures that contain the iPad every 60 images independently, resulting in several components in RealityCapture. Each component recorded the
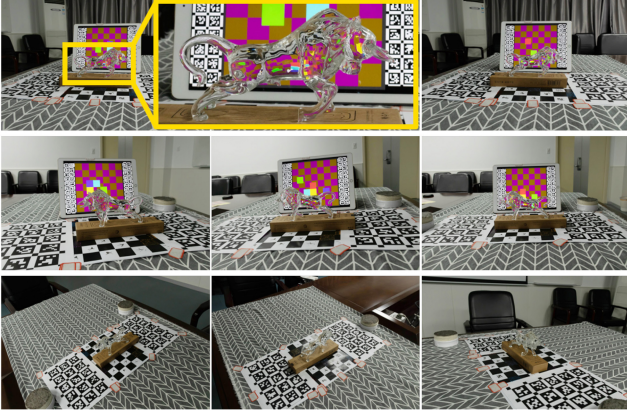
Fig. 3 Captured images.

information of each independently reconstructed 3D scene. All components are then registered based on AprilTags [60] beneath the object, as shown in Fig. 4.

Considering that the background pattern displayed on the iPad is changed during the capturing of every four images, incorrect matching points on the iPad's surface are produced, resulting in failure of 3D iPad plane reconstruction. To address this issue, we displayed additional AprilTags surrounding the background patterns to add extra matching points to guarantee the success of the iPad plane reconstruction.

Each 3D plane P_i of the background pattern was detected using the RANSAC shape-detection algorithm [62] as illustrated in Fig. 4. We also generated local coordinates for each grid pattern to map the 3D points on the plane onto the texture coordinates.

4.2 Initial shape reconstruction

We utilized IDR [7] with silhouette (mask) loss to obtain the initial shape of a transparent object. The

object masks were manually annotated on several selected images. The number of masks used is listed in Table 1. We only used silhouette loss, and thus, the "neural renderer" MLP in IDR was removed. As shown in Fig. 6, IDR with silhouette loss produces smoother reconstruction results than the space-carving algorithm. After reconstructing the initial mesh, we uniformly scale the mesh such that the diameter of its bounding ball equals one, extract fine-grain mesh and perform edge collapse mesh simplification. During simplification, we fixed the target-side length of each triangle to 0.005.

Table 1 Statistics of data acquisition. #Img denotes the number of captured images. #Mask, #Pattern, and #Move denote the number of annotated masks, the number of our grid-based patterns presented on iPad, and the number of iPad movements, respectively

Object	Size (cm)	#Img	#Mask	#Pattern	#Move
Cow	$9.5 \times 4.5 \times 17.5$	154	19	30	2
Cat	$8 \times 2.5 \times 4.5$	302	29	60	4
Dog	$11 \times 3.5 \times 11.5$	300	19	60	4
Trophy	$12 \times 7 \times 7$	172	14	30	2
Brick	$10 \times 4.5 \times 20$	178	15	30	2

(a) Input image (crop)   (b) Correspondences   (c) Designed pattern   (d) Colors

Fig. 5 Grid-based single image EnvMatt procedure. (a) Input image. (b) EnvMatt results: colored pixels indicate that their traced rays terminate inside the cells with designed salient colors; black pixels indicate that the rays terminate inside cells with checkboard colors, and gray pixels indicate that rays terminate outside the pattern. (c) The designed pattern. The circles indicate the centers of salient cells. (d) Chosen colors for salient and checkboard cells.
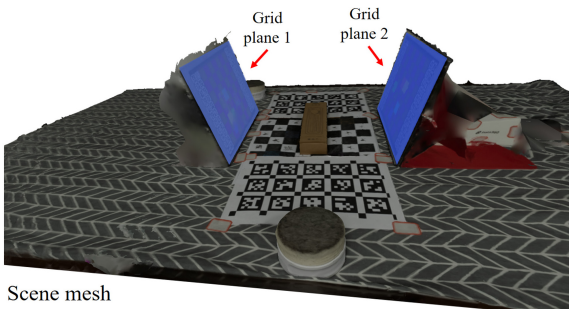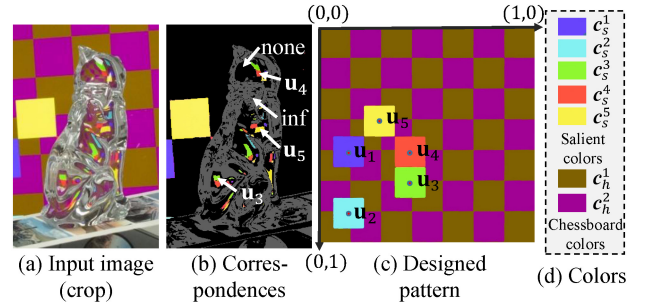

Fig. 4 Scene mesh and pattern planes. The iPad is moved twice. During image registration step, images are registered to the same global coordinates.

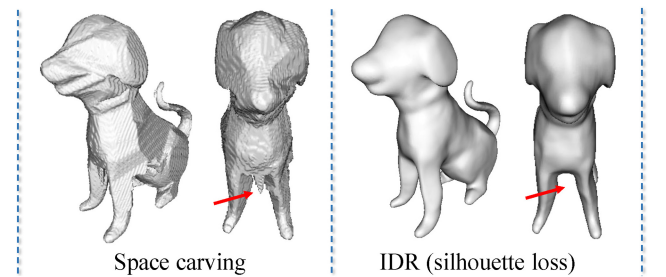Space carving   IDR (silhouette loss)

Fig. 6 Space carving vs. IDR with silhouette loss. The space carving method produces artifact in the occluded areas (indicated by the red arrow).

4.3 Surface optimization through differentiable rendering

Given the initial shape as the base mesh, we first group the mesh triangles into several clusters and then assign each cluster to a surface-based MLP. Thus, VDF can be computed as a fusion of the output of the surface-based local MLPs. In particular, for each vertex $\boldsymbol{x}_i$ within cluster $C_{\boldsymbol{x}_i}$, each local MLP MLP_k outputs a displacement vector $\hat{\delta}\boldsymbol{x}_i$ as Eq. (2).

$$\hat{\delta}\boldsymbol{x}_i = \mathrm{MLP}_{C_{\boldsymbol{x}_i}}(\boldsymbol{x}_i) \qquad (2)$$

To avoid discontinuity at the cluster boundaries, we introduce a differentiable fusion layer to obtain the final displacement vector $\delta\boldsymbol{x}_i$:

$$\delta\boldsymbol{x}_i = \sum_j w(\boldsymbol{x}_i, \boldsymbol{x}_j) \cdot \hat{\delta}\boldsymbol{x}_j \qquad (3)$$

where $w(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-d(\boldsymbol{x}_i, \boldsymbol{x}_j)/\sigma)$, and d is the geodesic distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, which is calculated using the heat transportation-based method proposed by Crane et al. [58]. We set the parameter $\sigma = 0.005$ in our experiments.

The architecture of each local MLP is illustrated in Fig. 7 with two fully connected (FC) layers. For each MLP, each input 3D vertex on the surface was first mapped to a 99-dimensional feature using positional encoding. We used positional encoding frequencies $L = 16$. In this case, the feature vector size is $N = 99$, where $N = L \times 2 \times 3 + 3$. Here, 2 for sin and cos and 3 for x, y, and z. The first FC layer with weight normalization and ReLu activation maps the 99-dimensional feature to a 128-dimensional latent feature. The second FC layer, with weight normalization and tanh activation, mapped the latent feature to a 3D displacement. Before
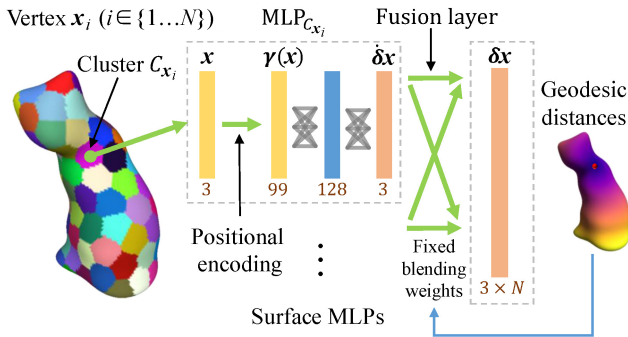

Fig. 7 Local MLP representation. Each local MLP is responsible for representing the vertex displacements inside a VSA cluster (shown as the colored patch on the object surface). A fusion layer is used to fuse the vertex displacements output by the local MLPs into a smooth VDF on the surface.

optimization, all weights were initialized to produce a zero-displacement field.

In the following, we first describe how to extract clusters from the base mesh and then describe the details of the designed loss terms and our optimization procedure.

4.3.1 Cluster extraction

We utilized the Variational Shape Approximation (VSA) algorithm [57] to segment the initial shape into several clusters. The VSA algorithm tends to merge co-planar vertices into the same cluster by minimizing $\mathcal{L}^2$ distortion error, which measures the error between a cluster and its linear proxy (plane). To balance the size of each cluster, we add a Euclidean distance error that sums the Euclidean distance between each vertex and its cluster center with a weight of 0.005.

4.3.2 Loss terms

We minimize the loss function to search for the weight parameters of local MLPs as given in Eq. (4).

$$\mathcal{L}_{\mathrm{total}} = \lambda_{\mathrm{rgb}}\mathcal{L}_{\mathrm{rgb}} + \lambda_{\mathrm{corr}}\mathcal{L}_{\mathrm{corr}} + \lambda_{\mathrm{ncorr}}\mathcal{L}_{\mathrm{ncorr}}$$
$$+ \lambda_{\mathrm{sil}}\mathcal{L}_{\mathrm{sil}} + \lambda_{\mathrm{reg}}\mathcal{L}_{\mathrm{reg}} \qquad (4)$$

which is the sum of the following five terms: RGB loss $\mathcal{L}_{\mathrm{rgb}}$, ray–cell correspondence loss $\mathcal{L}_{\mathrm{corr}}$, no-correspondence loss $\mathcal{L}_{\mathrm{ncorr}}$, silhouette loss $\mathcal{L}_{\mathrm{sil}}$ [63], and regularization loss $\mathcal{L}_{\mathrm{reg}}$. The default values of the weights, i.e., λ_{rgb}, λ_{corr}, λ_{ncorr}, λ_{sil} and λ_{reg} were set as 0.001, 0.1, 0.03, 50.0, and 1.0, respectively, in our experiments.

RGB loss. RGB loss measures the difference between the pixel color $\boldsymbol{c_p}$ and the rendering result $\boldsymbol{c_p^{in}}$ of its traced ray on the basis of the recursive ray tracing algorithm. For a pixel $\boldsymbol{p}$ on a captured image $\boldsymbol{I}$, we first trace a ray $\boldsymbol{l_p^{in}}$ from the viewpoint associated with $\boldsymbol{I}$. Then, the recursive ray-tracing algorithm is triggered to obtain the reflection color $\boldsymbol{c_p^{r1}}$ through the first-bounce reflection ray $\boldsymbol{l_p^{r1}}$ and the refraction color $\boldsymbol{c_p^{t2}}$ through the second-bounce refraction ray $\boldsymbol{l_p^{t2}}$ by intersecting and fetching textures from a scene mesh or a corresponding pattern. We consider only single-bounce reflection and double-bounce refraction in the proposed algorithm.

In particular, the camera ray $\boldsymbol{l_p^{in}}$ first intersects the front surface at point $\boldsymbol{x}_1$ and is refracted by the surface with normal $\boldsymbol{n}_1$ following Snell's law, as shown in Fig. 8. The normal of each mesh vertex is obtained by averaging the normals of the triangles incident to the vertex using triangle areas as the weighting
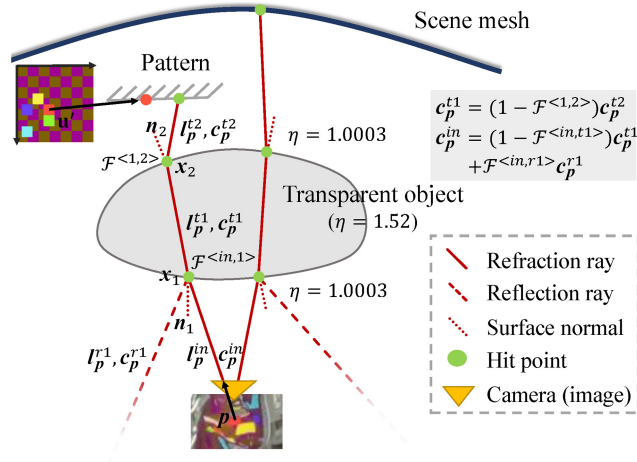
$$c_{\boldsymbol{p}}^{t1} = (1 - \mathcal{F}^{<1,2>})c_{\boldsymbol{p}}^{t2}$$
$$c_{\boldsymbol{p}}^{in} = (1 - \mathcal{F}^{<in,t1>})c_{\boldsymbol{p}}^{t1}$$
$$+ \mathcal{F}^{<in,r1>}c_{\boldsymbol{p}}^{r1}$$

Fig. 8 Recursive ray-tracing procedure. During rendering, the refraction and reflection rays fetch colors from the scene texture and background pattern. If pixel $\boldsymbol{p}$ has its corresponding cell center $\boldsymbol{u}$, then the mesh should be optimized to make the intersection between $\boldsymbol{l}_{\boldsymbol{p}}^{t2}$ and the 3D pattern plane within the corresponding cell (the red point in the illustration).

shceme, whereas the normal for each ray-triangle intersection point is obtained by interpolating the three vertex normals of the triangle using bary-centric coordinates. Then, the first-bounce refraction ray $\boldsymbol{l}_{\boldsymbol{p}}^{t1}$ is traced until it hits the back surface at point $\boldsymbol{x}_2$ with normal $\boldsymbol{n}_2$. The second-bounce refraction ray $\boldsymbol{l}_{\boldsymbol{p}}^{t2}$ is generated recursively. Similarly, the reflection ray $\boldsymbol{l}_{\boldsymbol{p}}^{r1}$ is traced in the mirror-reflection direction. For RGB loss, we only consider the camera rays with first-bounce reflection ray $\boldsymbol{l}_{\boldsymbol{p}}^{r1}$ and second-bounce refraction ray $\boldsymbol{l}_{\boldsymbol{p}}^{t2}$. We store the valid camera rays whose second-bounce refraction rays can hit the scene mesh or pattern without occlusion and total internal reflection as a binary mask $\boldsymbol{M}^t$ on the image $\boldsymbol{I}$.

For each refraction ray above, the refraction color is attenuated along the light path, according to the Fresnel term $\mathcal{F}$ [64]. Taking two successive refraction rays $\boldsymbol{l}_{\boldsymbol{p}}^{t1}$ and $\boldsymbol{l}_{\boldsymbol{p}}^{t2}$ as an example, where $\boldsymbol{n}_2$ is the surface normal and η^{i} and η^{o} are the two indices of refraction (IOR) inside and outside the object respectively, the Fresnel term $\mathcal{F}^{\langle t1,t2 \rangle}$ can be computed as Eq. (5).

$$\mathcal{F}^{\langle t1,t2 \rangle} = \frac{1}{2}\left(\frac{\eta^{\mathrm{i}}\boldsymbol{r}_{\boldsymbol{p}}^{t1} \cdot \boldsymbol{n}_2 - \eta^{\mathrm{o}}\boldsymbol{r}_{\boldsymbol{p}}^{t2} \cdot \boldsymbol{n}_2}{\eta^{\mathrm{i}}\boldsymbol{r}_{\boldsymbol{p}}^{t1} \cdot \boldsymbol{n}_2 + \eta^{\mathrm{o}}\boldsymbol{r}_{\boldsymbol{p}}^{t2} \cdot \boldsymbol{n}_2}\right)^2 \quad (5)$$
$$+ \frac{1}{2}\left(\frac{\eta^{\mathrm{o}}\boldsymbol{r}_{\boldsymbol{p}}^{t1} \cdot \boldsymbol{n}_2 - \eta^{\mathrm{i}}\boldsymbol{r}_{\boldsymbol{p}}^{t2} \cdot \boldsymbol{n}_2}{\eta^{\mathrm{o}}\boldsymbol{r}_{\boldsymbol{p}}^{t1} \cdot \boldsymbol{n}_2 + \eta^{\mathrm{i}}\boldsymbol{r}_{\boldsymbol{p}}^{t2} \cdot \boldsymbol{n}_2}\right)^2$$

$$c_{\boldsymbol{p}}^{t1} = \left(1 - \mathcal{F}^{\langle t1,t2 \rangle}\right)\left(\frac{\eta^{t1}}{\eta^{t2}}\right)^2 c_{\boldsymbol{p}}^{t2} \quad (6)$$

In our experiments, we set the IOR of air to 1.0003

and the IOR of the object material to 1.52. As shown in Fig. 8, the reflection color $c_{\boldsymbol{p}}^{r1}$ and refraction color $c_{\boldsymbol{p}}^{t2}$ are attenuated after ray tracing, and are used to obtain the rendered color $c_{\boldsymbol{p}}^{in}$.

The reflection and refraction colors are fetched from the textured scene mesh or corresponding pattern. If one reflected ray does not intersect with the scene mesh or pattern, we only set $c_{\boldsymbol{p}}^{r1}$ to zero. RGB loss $\mathcal{L}_{\mathrm{rgb}}$ is defined as Eq. (7).

$$\mathcal{L}_{\mathrm{rgb}} = \frac{1}{|\boldsymbol{M}^t|}\sum_{\boldsymbol{p}} \boldsymbol{M}_{\boldsymbol{p}}^t \|c_{\boldsymbol{p}}^{in} - c_{\boldsymbol{p}}\|_1 \quad (7)$$

where $\|\cdot\|_1$ indicates the L1 norm, $c_{\boldsymbol{p}}^{in}$ is the color rendered from ray tracing, and $c_{\boldsymbol{p}}$ is the color of the pixel $\boldsymbol{p}$. In our implementation, we sampled rays based on 200×200 cropped patches instead of individual rays. To match the pixel color and rendered color at the coarse-to-fine level, we filter $c_{\boldsymbol{p}}^{in}$ and $c_{\boldsymbol{p}}$ with three differentiable Gaussian filters ($7 \times 7, 11 \times 11, 13 \times 13$), and then add the outputs together [65]. These Gaussian filters had standard deviations of 2.5, 7.5, and 7.5, respectively. Moreover, we filtered the textures of the scene mesh and patterns using Gaussian filters to smooth the gradients.

Correspondence loss and no correspondence loss. Correspondence loss $\mathcal{L}_{\mathrm{corr}}$ and no-correspondence loss $\mathcal{L}_{\mathrm{ncorr}}$ were used to enforce the refraction ray $\boldsymbol{l}_{\boldsymbol{p}}^{t2}$ to match the ray–cell correspondence described in Section 4.1. For each pixel $\boldsymbol{p}$ and its corresponding cell, where the cell center is $\boldsymbol{u}$ and the side length is l, we obtained the ray-plane intersection point between $\boldsymbol{l}_{\boldsymbol{p}}^{t2}$ and the current pattern plane $\boldsymbol{P}$. Then, the intersection point was projected onto 2D space and transformed into texture coordinates as $\boldsymbol{v}_{\boldsymbol{l}_{\boldsymbol{p}}^{t2}}$. Here, the 2D image space of the planar patterns was pre-rectified to align the texture coordinates with the grid boundaries. As shown in Fig. 5, the texture coordinates of the four grid corners were set as $(0,0)$, $(1,0)$, $(1,1)$, and $(0,1)$, with the texture coordinates of the grid center $g = (0.5, 0.5)$. The correspondence loss $\mathcal{L}_{\mathrm{corr}}$ constrained the projected pixel $\boldsymbol{v}_{\boldsymbol{l}_{\boldsymbol{p}}^{t2}}$ to its corresponding grid center $\boldsymbol{u}$ as Eq. (8).

$$\mathcal{L}_{\mathrm{corr}} = \frac{1}{|\boldsymbol{M}^t|}\sum_{\boldsymbol{p}} \boldsymbol{M}_{\boldsymbol{p}}^t d\left(\boldsymbol{v}_{\boldsymbol{l}_{\boldsymbol{p}}^{t2}}, \boldsymbol{u}\right) \quad (8)$$

where $d(\cdot, \cdot)$ is the clipped L2 distance that reduces the loss to zero if the projected pixel is within the cell.

$$d(\boldsymbol{q}_1, \boldsymbol{q}_2) = \begin{cases} \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|_2, & \text{if } \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|_\infty > l/2 \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

where $l/2$ is the half-side length of a square cell. As we clipped the distance function, our correspondence loss imposed only coarse constraints at the cell level. However, RGB loss can help refine the surface and correspondences at a fine level.

For pixels with no salient correspondence, where $\boldsymbol{u} = \inf$, we can also add constraints based on the no-correspondence loss $\mathcal{L}_{\mathrm{ncorr}}$. For each pixel with $\boldsymbol{u} = \inf$, $\boldsymbol{l}_{\boldsymbol{p}}^{t2}$ has no intersection with plane $\boldsymbol{P}$ or the intersection point is outside the grid area. Thus, the no-correspondence loss $\mathcal{L}_{\mathrm{ncorr}}$ is defined as Eq. (10).

$$\mathcal{L}_{\mathrm{ncorr}} = -\frac{1}{|\boldsymbol{M}^t|} \sum_{\boldsymbol{p}} \boldsymbol{M}_{\boldsymbol{p}}^t \hat{d}\left(\boldsymbol{v}_{\boldsymbol{l}_{\boldsymbol{p}}^{t2}}, \boldsymbol{g}\right) \qquad (10)$$

where the clipped distance $\hat{d}(\cdot, \cdot)$ is the L2 distance when the texture coordinate of the contact point is located outside the grid.

$$\hat{d}\left(\boldsymbol{q}_1, \boldsymbol{q}_2\right) = \begin{cases} \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|_2, & \text{if } \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|_\infty < 0.5 \\ 0, & \text{otherwise} \end{cases}$$

$$(11)$$

where 0.5 is the half-side length of the entire grid pattern in texture coordinates.

Silhouette loss and regularization loss. We also added silhouette loss [63] to the annotated object masks similar to the initial shape reconstruction step (Section 4.2). Moreover, to further constrain the optimization, we added a regularization loss as Eq. (12):

$$\mathcal{L}_{\mathrm{reg}} = \lambda_{\mathrm{ls}} \mathcal{L}_{\mathrm{ls}} + \lambda_{\mathrm{nc}} \mathcal{L}_{\mathrm{nc}} + \lambda_{\mathrm{pc}} \mathcal{L}_{\mathrm{pc}} \qquad (12)$$

which has three loss terms for shape regularization: a Laplacian smoothness loss $\mathcal{L}_{\mathrm{ls}}$ similar to that in Ref. [63], a normal consistency loss $\mathcal{L}_{\mathrm{nc}}$ as in Ref. [4], and a point cloud regularization loss $\mathcal{L}_{\mathrm{pc}}$ that minimizes the chamfer distance [63] between the optimized points on the current mesh and the initial shape. The losses are defined as Eqs. (13)–(15).

$$\mathcal{L}_{\mathrm{ls}} = \sum_{\boldsymbol{v}_j \in \mathcal{N}(\boldsymbol{v}_i)} \frac{1}{|\mathcal{N}(\boldsymbol{v}_i)|} (\boldsymbol{v}_j - \boldsymbol{v}_i) \qquad (13)$$

$$\mathcal{L}_{\mathrm{nc}} = \sum_{e \in \mathcal{E}} (1 - \log(1 + \boldsymbol{n}_1^e \cdot \boldsymbol{n}_2^e)) \qquad (14)$$

$$\mathcal{L}_{\mathrm{pc}} = \frac{1}{|\mathcal{S}^1|} \sum_{\boldsymbol{x}^1 \in \mathcal{S}^1} \min_{\boldsymbol{x}^2 \in \mathcal{S}^2} \|\boldsymbol{x}^1 - \boldsymbol{x}^2\|_2^2$$

$$+ \frac{1}{|\mathcal{S}^2|} \sum_{\boldsymbol{x}^2 \in \mathcal{S}^2} \min_{\boldsymbol{x}^1 \in \mathcal{S}^2} \|\boldsymbol{x}^1 - \boldsymbol{x}^2\|_2^2 \qquad (15)$$

where $\mathcal{N}(\boldsymbol{v}_i)$ denotes the neighboring vertices of vertex $\boldsymbol{v}_i$, $\mathcal{E}$ is the set of all edges, and $\boldsymbol{n}_1^e \cdot \boldsymbol{n}_2^e$ is the dot product of the normals of the two adjacent

triangles sharing the same edge e. We set $\lambda_{\mathrm{ls}} = 0.2$, $\lambda_{\mathrm{nc}} = 1.0$, and $\lambda_{\mathrm{pc}} = 100.0$.

Remark. To increase stability during optimization, we removed three types of camera rays: (1) nearly perpendicular to the surface normals at their intersection with the surface during recursive ray tracing ($|\boldsymbol{l} \cdot \boldsymbol{n}| < 0.2$); (2) traced from the pixels near the boundary of the 2D mask created by the shape projection (fewer than seven pixels); and (3) traced from pixels with a nearly overexposed color (larger than 220 in all RGB channels). After these reflection pruning operations, we found that our algorithm can produce high-quality reconstruction results without an environment map, because the remaining rays are dominated by refraction or reflection from the scene mesh.

Because the light path inside a transparent object is complex, optimizing the surface shape based on local RGB loss was not sufficient, even with our pyramid loss or perceptual loss (VGG loss). Consequently, we utilized correspondence-based loss to obtain gradients to move $\boldsymbol{l}_{\boldsymbol{p}}^{t2}$ inside the corresponding cell. Within a cell, the distance between a pixel in the rendered image and its corresponding pixel in the captured image was short. Consequently, RGB loss could improve the surface details. We verified the preceding observation through the ablation study discussed in Section 5 (with and without correspondence, and no correspondence losses).

4.4 Implementation details

Initial shape optimization. As described earlier, our initial shape reconstruction step is based on IDR [7]. We found that with only silhouette loss, an insufficient number of rays may cause holes on the surfaces or sometimes generate another surface beneath the surface of the object. Thus, we increased the number of rays sampled from an image to 20,800, and each batch contained rays sampled from three images. We set the learning rate as 1×10^{-4} at the beginning with the same decay strategy as in IDR. The initial shape reconstruction step took approximately 1–2 h after 2000 epochs on NVIDIA GeForce RTX 3090 GPU.

Surface-based MLP optimization. In the surface-based MLP optimization step, we randomly cropped 200×200 image patches from the RGB and silhouette mask images to compute the RGB and silhouette loss terms. In our implementation, we

projected the initial shape onto all views to generate the projected mask images, denoted by M^p. During optimization, the RGB image patches were sampled if they overlapped with M^p to improve patch sampling efficiency. For silhouette loss, we only sampled crops from masks annotated for the initial shape reconstruction at each iteration. The number of images with annotated masks was much smaller than the total number of captured RGB images in our implementation to ease the burden of the annotation procedure. The optimizer for surface-based local MLPs optimization was ADAM [68], and the learning rate was set to 1×10^{-5} with a cosine annealing [69] scheduler. The number of epochs used to train the network was 300. The optimization procedure implemented using PyTorch [70] took about one hour on one NVIDIA GeForce RTX 3090 GPU. The ray tracing procedure was accelerated using OptiX engine. For the index of refraction (IOR) coefficient, we fixed the coefficient η of the object as 1.52 for all three channels and ignored the dispersion.

View selection. We captured images and moved the iPad positions following the condition that for each high-curvature region not near silhouettes, the salient cells in the background pattern were refracted by this region more than twice among all images. Because we did not initially have the object geometry, the high-curvature regions were determined through our observation. The masks were adaptively selected for manual annotation using a view-selection algorithm. This iteratively added views with at least a 60-degree angle distance compared with all previously selected views. Subsequently, some extra views were needed in some cases, as determined by checking the visual hull.

5 Experiments

We applied our algorithm to reconstruct the 3D shapes of five transparent objects, as shown in Fig. 1 and Fig. 10, which were made from glass or crystals. The captured images of the five objects are shown in Fig. 9. The size of each object, the number of input images, the number of manually annotated masks, and the number of random patterns with moving frequency are listed in Table 1. In the following section, we demonstrate the advantages of our surface-based local MLP representation, perform ablation
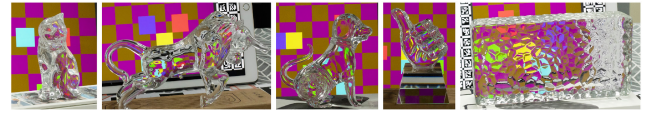

Fig. 9 Captured images for five transparent objects: the cat object, the cow object, the dog object, the trophy object, and the brick object with bumpy front surface.

Fig. 10 Reconstruction results and their corresponding rendering results for a trophy object and a brick object with bumpy front surface.

studies, and compare our method with state-of-the-art transparent object reconstruction methods.

To evaluate the accuracy of reconstruction quantitatively, we painted each object with DPT-5 developer as in Ref. [2] and then scanned it with a scanner to obtain a ground truth mesh in SI unit (m), as shown in Fig. 11. We compared the reconstructed results with the ground truth after aligning them using the ICP [71]. Subsequently, we evaluated the reconstruction by measuring the chamfer distance between the two point clouds.

5.1 Evaluations

5.1.1 Surface-based local MLP representation

We performed a comparison to verify the importance of the surface-based local MLP representation. Thus, four other shape representations were incorporated: (1) explicit mesh vertices, as in Ref. [4], denoted by Vert; (2) mesh vertices with an advanced optimizer in Ref. [66], denoted by Vert-LS (Large Step); (3) SDF encoded by a single MLP, similar to IDR [7], denoted by Refract-IDR; and (4) SDF encoded by a single MLP with explicit flows, as in Ref. [67], denoted by SDF-EF (explicit flows). All representations were optimized using the same loss functions in Section 4.3. The implementation details of the representations used in the comparison are as follows.

Vert. This representation explicitly optimizes the position of each vertex using the same loss as ours and with an ADAM optimizer. The learning rate was set to 1×10^{-5} with a cosine annealing [69] scheduler.

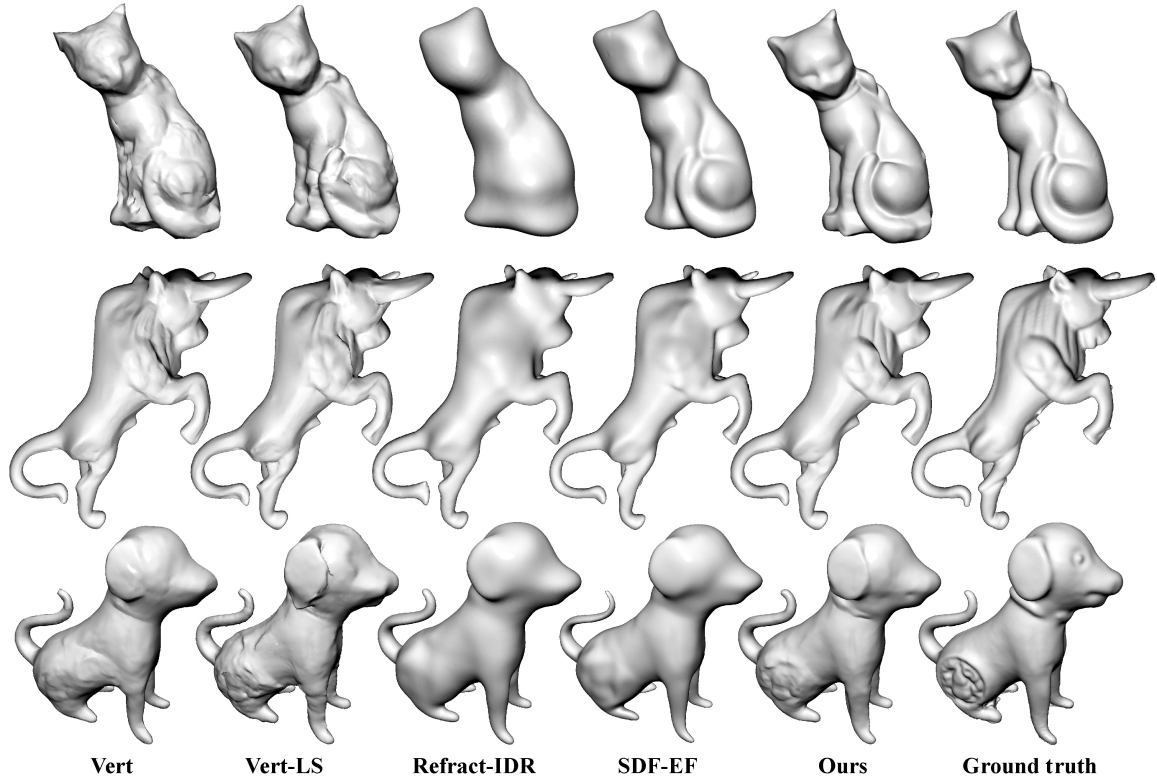| Vert | Vert-LS | Refract-IDR | SDF-EF | Ours | Ground truth |

Fig. 11   Comparisons with Vert, Vert-LS [66], Refract-IDR [7], and SDF-EF [67].

Vert-LS. This representation is similar to Vert but with the gradient calculation method and the new optimizer proposed in Ref. [66]. The gradient steps in Ref. [66] already involve the Laplacian energy. Therefore, we removed the explicit Laplacian smoothness loss. We set the diffusion weight λ to 10 and the learning rate to 2×10^{-3} as in their study.

Refract-IDR. The Refract-IDR representation is a modified version of IDR [7]. We keep the mask loss and eikonal loss the same as IDR and extend the spherical ray tracing in IDR for two-bounce refract and one-bounce reflection rays. The RGB loss in IDR was replaced by our RGB loss. We also added our corr and ncorr loss, where the weight for each loss term was the same as in our study. During optimization, the gradient of the normal for each intersection point was back-propagated to the MLP through the autograd operation. The architecture of the Refract-IDR network involves eight hidden layers, with each layer having dimensions of 256. The activation function, level of positional encoding, optimizer, and learning rate were set the same as in IDR.

SDF-EF. The SDF-EF representation is the same as in Mehta et al. [67]. It also utilizes the SDF

encoded by a single MLP to represent the surfaces. Unlike the Refract-IDR representation, it back-propagates the gradients to the MLP network based on explicit mesh vertices extracted using the marching cube algorithm [67, 72]. During the optimization, this representation performs a marching cube to extract the mesh at each iteration. The architecture of the SDF-MLP network, activation function, level of positional encoding, optimizer, and learning rate were set the same as in Refract-IDR. We also add the Laplacian smooth term, as in Ref. [67] to regularize the surface.

As shown in Fig. 11 and Table 2, our Surf-MLP representation outperformed other representations. Explicit mesh optimization introduces high-frequency artifacts. Although the Vert-LS method can reduce artifacts, it introduces folds into some areas. Simultaneously, the Refract-IDR and SDF-EF

Table 2   Chamfer distance measurement for Fig. 11

Object	Vert	Vert-LS	Refract-IDR	SDF-EF	Ours
Cat	1.010	0.792	1.397	0.978	**0.590**
Cow	1.035	1.032	1.092	1.182	**0.986**
Dog	1.568	1.398	1.472	1.423	**1.014**

representations led to slow convergence and produced over-smooth results that lost some details.

The explicit mesh optimization described above yields some artifacts because explicit optimization is sensitive to noise. However, these artifacts can be reduced by optimizing the vertices in a coarse-to-fine manner. We can progressively performed remeshing during explicit mesh optimization. We denote Vert and Vert-LS with remeshing as Vert R and Vert-LS R and compared our methods with them. Similar to Lyu et al. [4], we do remeshing after every fixed iteration. (30 epochs were used in the experiment.) At each remeshing stage: The target triangle size is fixed and decreases iteratively. As shown in Fig. 12, the remeshing procedure can increase quality of the reconstructed results compared with Vert and Vert-LS. However, our method still outperformed these two methods based on the chamfer distance between the mesh and the ground truth, as shown in Fig. 12. We believe this is because the weights of the loss terms after each remeshing stage must be tuned during optimization. If the weights are not set adaptively, the results for Vert R and the Vert-LS R may suffer from oversmoothing or local high-frequency artifact.

5.1.2 Comparison with Li et al. [5] and Lyu et al. [4]

The method proposed by Li et al. [5] needs to capture images in an environment large enough to meet the distant-illumination assumption for the environment map. Thus, we chose to evaluate this method using images obtained by rendering the ground-truth mesh illuminated by an environment map (no background geometry). Specifically, we used our ground truth mesh with one environment map to render multi-view images, corresponding object masks, and mirror sphere images, as shown in Fig. 13. The generated environmental map is shown in Fig. 13. A total of 36 images were rendered for visual hull reconstruction, and 10 of them were uniformly sampled as the input images to the network. As illustrated in Fig. 14, this method produces smooth surfaces that lose detail near the object boundaries. Compared with the ground-truth mesh, our reconstructed model can preserve relatively high-precision details. Despite the results using 10 views, we also test 20 input views in our experiments. For the method proposed by Li et al. [5], we found that adding more views to the network did not improve the details of the reconstructed surface in this experiment, as shown in Fig. 14. Here, 10 or 20 views were used as the inputs of the network. The initial visual hull is calculated using all 36 images.

For comparison with Lyu et al. [4], as they only provide ray–pixel correspondences and masks in their

Fig. 12 Comparisons with Vert R and Vert-LS R [66]. The numbers indicate the chamfer distances between the mesh and the ground truth illustrated in Fig. 11.

Fig. 13 Input images for Li et al. [5]. The mirror sphere images are used in Li et al. [5] to generate the environment map.
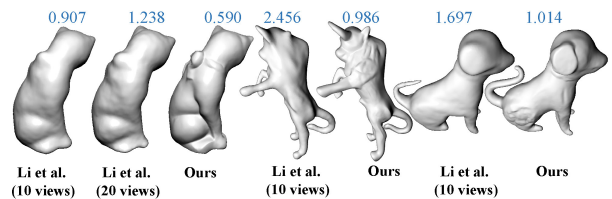

Fig. 14 Comparisons with Li et al. [5]. The numbers indicate the chamfer distances between the mesh and the ground truth illustrated in Fig. 11.

dataset, we replace our RGB loss, corr loss, and ncorr loss with fraction loss [4]. As illustrated in Fig. 15, our reconstructed model can preserve more surface detail. Qualitative and quantitative comparison results are shown in Fig. 15 and Table 3. The "mask diff" is the mean L1 distance between rendered masks and the input masks. As shown in the fifth column of Table 3, the ground-truth mesh is not precisely consistent with the input masks. This may be due to the DPT-5 developer coating used in the scanning. As shown in Table 3, the chamfer distances of our results are slightly larger than those of this method. However, our mask difference is smaller, which means that our reconstructed model matches the input masks better. In this experiment, we added the mask difference measurement to demonstrate the advantage of our method because the silhouettes obtained by Lyu et al. [4] are much more accurate than those of the segmentation results.

5.1.3 Number of clusters

We first performed ablation studies to evaluate the influence of the cluster number (MLP number). Figure 16 shows that our surface-based local MLP representation can obtain better results by increasing the number of clusters from 1, 50, 100, and 150. We also compared our local MLP representation to a global MLP with nine hidden layers of 256 dimensions. For the global MLP, we also test it

Fig. 15 Comparisons with Lyu et al. [4] using all 72 views as input. Our results preserve more surface details (indicated by the red arrow). The chamfer distances and mask differences are shown in Table 2.

Table 3 Chamfer distance and mask difference measurement for Fig. 15

Object	Measurement	Lyu et al.	Ours	Ground truth
Horse	chamfer dist↓	**0.003654**	0.004183	N/A
	mask diff↓	0.003135	**0.002961**	0.005236
Tiger	chamfer dist↓	**0.005122**	0.005438	N/A
	mask diff↓	0.004571	**0.004497**	0.007975

with two different positional encoding settings, using the number of positional encoding frequencies $L = 6$ and $L = 16$. As illustrated in Fig. 16, our multiple local MLP representation outperformed the single global MLP representation with a lower chamfer distance, despite the number of positional encoding frequencies. This experiment also shows that our local MLP representation can accelerate convergence: it can achieve a lower chamfer distance with the same number of epochs. We believe that this is because all the vertices share the MLP weights in the global MLP representation, which means VDF in distant clusters will affect each other during training.

5.2 Ablation study

We remove each loss term individually to evaluate its impact on the reconstruction result of the cat object, as shown in Fig. 17. Overall, the correspondence loss $\mathcal{L}_{\text{corr}}$, silhouette loss $\mathcal{L}_{\text{sil}}$, and normal consistency loss $\mathcal{L}_{\text{nc}}$ were the most important for our method. As shown in the first row in Fig. 17, $\mathcal{L}_{\text{corr}}$ is essential for our method to obtain surface details. $\mathcal{L}_{\text{sil}}$ can help preserve boundaries, such as the ears of the cat. The no-correspondence loss $\mathcal{L}_{\text{ncorr}}$ can also improve the result, which implies lower chamfer distances. However, because $\mathcal{L}_{\text{ncorr}}$ only constrains some intersection points outside the grid area, it cannot help too much for surface reconstruction.

In the second row of Fig. 17, we demonstrate the purpose of the RGB loss $\mathcal{L}_{\text{rgb}}$. The first two images were rendered using recursive ray tracing. The third image shows the corresponding patch of the input image. Our correspondence-based loss method can
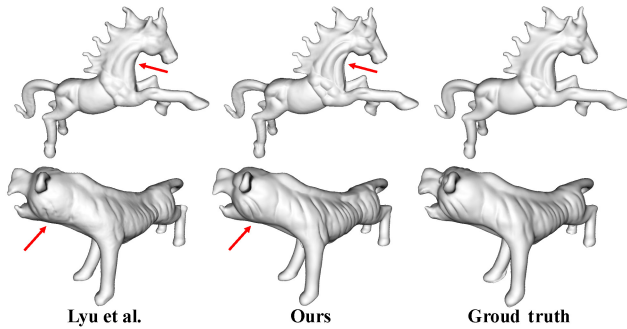

Arch	**#MLP 1**	**#MLP 50**	**#MLP 100**	**#MLP 150**
PE_6 256_9	1.113			
PE_16 256_9	1.251			
PE_16 128_2	1.410	0.686	0.676	**0.590**

Metric: Chamfer-distance

Fig. 16 Influence of MLP number (#MLP) and the MLP architecture measured by the chamfer distance between reconstructed mesh and ground truth mesh. PE_k/d_l: positional encoding $L = k$, MLP with l layer of d dimension. All the results are obtained after training networks with 300 epochs. Back slash means the training cannot be performed due to computational cost.
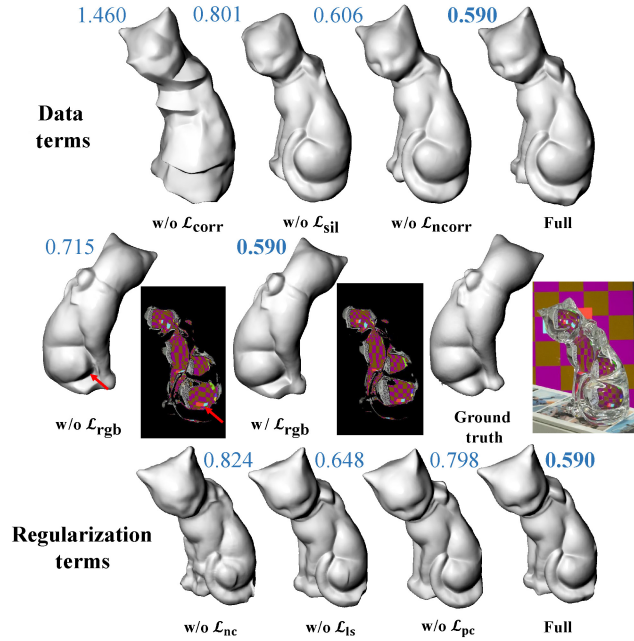
Fig. 17 Ablation study of loss terms on the cat object. The numbers indicate the chamfer distances between the reconstructed mesh and the ground truth mesh.

optimize the surface to move the refracted rays toward their corresponding cells. Then, as indicated by the red arrow, RGB loss $\mathcal{L}_{\text{rgb}}$ can constrain the refracted rays near the cells by minimizing per-pixel color differences. By utilizing the gradient in each image's RGB space, $\mathcal{L}_{\text{rgb}}$ can fix small geometric inaccuracies and improve the surface details, as indicated by the first red arrow. We also test the purpose of each regularization term. The results are presented in the last row of Fig. 17. We can see that our method can obtain relatively good results without Laplacian smoothness loss $\mathcal{L}_{\text{ls}}$ and point cloud regularization loss $\mathcal{L}_{\text{pc}}$. However, $\mathcal{L}_{\text{ls}}$ can further improve surface smoothness, as shown in Fig. 17. In our experiments, we found $\mathcal{L}_{\text{pc}}$ can improve the stability of our patch-based optimization.

6   Limitations and future work

Our environment-matting algorithm can only find correspondences, assuming that the transparent object has no intrinsic color. Consequently, our method cannot reconstruct colored transparent objects. Next, as the ray–cell correspondences are sparse, our method requires more views to reconstruct the surface and may miss some details, especially for surfaces with complex occlusions. Another limitation

is that we need to manually annotate the object masks. In the future, it would be interesting to investigate how to integrate the variables for the color or other material properties of the transparent object to overcome the "no intrinsic color" limitation and how to extract accurate transparent object masks based on single-image EnvMatt.

7   Conclusions

In this study, we developed a method to reconstruct 3D shapes of transparent objects from handheld captured images under natural light conditions. Our method comprises two components: a surface-based MLP representation that encodes the vertex displacement field based on the initial shape, and a surface optimization through differentiable rendering and EnvMatt. We used an iPad as a background to provide ray–cell correspondences, a simplified capture setting, to facilitate the optimization. Our method can produce high-quality reconstruction results with fine details under natural lighting conditions.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Kutulakos, K. N.; Steger, E. A theory of refractive and specular 3D shape by light-path triangulation. *International Journal of Computer Vision* Vol. 76, No. 1, 13–29, 2008.

[2] Wu, B.; Zhou, Y.; Qian, Y.; Cong, M.; Huang, H. Full 3D reconstruction of transparent objects. *ACM Transactions on Graphics* Vol. 37, No. 4, Article No. 103, 2018.

[3] Ihrke, I.; Kutulakos, K. N.; Lensch, H. P. A.; Magnor, M.; Heidrich, W. Transparent and specular object reconstruction. *Computer Graphics Forum* Vol. 29, No. 8, 2400–2426, 2010.

[4] Lyu, J.; Wu, B.; Lischinski, D.; Cohen-Or, D.; Huang, H. Differentiable refraction-tracing for mesh reconstruction of transparent objects. *ACM Transactions on Graphics* Vol. 39, No. 6, Article No. 195, 2020.

[5] Li, Z.; Yeh, Y. Y.; Chandraker, M. Through the looking glass: Neural 3D reconstruction of transparent shapes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1259–1268, 2020.

[6] Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. In: *Computer Vision – ECCV 2020. Lecture Notes in Computer Science, Vol. 12346.* Vedaldi, A.; Bischof, H.; Brox, T.; Frahm, J. M. Eds. Springer Cham, 405–421, 2020.

[7] Yariv, L.; Kasten, Y.; Moran, D.; Galun, M.; Atzmon, M.; Basri, R.; Lipman, Y. Multiview neural surface reconstruction by disentangling geometry and appearance. *arXiv preprint* arXiv:2003.09852, 2020.

[8] Oechsle, M.; Peng, S.; Geiger, A. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 5569–5579, 2021.

[9] Chen, A.; Wu, M.; Zhang, Y.; Li, N.; Lu, J.; Gao, S.; Yu, J. Deep surface light fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* Vol. 1, No. 1, Article No. 14, 2018.

[10] Cui, Z.; Gu, J.; Shi, B.; Tan, P.; Kautz, J. Polarimetric multi-view stereo. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 369–378, 2017.

[11] Huynh, C. P.; Robles-Kelly, A.; Hancock, E. Shape and refractive index recovery from single-view polarisation images. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1229–1236, 2010.

[12] Miyazaki, D.; Ikeuchi, K. Inverse polarization raytracing: Estimating surface shapes of transparent objects. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 910–917, 2005.

[13] Tanaka, K.; Mukaigawa, Y.; Kubo, H.; Matsushita, Y.; Yagi, Y. Recovering transparent shape from time-of-flight distortion. In: Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition, 4387–4395, 2016.

[14] Trifonov, B.; Bradley, D.; Heidrich, W. Tomographic reconstruction of transparent objects. In: Proceedings of the ACM SIGGRAPH Sketches, 55–es, 2006.

[15] Chen, T.; Goesele, M.; Seidel, H. P. Mesostructure from specularity. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1825–1832, 2006.

[16] Morris, N. J. W.; Kutulakos, K. N. Reconstructing the surface of inhomogeneous transparent scenes by scatter-trace photography. In: Proceedings of the IEEE 11th International Conference on Computer Vision, 1–8, 2007.

[17] Wetzstein, G.; Roodnick, D.; Heidrich, W.; Raskar, R. Refractive shape from light field distortion. In: Proceedings of the International Conference on Computer Vision, 1180–1186, 2011.

[18] Schwartzburg, Y.; Testuz, R.; Tagliasacchi, A.; Pauly, M. High-contrast computational caustic design. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 74, 2014.

[19] Shan, Q.; Agarwal, S.; Curless, B. Refractive height fields from single and multiple images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 286–293, 2012.

[20] Yue, Y.; Iwasaki, K.; Chen, B. Y.; Dobashi, Y.; Nishita, T. Poisson-based continuous surface generation for goal-based caustics. *ACM Transactions on Graphics* Vol. 33, No. 3, Article No. 31, 2014.

[21] Morris, N. J. W.; Kutulakos, K. N. Dynamic refraction stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 33, No. 8, 1518–1531, 2011.

[22] Qian, Y.; Gong, M.; Yang, Y. H. Stereo-based 3D reconstruction of dynamic fluid surfaces by global optimization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 6650–6659, 2017.

[23] Zhang, M.; Lin, X.; Gupta, M.; Suo, J.; Dai, Q. Recovering scene geometry under wavy fluid via distortion and defocus analysis. In: *Computer Vision – ECCV 2014. Lecture Notes in Computer Science, Vol. 8693.* Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. Eds. Springer Cham, 234–250, 2014.

[24] Tsai, C. Y.; Veeraraghavan, A.; Sankaranarayanan, A. C. What does a single light-ray reveal about a transparent object? In: Proceedings of the IEEE International Conference on Image Processing, 606–610, 2015.

[25] Chari, V.; Sturm, P. A theory of refractive photo-light-path triangulation. In: Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition, 1438–1445, 2013.

[26] Qian, Y.; Gong, M.; Yang, Y. H. 3D reconstruction of transparent objects with position-normal consistency. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4369–4377, 2016.

[27] Zongker, D. E.; Werner, D. M.; Curless, B.; Salesin, D. H. Environment matting and compositing. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, 205–214, 1999.

[28] Stets, J.; Li, Z.; Frisvad, J. R.; Chandraker, M. Single-shot analysis of refractive shape using convolutional neural networks. In: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, 995–1003, 2019.

[29] Sajjan, S.; Moore, M.; Pan, M.; Nagaraja, G.; Lee, J.; Zeng, A.; Song, S. Clear grasp: 3D shape estimation of transparent objects for manipulation. In: Proceedings of the IEEE International Conference on Robotics and Automation, 3634–3642, 2020.

[30] Loper, M. M.; Black, M. J. OpenDR: An approximate differentiable renderer. In: *Computer Vision – ECCV 2014. Lecture Notes in Computer Science, Vol. 8695.* Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. Eds. Springer Cham, 154–169, 2014.

[31] Kato, H.; Ushiku, Y.; Harada, T. Neural 3D mesh renderer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3907–3916, 2018.

[32] Li, T. M.; Aittala, M.; Durand, F.; Lehtinen, J. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Transactions on Graphics* Vol. 37, No. 6, Article No. 222, 2018.

[33] Liu, S.; Chen, W.; Li, T.; Li, H. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 7707–7716, 2019.

[34] Laine, S.; Hellsten, J.; Karras, T.; Seol, Y.; Lehtinen, J.; Aila, T. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics* Vol. 39, No. 6, Article No. 194, 2020.

[35] Wang, P.; Liu, L.; Liu, Y.; Theobalt, C.; Komura, T.; Wang, W. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In: Proceedings of the 35th International Conference on Neural Information Processing Systems, Article No. 2081, 27171–27183, 2024.

[36] Yariv, L.; Gu, J.; Kasten, Y.; Lipman, Y. Volume rendering of neural implicit surfaces. In: Proceedings of the 35th International Conference on Neural

Information Processing Systems, Article No. 367, 4805–4815, 2024.

[37] Nimier-David, M.; Vicini, D.; Zeltner, T.; Jakob, W. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics* Vol. 38, No. 6, Article No. 203, 2019.

[38] Li, T. M.; Aittala, M.; Durand, F.; Lehtinen, J. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Transactions on Graphics* Vol. 37, No. 6, Article No. 222, 2018.

[39] Li, T. M.; Lehtinen, J.; Ramamoorthi, R.; Jakob, W.; Durand, F. Anisotropic Gaussian mutations for metropolis light transport through Hessian-Hamiltonian dynamics. *ACM Transactions on Graphics* Vol. 34, No. 6, Article No. 209, 2015.

[40] Luan, F.; Zhao, S.; Bala, K.; Gkioulekas, I. Langevin Monte Carlo rendering with gradient-based adaptation. *ACM Transactions on Graphics* Vol. 39, No. 4, Article No. 140, 2020.

[41] Zhang, C.; Miller, B.; Yan, K.; Gkioulekas, I.; Zhao, S. Path-space differentiable rendering. *ACM Transactions on Graphics* Vol. 39, No. 4, Article No. 143, 2020.

[42] Zhang, C.; Wu, L.; Zheng, C.; Gkioulekas, I.; Ramamoorthi, R.; Zhao, S. A differential theory of radiative transfer. *ACM Transactions on Graphics* Vol. 38, No. 6, Article No. 227, 2019.

[43] Bangaru, S. P.; Li, T. M.; Durand, F. Unbiased warped-area sampling for differentiable rendering. *ACM Transactions on Graphics* Vol. 39, No. 6, Article No. 245, 2020.

[44] Mescheder, L.; Oechsle, M.; Niemeyer, M.; Nowozin, S.; Geiger, A. Occupancy networks: Learning 3D reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4455–4465, 2019.

[45] Chen, Z.; Zhang, H. Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5932–5941, 2019.

[46] Park, J. J.; Florence, P.; Straub, J.; Newcombe, R.; Lovegrove, S. DeepSDF: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 165–174, 2019.

[47] Atzmon, M.; Lipman, Y. SAL: Sign agnostic learning of shapes from raw data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2562–2571, 2020.

[48] Porter, T.; Duff, T. Compositing digital images. *ACM SIGGRAPH Computer Graphics* Vol. 18, No. 3, 253–259, 1984.

[49] Levin, A.; Lischinski, D.; Weiss, Y. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 30, No. 2, 228–242, 2008.

[50] Chuang, Y. Y.; Zongker, D. E.; Hindorff, J.; Curless, B.; Salesin, D. H.; Szeliski, R. Environment matting extensions: Towards higher accuracy and real-time capture. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, 121–130, 2000.

[51] Peers, P.; Dutré, P. Wavelet environment matting. In: Proceedings of the 14th Eurographics Workshop on Rendering, 157–166, 2003.

[52] Qian, Y.; Gong, M.; Yang, Y. H. Frequency-based environment matting by compressive sensing. In: Proceedings of the IEEE International Conference on Computer Vision, 3532–3540, 2015.

[53] Duan, Q.; Cai, J.; Zheng, J. Compressive environment matting. *The Visual Computer* Vol. 31, No. 12, 1587–1600, 2015.

[54] Chen, G.; Han, K.; Wong, K. Y K. TOM-net: Learning transparent object matting from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9233–9241, 2018.

[55] Wexler, Y.; Fitzgibbon, A. W.; Zisserman, A. Image-based environment matting. In: Proceedings of the ACM SIGGRAPH Conference Abstracts and Applications, 198, 2002.

[56] Kutulakos, K. N.; Seitz, S. M. A theory of shape by space carving. *International Journal of Computer Vision* Vol. 38, No. 3, 199–218, 2000.

[57] Cohen-Steiner, D.; Alliez, P.; Desbrun, M. Variational shape approximation. In: Proceedings of the ACM SIGGRAPH Papers, 905–914, 2004.

[58] Crane, K.; Weischedel, C.; Wardetzky, M. The heat method for distance computation. *Communications of the ACM* Vol. 60, No. 11, 90–99, 2017.

[59] Li, T. M. Differentiable visual computing. *arXiv preprint* arXiv:1904.12228, 2019.

[60] Olson, E. AprilTag: A robust and flexible visual fiducial system. In: Proceedings of the IEEE International Conference on Robotics and Automation, 3400–3407, 2011.

[61] CapturingReality. Reality capture. 2016. Available at http://capturingreality.com

[62] Schnabel, R.; Wahl, R.; Klein, R. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum* Vol. 26, No. 2, 214–226, 2007.

[63] Ravi, N.; Reizenstein, J.; Novotny, D.; Gordon, T.; Lo, W. Y.; Johnson, J.; Gkioxari, G. Accelerating 3D deep learning with PyTorch3D. *arXiv preprint* arXiv:2007.08501, 2020.

[64] Born, M.; Wolf, E.; Hecht, E. Principles of optics: Electromagnetic theory of propagation, interference and diffraction of light. *Physics Today* Vol. 53, No. 10, 77–78, 2000.

[65] Zitová, B.; Flusser, J. Image registration methods: A survey. *Image and Vision Computing* Vol. 21, No. 11, 977–1000, 2003.

[66] Nicolet, B.; Jacobson, A.; Jakob, W. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics* Vol. 40, No. 6, Article No. 248, 2021.

[67] Mehta, I.; Chandraker, M.; Ramamoorthi, R. A level set theory for neural implicit evolution under explicit flows. In: *Computer Vision – ECCV 2022. Lecture Notes in Computer Science, Vol. 13662.* Avidan, S.; Brostow, G.; Cissé, M.; Farinella, G. M.; Hassner, T. Eds. Springer Cham, 711–729, 2022.

[68] Kingma, D. P.; Ba, J.; Hammad, M. M. Adam: A method for stochastic optimization. *arXiv preprint* arXiv:1412.6980, 2014.

[69] Loshchilov, I.; Hutter, F. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint* arXiv:1608.03983, 2016.

[70] Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in PyTorch. In: Proceedings of the 31st Conference on Neural Information Processing Systems, 2017.

[71] Zhou, Q. Y.; Park, J.; Koltun, V. Open3D: A modern library for 3D data processing. *arXiv preprint* arXiv:1801.09847, 2018.

[72] Remelli, E.; Lukoianov, A.; Richter, S. R.; Guillard, B.; Bagautdinov, T.; Baque, P.; Fua, P. MeshSDF: Differentiable iso-surface extraction. *arXiv preprint* arXiv:2006.03997, 2020.

Jiamin Xu received his Ph.D. degree in computer graphics from Zhejiang University in 2022, advised by Professor Hujun Bao and Professor Weiwei Xu. He is currently with the College of Computer Science and Technology, Hangzhou Dianzi University, China. His research interests include 3D reconstruction, image-based rendering, and neural rendering.

Zihan Zhu is currently a direct doctorate student in the Department of Computer Science, ETH Zürich. He received his B.S. degree in computer science from Zhejiang University in 2022. His research interests lie in the intersection between computer vision and computer graphics.

Hujun Bao received his B.Sc. degree in mathematics from Zhejiang University, China, in 1987, and his Ph.D. degree in applied mathematics from Zhejiang University, China, in 1993. He is currently a professor with the State Key Laboratory of CAD&CG and College of Computer Science and Technology, Zhejiang University, China.

Weiwei Xu is currently a professor at the State Key Lab of CAD&CG in Zhejiang University. He was a Qianjiang Professor at Hangzhou Normal University and a researcher in the Internet Graphics Group at Microsoft Research Asia from 2005 to 2012. He was a post-doc researcher at Ritsmeikan University in Japan for over one year. He received his Ph.D. degree in computer graphics from Zhejiang University, and B.S. and master degrees in computer science from Hohai University in 1996 and 1999, respectively.