




Novel View Synthesis Of Transparent Object From a Single Image

Shizhe Zhou,  Zezu Wang and Dongwei Ye

College of Computer Science and Electronic Engineering, Hunan University, Changsha, China

Abstract

We propose a method for converting a single image of a transparent object into multi-view photo that enables users observing the object from multiple new angles, without inputting any 3D shape. The complex light paths formed by refraction and reflection makes it challenging to compute the lighting effects of transparent objects from a new angle. We construct an encoder–decoder network for normal reconstruction and texture extraction, which enables synthesizing novel views of transparent object from a set of new views and new environment maps using only one RGB image. By simultaneously considering the optical transmission and perspective variation, our network learns the characteristics of optical transmission and the change of perspective as guidance to the conversion from RGB colours to surface normals. A texture extraction subnetwork is proposed to alleviate the contour loss phenomenon during normal map generation. We test our method using 3D objects within and without our training data, including real 3D objects that exists in our lab, and completely new environment maps that we take using our phones. The results show that our method performs better on view synthesis of transparent objects in complex scenes using only a single-view image.

Keywords: image and video processing, view independent and 3D video, rendering, texture synthesis

CCS Concepts: • Picture/Image Generation → View Synthesis; Transparent Object

1. Introduction

Transparent objects are studied in both computer graphics and computer vision such as segmentation [CHW18, KTR*20], recognition [MNSiT13, CWY*20] and 3D reconstruction [HWL17, LYC20]. In this work, we propose a system for new view synthesis (NVS) of transparent objects. NVS generates images of a given object from an unknown perspective [WGSJ20, MST*20, XBS*19] which enables many applications in 3D modelling, Augmented Reality, image editing and 3D RGB visions. However, most of the existing view synthesis methods are designed for opaque objects. Our system is specifically designed for transparent objects. According to Snell’s law, when the viewing angle changes, the direction of the refracted light on transparent objects also changes which introduces pixels from unseen background locations onto the object surface. This fundamental difference against opaque object makes transparent object NVS a difficult problem.

Our goal is to synthesize novel views of transparent objects in complex and real scenes. To this aim, we propose an encoder–decoder network based on normal reconstruction and texture extraction to solve the problem of view synthesis from a single transparent object image. Our method is useful for creating realistic images with

parallax display. Figure 1 shows some outputs of our method. The technical contributions of our work are fourfold:

- Our work is the first one to synthesize novel views of transparent object using only a single input RGB image. Both the perspectives and background environment can be changed.
- We construct a complex deep neural network that simulates the effects of optical transmission and perspective transformation from new view angles. The network learns the characteristics of optical transmission and perspective conversion from the transfer mappings of both RGB-to-normal and normal-to-normal due to the change of perspectives through an encoder–decoder network.
- We specifically design a texture extractor sub-network based on VGG19[SZ14] to improve the quality of the generated normal maps by reducing contour and line feature losses of the foreground 3D transparent objects. The extracted texture features are evidently helpful for obtaining a better normal map of transparent objects.
- We build a novel dataset of multi-view images of various transparent objects in complex daily scenes. This dataset contains the rendered images of many 3D objects with various shapes from a series of varying viewpoints. Taking a certain fixed line of sight as

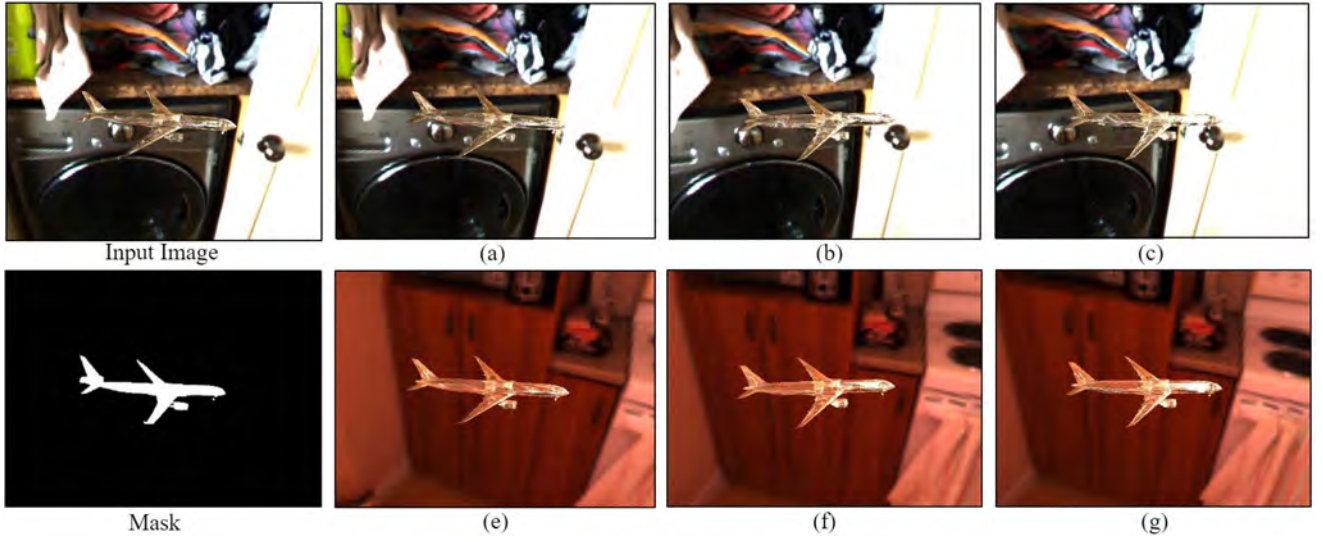


Figure 1: We propose a method to synthesize a sequence of new views of a transparent object given a single image of this object as input. We construct an encoder-decoder networks to achieve image-based normal reconstruction and texture extraction. Given an input image, our approach can generate new images of transparent object with new perspectives (a)–(c) and new environment maps views (e)–(g), given its corresponding object mask.

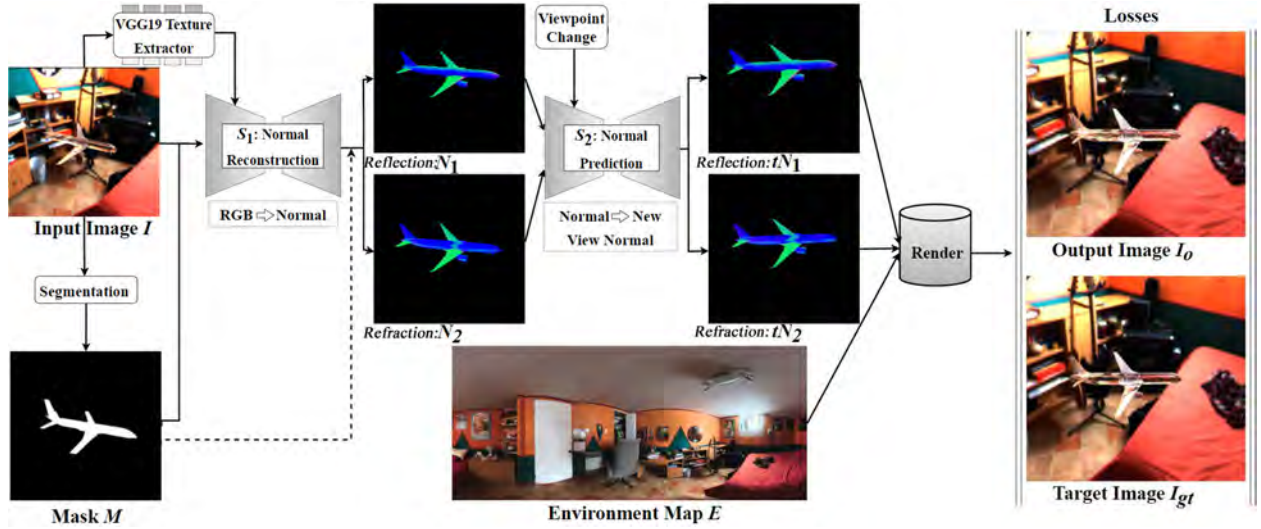


Figure 2: Our framework for view synthesis of transparent object. The whole method is divided into three steps: S_1 normal reconstruction, S_2 normal prediction and rendering.

the axis, the range of the angles is from -9° to 9° , interval being 1° .

2. Related Work

NVS is a long-standing problem at the intersection of computer graphics and computer vision. Main differences between existing works are whether they input multiple perspectives or a single perspective image, and whether they have additional 3D features such

as depth maps, normal maps or other 2D/3D semantic information. There exists four types of methods on this problem.

Multi-view stereo. The 1st type of methods use multiple perspectives of a 3D scene to reconstruct its 3D model, and then generate a new view from a given viewpoint [FWZ05, MST*20, KLS*13]. The advantage of these methods is that more viewing angles reduce the occluded area and simplify the difficulty of 3D modelling. DNNs are trained to learn depth or other 3D semantic information to generate a target 3D model [KLS*13, STB*19]. Li *et al.* [LYC20] learns

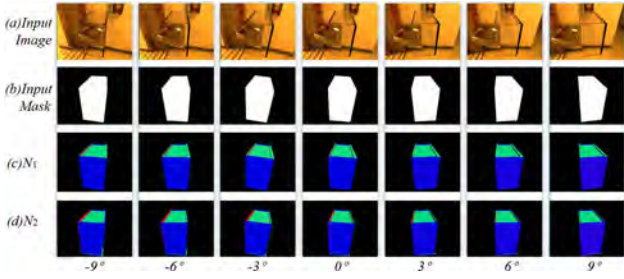


Figure 7: The illustration of our dataset.

Table 1: Image quality and normal accuracy comparison of our network trained with 2-Bounce 3D shapes only against with the full dataset. The data here are statistic over all 2-Bounce 3D shapes.

Methods	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	$tN_1\text{mean}(\circ)\downarrow$	$tN_2\text{mean}(\circ)\downarrow$
Ours	19.4330	0.9392	0.0538	5.9277	8.2884
2-Bounce	18.9217	0.9386	0.0557	5.9793	8.6447

normal or the refraction–reflection flow, i.e., the relationship between light and pixels in the scene [WGSJ20, ZTS*16, XYL*19, HPP*18, WGL*18]. Wiles *et al.* [WGSJ20] input a single scene image into an encoder–decoder network to learn the characteristics of high-resolution point clouds to represent the 3D scene structure. In order to generate a new view from the point cloud, a high-performance distinguishable point cloud renderer is used to render from the target view. Zhou *et al.* [ZTS*16] estimate the target’s refraction–reflection flow and attenuation function from a given perspective based on optical flows, so as to render a new perspective scene. The theory of these methods is suitable for NVS of complex scenes with transparent objects.

3. Method

Overview. The overall framework of our method is shown in Figure 2. Our goal is to automatically generate novel views of transparent objects in complex scenes given a single-view RGB image of the objects. The first step is to use the input single view image to reconstruct its surface normal map, shown as S_1 in Figure 2. Here we assume that the object segmentation mask is given manually by the users or using existing techniques. Aiming at the problem of serious contour loss in convolutional neural networks, we propose a VGG19[SZ14]-based texture extractor to optimize the contour loss during normal map generation, and use the extracted texture features as the input to the previous step to obtain the better quality of the transparent object normal map. In the second step shown as S_2 in Figure 2, because the transparent object changes its complex optical path with the viewing in a highly non-linear way, we learn the light transmission features between the new viewpoint and the normal map through the neural network. In the third step, we calculate the refracted and reflected light from the previously estimated normal map according to Snell’s law, and then calculate the incoming radiance of any complex environment map through bilinear sampling.

A simple local computation is executed to simulate the refraction and reflection of transparent objects in the rendering process.

3.1. Normal reconstruction

Due to the complex optical transmission behaviours of transparent objects, the network needs to learn the reflection and refraction characteristics and contour features of transparent objects. We propose an encoder–decoder network for reconstructing normal map, which only uses a single RGB image to obtain the complex light transmission effects of transparent objects, without performing complex and time-consuming ray tracing. In other words, after accurately reconstructing the normal map, we can directly calculate the direction of the refracted light and the reflected light after the incident light passes through the transparent object. The network architecture is shown in Figure 3. The input of our network is an image I of a transparent object with known refractive index (IoR), and the output is two normal maps N_1 and N_2 (They are the normals of two contact points P_1 and P_2 on which the incident light pass through a transparent object, see Figure 4). Given the corresponding segmentation mask M , we use the ground truth of N_1 , N_2 and M for supervision (denoted as N_1^{gt} , N_2^{gt} , M). The encoder–decoder network estimates

$$N_1, N_2 = S_1(I, M) \quad (1)$$

The loss function L_{in} is defined as the L_2 loss between N_1 and N_2 :

$$L_{in} = \|N_1^{gt} - N_1\|_2^2 + \|N_2^{gt} - N_2\|_2^2. \quad (2)$$

3.2. Texture extractor

Because the spatial transformation invariance of CNN makes it insensitive to edge details, it is difficult to capture the detailed information of the normal map. This often results in the significant loss of the details of the normal map especially at the contours and silhouette lines of the object[CPK*17]. Aiming at this problem, we refer to the idea of Refs. [GEB15, ZWLQ19] and propose a VGG19[SZ14]-based texture feature extractor to effectively alleviate the contour loss during normal map generation, and use the extracted texture features as the input of the previous stage, producing normal maps of transparent objects with much better quality. The principle of our texture extractor is to fully exploit the object’s texture detail in the input image I to compensate for the detailed information about the generated normal map. We use the pre-trained VGG19 model proposed by Pytorch[PGM*19] to learn the texture features (denoted as TF) of the corresponding transparent objects in Section 3.1, and combine image I and mask M to send them to the encoding layer, to convert them into the latent feature space. The features after encoding are used as the input of the decoder to predict normal maps. With the texture extractor, our normal reconstruction can be reformulated as

$$N_1, N_2 = S_1(I, M, TF) \quad (3)$$

$$\text{where } TF = VGG19(I, M)$$

3.3. Normals and perspective changes

The key to S_2 is to learn the relationship between normals and perspective changes in deep features through neural networks.

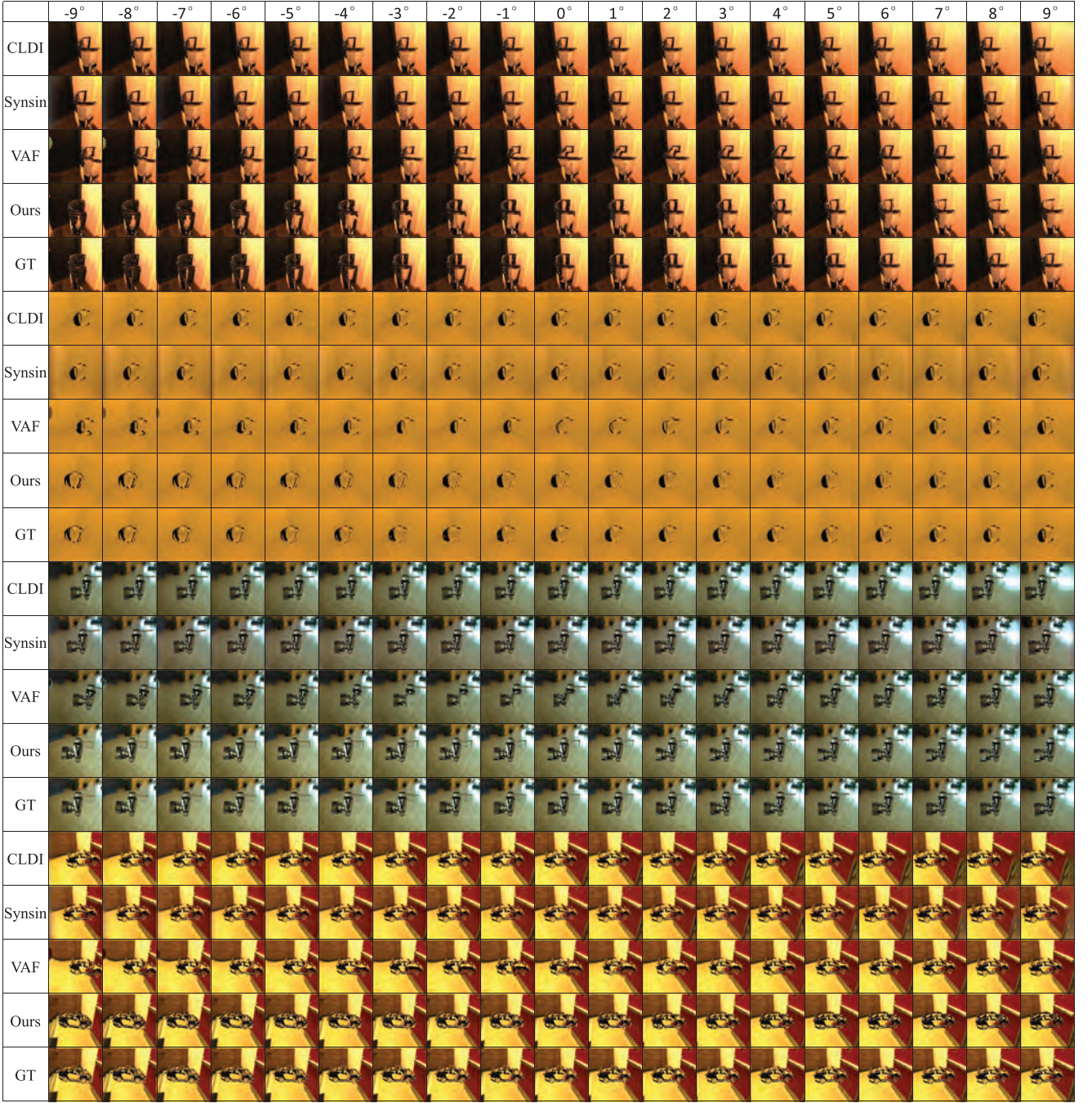


Figure 8: Visual comparison of our results with CLDI[SSKH20], Synsin[WGSJ20] and VAF[ZTS*16]. GT means ground truth. Frame 0° is the input view. All methods use the same camera parameter and input view. In general, our results have more accurate lighting changes and texture details which are similar to the ground truth. Please refer to our webpage to see more comparison results in GIF format.

Existing works [SSKH20, ZTS*16] for view synthesis often directly operate in the image pixel space, using techniques such as image inpainting, without recomputing the actual lighting effects of the object. These methods can not be applied onto transparent objects due to the complex light path formed by the refraction and reflection. For our cases, it is difficult to directly estimate the

normal map of the new viewpoint from the RGB colour space, which is essential for generating realistic high-quality scene images of the new viewpoint. Therefore, we do not directly learn the mapping from the feature changes of the image I to the normal map of the new viewpoint. Instead, we first estimate the normal map of the transparent object under the initial viewpoint, and then learn

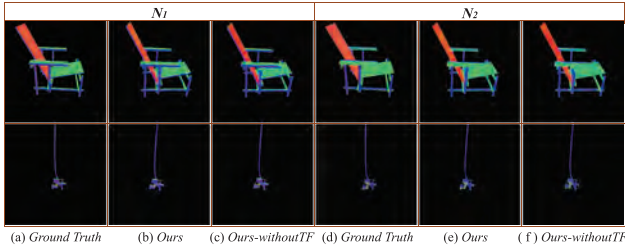


Figure 9: Visual comparison with or without texture extractor. (c) and (f) shows normal maps generated without extracting the texture feature in the network. The texture extractor lead to better visual quality and more accurate normal map details.

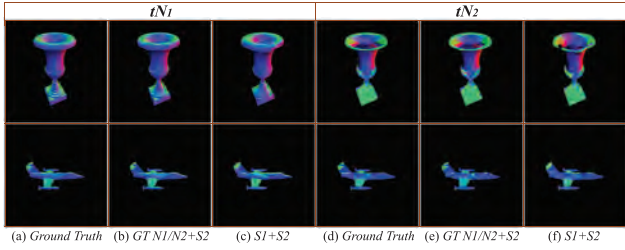


Figure 10: Visual comparison of generated normal maps tN_1/tN_2 using our whole network $S1 + S2$ against using the ground truth N_1/N_2 and $S2$ step only.

the mapping between normal maps with regard to the changes of perspectives.

We define viewpoint data as a 3×3 matrix, where the 1st, 2nd and the 3rd row are the eye position Y , centre point C and an up vector U , respectively. C is the centre point of the 3D object. Q is on an observation circle lying approximately parallel to the ground, whose centre is C and $radius = 3$. U is perpendicular to the plane circle Q belongs to. We sample 19 eye positions locating uniformly on circle Q from -9° to 9° to synthesize 19 continuous new views.

Given an initial view point v_s of the image I and a target view point v_t , we need to generate a synthesized image of transparent objects from the target view point v_t . We use a convolutional network layer to learn the spatial change features between v_s and v_t , and combine them with the features of the predicted normal maps N_1 and N_2 learned by the encoder layer of in Section 3.1. In detail, we concatenate the input v_s and the target v_t to be a 6×3 viewpoint change matrix, and convolute it to be a channel-64 layer and then

concatenate it with the 2nd normal feature layer to be a channel-128 mixed feature layer. For all of our training data, we set v_s is from the view of 0° . Different from VAF [ZTS*16], we input this viewpoint change at a relatively earlier stage. The mixed features are used as the input of the decoder to predict the normal maps tN_1 and tN_2 of the new view. We use the ground-truth tN_1^{gt} , tN_2^{gt} shape for supervision of tN_1 and tN_2 . The network architecture in this section is similar to the network architecture based on Section 3.1.

$$tN_1, tN_2 = S_2(N_1, N_2, v_s, v_t) \quad (4)$$

The loss function L_{nn} is the L_2 loss for N_1 and N_2 .

$$L_{nn} = ||tN_1^{gt} - tN_1||_2^2 + ||tN_2^{gt} - tN_2||_2^2 \quad (5)$$

3.4. Rendering layer

We use the rendering module proposed by Refs. [LSR*20, LYC20] to achieve the rendering of a new view of transparent objects, and simulate the refraction and reflection of transparent objects in the rendering process through a non-iterative local computation. Given a new viewpoint v_t and a scene environment map E (an arbitrary known and distant environment map). First as shown in Figure 4, the refracted and reflected rays r_1, r_2 are calculated from the previously estimated normal map according to Snell's law, and then r_1, r_2 are converted from the camera coordinate system to the world coordinate system according to the new viewpoint. Second, the incoming radiance r of the environment map E is calculated via bilinear sampling to obtain the pixel value. After the camera calibration, a reflection layer I_1 and refraction layer I_2 of the new views are generated based on the incoming light direction. Due to the occurrence of total internal reflection, some light may not reach the environment map after it bounces again after entering the transparent object. All of these unreachable pixels are recorded by the rendering layer onto a binary mask M_{err} , which is discarded since we do not consider any total internal reflection. The rendering layer obtain a new view image I_0 of the transparent object by adding reflection and refraction.

$$I_1, I_2, M_{err} = RenderLayer(E, tN_1, tN_2) \quad (6)$$

$$I_0 = I_1 + I_2 \quad (7)$$

The difference between the corresponding new perspective ground-truth image I_{gt} and the generated image I_0 is the rendering loss L_r , we use the rendering loss L_r as an extra supervision:

$$L_r = |I_{gt} - I_0| \odot M \quad (8)$$

Table 2: Quantitative comparison.

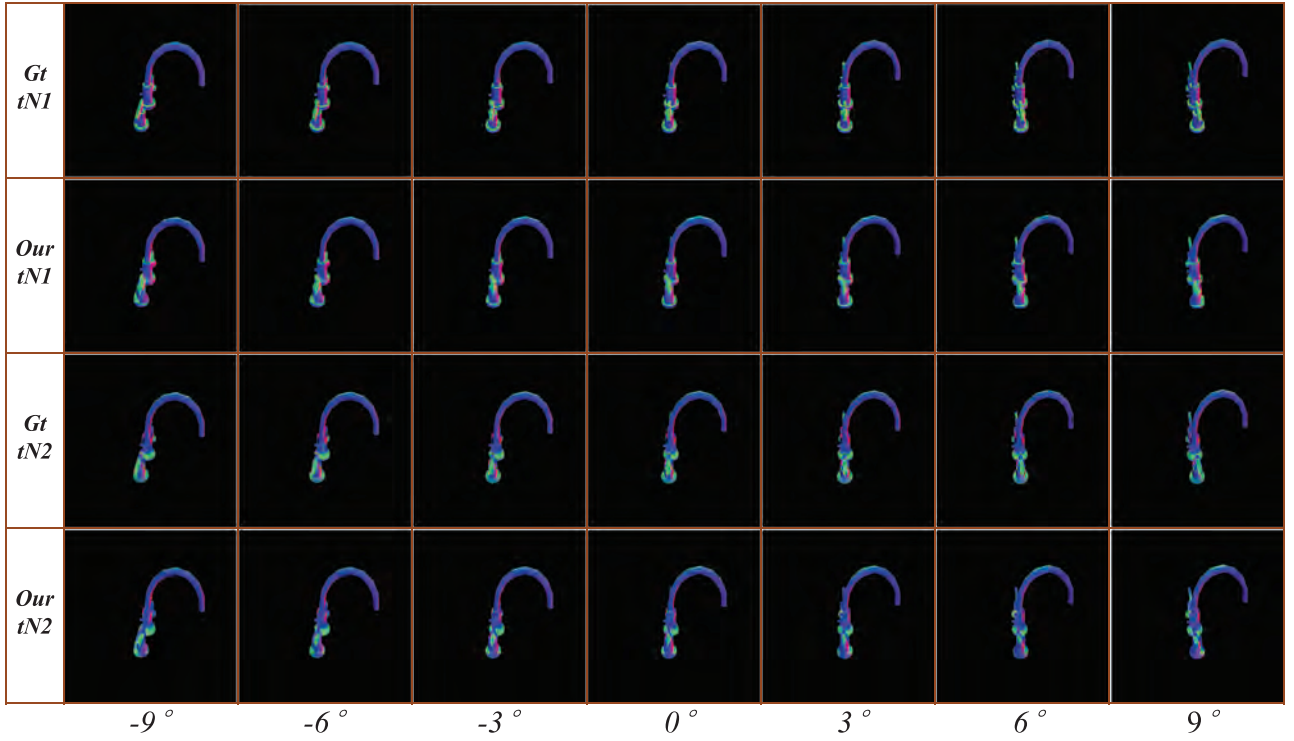
Methods	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
VAF	13.777521	0.469289	0.387784
CLDI	15.449004	0.67885	0.268121
Synsin	20.954771	0.90608	0.186442
Ours-withoutTF	18.325105	0.749121	0.257429
Ours	20.799537	0.945257	0.147385

3.5. Training

Training objective. Our network uses three loss functions for the three stages, respectively: the first one is a L2 loss L_{in} computing a conversion error from RGB pixel to its corresponding normal, the second one is a L2 loss L_{nn} computing normal vector difference between the new viewpoint and the original viewpoint, the last one is render loss L_r computing difference between the generated and target image. The total loss is defined as : $L = \lambda_1 L_{in} + \lambda_2 L_{nn} + \lambda_3 L_r$.

Table 3: In normal reconstruction S_1 in Figure 2, using the texture extractor as guidance improves the results.

Category	Ours					Ours-withoutTF				
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	N_1 mean($^\circ$) \downarrow	N_2 mean($^\circ$) \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	N_1 mean($^\circ$) \downarrow	N_2 mean($^\circ$) \downarrow
Airplane	32.5895	0.9823	0.0122	2.4860	6.6173	21.6490	0.9452	0.0492	9.1263	11.2179
Bathhub	27.4386	0.9466	0.0439	2.1634	8.3368	18.7978	0.8882	0.0983	5.8704	11.4328
Car	24.3677	0.9295	0.0655	3.4358	6.5170	20.0773	0.8956	0.0931	5.6944	9.3478
Chair	25.9679	0.9493	0.0376	3.0472	6.1562	18.6667	0.8954	0.0877	8.3584	11.5233
Faucet	30.3462	0.9790	0.0168	2.9659	5.2854	21.8057	0.9454	0.0453	8.5399	9.9096
Jar	29.1889	0.9671	0.0302	2.2816	5.8252	23.6336	0.9438	0.0412	4.5107	7.5939
Lamp	31.3621	0.9763	0.0205	2.5172	7.7641	24.3494	0.9545	0.0366	6.3432	10.7739
Cellphone	30.7037	0.9779	0.0214	2.4371	6.3997	19.5175	0.9317	0.0628	9.7623	14.7964
Vessel	27.9002	0.9659	0.0287	3.1224	8.4838	20.3468	0.9286	0.0655	8.2276	12.5942
Irregularly shaped	33.3888	0.9796	0.0190	2.1099	3.5423	21.0715	0.9365	0.0458	8.3731	10.3968

**Figure 11:** Visual comparison of generated normal maps tN_1/tN_2 over different angles from -9° to 9° , using our whole network $S_1 + S_2$ against the ground truth tN_1/tN_2 directly obtained from the 3D object normal maps.

Training details. The models are trained with the Adam optimiser using a 10^{-4} learning rate for the encoder and decoder, and momentum parameters (0.5,0.9). We have the learning rate every 50 epochs. $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 1$. All networks are trained over 200 epochs. We implement our models in PyTorch [PGM*19] and take 2 days to train on 1 NVIDIA GeForce RTX 3090 GPU.

4. Experiments

Dataset. Because there is no open source dataset dedicated to the view synthesis of transparent objects in existing works, we need to

create a synthetic dataset to evaluate our method. The first step is to collect a set of environment maps of complex daily scenes. With the support of the author in Gardner *et al.* [GSY*17], we obtain 2233 HDR panoramic pictures, from which we randomly select 1449 scene pictures to form our training set, and use the rest as our test sets. The second step is to collect transparent objects. As shown in Figure 5, we collect nine types of models from ShapeNet[CFG*15], which are *airplane*, *bathtub*, *car*, *chair*, *faucet*, *jar*, *lamp*, *cellphone*, *vessel* (including complex surfaces with holes and large concave areas). For each category, 125 models are included, 100 of which are used in the training set and the rest 25 of them are used in the test set. In addition, to prove generalization ability of our model

Table 4: Comparing view synthesis performance between left: using $S_1 + S_2$ and right: using the ground truth $N1/N2 + S_2$. As shown in the table, the quality of synthesis result of $S_1 + S_2$ is almost identical to S_2 only. This shows that our normal reconstruction step S_1 can reliably reconstruct normal maps of the 3D object from the input image.

Category	$S_1 + S_2$					S_2 only				
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	$tN_1mean(^{\circ})\downarrow$	$tN_2mean(^{\circ})\downarrow$	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	$tN_1mean(^{\circ})\downarrow$	$tN_2mean(^{\circ})\downarrow$
Airplane	23.6507	0.9594	0.0480	10.4435	12.3500	23.3725	0.9589	0.0474	10.2295	11.7814
Bathhub	20.2473	0.9086	0.1073	5.7995	11.5231	18.7737	0.8999	0.1146	5.8442	11.3736
Car	18.0402	0.8861	0.1109	5.9644	10.1538	18.5194	0.8912	0.0981	5.5936	9.1842
Chair	20.6813	0.9174	0.0904	8.2015	11.1875	20.0987	0.9151	0.0906	8.0313	10.8661
Faucet	22.0330	0.9439	0.0581	8.6676	10.5129	22.5703	0.9473	0.0518	7.9008	9.3033
Jar	19.5120	0.9104	0.1015	4.8007	9.5713	21.2901	0.9242	0.0834	3.4041	6.6167
Lamp	23.2461	0.9473	0.0619	6.0039	11.0784	22.7644	0.9465	0.0599	5.3096	10.2393
Cellphone	21.8002	0.9507	0.0588	7.7535	13.9678	22.6564	0.9552	0.0528	7.9470	13.3464
Vessel	21.3644	0.9403	0.0674	8.9847	12.9352	20.0495	0.9323	0.0721	8.5728	12.6111
Irregularly shaped	21.6182	0.9424	0.0613	5.9292	8.2836	22.8810	0.9495	0.0480	2.7686	3.5344

Table 5: Final result quality over all 3D object types.

Angel	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	$tN_1mean(^{\circ})\downarrow$	$tN_2mean(^{\circ})\downarrow$
-9 $^{\circ}$	23.4992	0.9228	0.0689	8.4473	12.0863
-6 $^{\circ}$	23.6783	0.9257	0.0667	6.9529	10.9429
-3 $^{\circ}$	24.0611	0.9291	0.0642	5.6860	9.8890
0 $^{\circ}$	24.2164	0.9312	0.0634	4.1596	8.6676
3 $^{\circ}$	23.9861	0.9293	0.0656	5.9192	10.1536
6 $^{\circ}$	23.7884	0.9271	0.0684	7.6611	11.5724
9 $^{\circ}$	23.4870	0.9244	0.0718	9.0351	12.6357

Table 6: Final result quality on 2-Bounce 3D shapes.

Angel	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	$tN_1mean(^{\circ})\downarrow$	$tN_2mean(^{\circ})\downarrow$
-9 $^{\circ}$	19.3504	0.9194	0.0582	6.2791	8.5828
-6 $^{\circ}$	19.5326	0.9212	0.0563	5.9405	8.2997
-3 $^{\circ}$	19.7247	0.9231	0.0550	5.6236	7.9925
0 $^{\circ}$	19.9327	0.9250	0.0541	5.3022	7.7019
3 $^{\circ}$	19.7583	0.9250	0.0551	5.5983	7.9907
6 $^{\circ}$	19.4203	0.9241	0.0562	5.9123	8.2959
9 $^{\circ}$	19.1846	0.9237	0.0570	6.2515	8.5855

and effectively apply it onto various 3D shapes, we randomly generate a class of irregularly shaped 3D models according to Li *et al.* [LYC20]. The IoR of all shapes is set to 1.4723 according to the average real-world transparent object. The camera's fov is 63.4149 $^{\circ}$. Assuming the camera is a perspective camera, the camera's view-point changes from -9 $^{\circ}$ to 9 $^{\circ}$ with 1 $^{\circ}$ interval along circle Q , and the resolution of the generated image is 480*360. Figure 6 shows an example of our dataset. All the transparent objects in our dataset are rendered using approach described in Li *et al.* [LYC20].

Note that in our training data, there is only a portion of the **irregular shape** and the **cellphone** categories that strictly satisfy the 2-Bounce property, roughly 10% of the entire 3D shapes. There are many 3D objects that do not satisfy the 2-Bounce property due to their complex, concave or even hollow shapes. However, we still

include these complex and concave 3D shapes and use the entire 3D dataset to train our networks. This is because we find out that by doing this, our network can obtain much better generalization ability, not just on our test data, but also on those real transparent objects we use in Figure 7. Using the entire 3D shape to train our network is significantly better than just using 2-Bounce 3D objects. The verification results are shown in Table 1 and Figure 8. The main reason behind this is also very straightforward: A larger and more versatile 3D shape training set enables our network to capture general knowledge about the silhouette, curvature and orientation from many complex 3D surfaces, not just simple convex shapes.

Metrics. We use a variety of evaluation indicators to measure the quality of the synthesized views of transparent objects, namely PSNR, SSIM and perceptual similarity (Zhang *et al.* [ZIE*18] has shown that perceptual similarity is an effective method for comparing image similarity). In addition, for the prediction accuracy of the normal map, we also added the indicator of the average angle error of the normal direction.

4.1. Visual comparisons

We compare our proposed model with Synsin[WGSJ20], CLDI[SSKH20] and VAF[ZTS*16] on the above dataset. For VAF[ZTS*16], we re-trained their networks using our dataset. For Synsin[WGSJ20] and CLDI[SSKH20], we directly use their released pre-trained networks. For all methods, we use the same input view image and camera parameters to synthesize new views of the same 19 target perspectives. Figure 9 shows the generated result of our model and other models. It can be seen from the figure that the pattern formed on the surface of a transparent object varies with the viewing angle. Our method can correctly capture the light changes and appearance of transparent objects under new perspectives, while the other three methods only consider the relationship between the depth map and the pixel changes, resulting in the transparent object surface pixels only learning information in the RGB context. CLDI[SSKH20] and VAF[ZTS*16] produce artifacts around the edge contour of transparent objects.

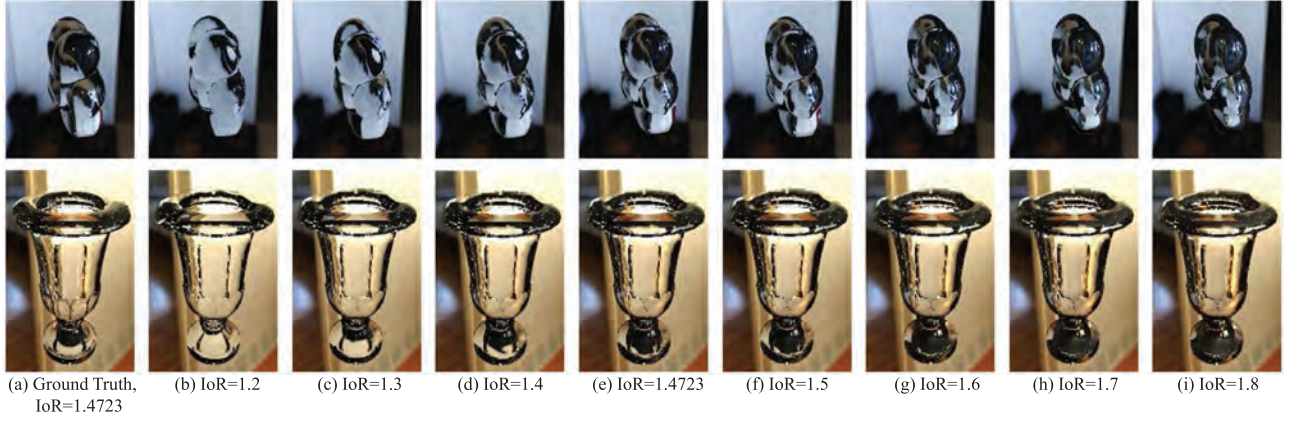


Figure 12: Our results of increasing IoR from 1.2 to 1.8. Please refer to our webpage for more results of IoR changes in GIF format.

4.2. Quantitative comparisons

To perform quantitative comparison with Synsin[WGSJ20], CLDI[SSKH20] and VAF[ZTS*16], we use SSIM, PSNR and LPIPS[ZIE*18] metrics between the synthesized image and the ground truth image of the target view to quantify the performance of each model. Because these three existing methods perform the synthesis process over the whole image space and do not use any segmentation of the foreground object, we can only compute these quality metrics over the full image space for both their methods and our method. Based on the same reason, their results may contain noticeable background distortions especially when the view change is large, we consider it is reasonable to only compare a narrow range of views from -3° to 3° without 0° , six views in total. Specifically, we use all of our 250 3D models from our 10 types mentioned in Section 4, along with various complex background environment scenes, and for each 3D model, we form six test pairs each of which the input view is 0° and the target views are $T_v \in \{-3^\circ, -2^\circ, -1^\circ, 1^\circ, 2^\circ, 3^\circ\}$. We use the same camera parameters and input views for all methods in the quantitative comparison. In Figure 9, we show the full 19 frames of four examples of this comparison.

We report the evaluation results in Table 2. For all three metrics, our results outperform these baselines except only in PSNR our results are slightly worse than Synsin[WGSJ20] by 0.74%. Our method is evidently better in SSIM and LPIPS. From both the static frames in Figure 9 and the GIF results on our webpage, it is also demonstrated that our results have better lighting and textural correctness, perceptual quality and structural integrity of the synthesized transparent objects. Besides, in Table 2, we also conduct ablation studies of our proposed individual network components which are detailed in the next Section 4.4.

4.3. Ablation study

In order to see the influences of our proposed components on the final result quality, we conduct the following ablation studies.

First, we verify the influence of the texture extractor in step S_1 on the quality of the reconstructed normals of transparent objects. The

visualization results of these studies are shown in Figure 10. Using our test set, we also compute PSNR, SSIM, LPIPS and the average angle bias in the normal direction to evaluate the difference between the generated normal map and the ground truth normal map. We report the evaluation results in Table 3. The results show that our texture extractor brings improvement in PSNR, SSIM and LPIPS, especially in PSNR. When estimating the normal direction, the average error is reduced by approximately 4.824° . Besides, the loss curves in the supplementary files also show that with the texture extractor, our network has lower losses especially during the first 50 epochs.

Then, we test the accuracy of our step S_1 and the correctness of combining S_1 with S_2 . Within our test set, we replace the reconstructed normals N_1/N_2 with the ground truth N_1/N_2 , and input them into our step S_2 . The visualization results are shown in Figure 11, and the image quality measurements and normal accuracy metrics are listed in Table 4. Comparing the left and the right of the table suggests that our step S_1 can faithfully recover normals from a single input image and the quality of our final view synthesis results is only slightly worse than inputting ground truth N_1/N_2 into step S_2 . In the categories of *chair*, *bathub* and *airplane*, our network even gives higher score on PSNR and SSIM. However, a row-by-row comparison of the table shows that NVS results of objects with large hollow shapes (bath tub) and complex structure and severe self-occlusions (chair and airplane) has lower quality, this is because our renderer is limited by 2-Bounce assumption.

We next examine how image quality varies with viewing angles. We test our method on all 3D types in our test set and on the 2-Bounce 3D shape subset, and the evaluation results are shown in Tables 5 and 6, respectively. Both of these two tables show that the quality of the synthesized new viewing angle image decreases very slightly with the increase of the viewing angle variation. Zero degrees is the best, and 9° and -9° are the worst. But the quality of the synthetic results are acceptable without any visible differences throughout all viewing angles. For instance, on the full 3D model, the drop rate is only 3.02%, while on the 2-Bounce model, it is 3.75%. In Figure 12, we show both the ground truth tN_1/tN_2 and their generated version by our method using an example from the *faucet* category.

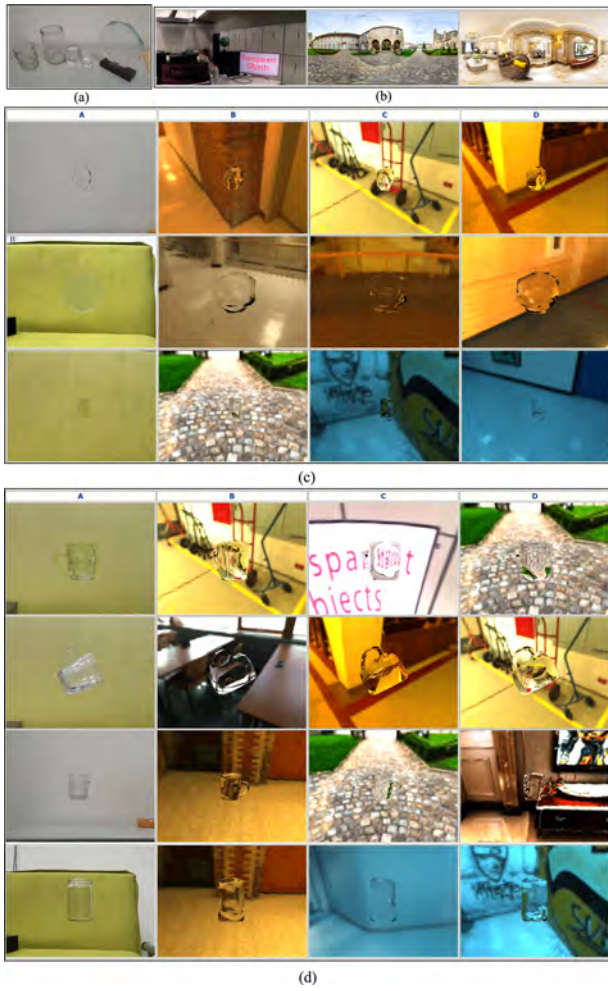


Figure 13: View synthesis results using real transparent objects. Please refer to our webpage for GIF results. (a) Real objects. (b) New environment maps, the first is a photo of our lab. (c, d) The first column is the input image, the masks are manually labelled. (c) 2-Bounce objects. (d) Objects that are not 2-Bounce.

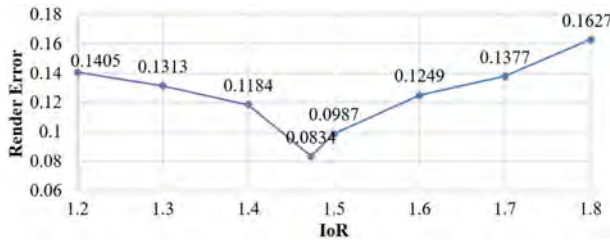


Figure 14: The rendering error of different IoR on the test set.

4.4. Index of refraction

We also report a sensitivity analysis on the influence of IoR on the render result. We use the IoR in the range (1.2, 1.8) to test our network, and compare the generated image with the ground truth. Figure 13 shows transparent objects rendered under different IoRs,

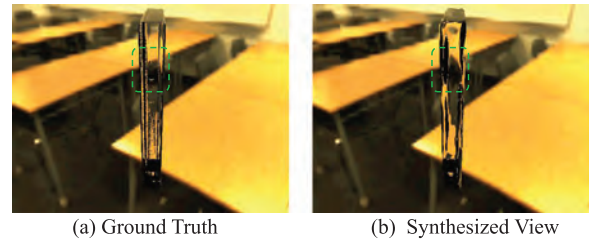


Figure 15: One of our synthesized view showing texture distortion.

Figure 14 shows the rendering error of different IoRs on the test set. It can be seen that the closer the IoR of the transparent object is to 1.4723, the smaller the rendering error. This is because our dataset uniformly sets the IoR of transparent objects to 1.4723, which is in line with our real-world expectations. At the same time, the rendering error generated by IoRs in this range only slightly changes. This suggests that our method can effectively adjust the IoR for any given 3D model and perspective.

4.5. Real transparent objects and environment maps

We test our method using photos of the real 3D transparent objects such as glass cup and trophy as displayed in Figure 7a. Here we also use some completely new environment maps that are not from our training dataset, some of them are downloaded from the Internet, and the others are made from the photos we take in our lab or certain outdoor places as shown in Figure 7b. The results are shown in Figure 7c. Note here we manually mask out the transparent objects before inputting them into our networks.

5. Limitation

Even though we do not require inputting 3D shape, it is still necessary to input segmentation masks when generating novel views for real 3D objects from real photos. Our loss function only penalizes the point-wise differences between the ground truth normal and the predicted normal, this may lead to noticeable distortion in the results, see Figure 15. Due to GPU memory limit, it is currently inapplicable for us to induce more geometric constraints such smooth error for neighbouring normal vectors or other surface-based curvature guidance. In our experiments, a single RTX-3090 GPU can not afford the peak memory consumption when such geometric constraints are added. Besides, our method currently does not handle total internal reflection. Furthermore, we assume that objects are totally smooth and homogeneous with uniform IOR, and our networks are entirely trained with synthetic dataset.

6. Conclusion

In this paper, we introduce a method to synthesize novel views from a single input view. Our core technique is to consider light transmission characteristics and viewing angle-related effects, and learn the relationship between light and pixels under new perspectives. Experimental results show that our method can better capture the light changes and appearance of transparent objects under new viewing

angles. Currently our method only consider the view synthesis problem of a single transparent object in a complex scene, we will consider the case of multiple transparent objects in the future.

Acknowledgements

This work was funded by the National Science Foundation of China No. 62076090, Huxiang Youth Talent Support Program, Hunan Province China, No. 2020RC3014, Natural Science Foundation of Hunan Province, China, No. 2022JJ30173. We thank Prof. Jean-François Lalonde for providing us datasets. We Thank Lei Xia for helping with experiments.

References

- [AYTK20] Alex Yu, Vickie Ye, Matthew Tancik, Angjoo Kanazawa: pixelNeRF: Neural Radiance Fields From One or Few Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, IEEE. 4578–4587.
- [CFG*15] Chang A. X., Funkhouser T., Guibas L., Hanrahan P., Huang Q., Li Z., Savarese S., Savva M., Song S., Su H., Xiao J., Yi L., Yu F.: ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012* (2015).
- [CHW18] Chen G., Han K., Wong K.: TOM-Net: Learning transparent object matting from a single image. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 9233–9241.
- [CPK*17] Chen L. C., Papandreou G., Kokkinos I., Murphy K., Yuille A. L.: DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 4 (2017), 834–848.
- [CWY*20] Chao C. K., Wu S. Y., Yan Z. T., Tsai M. L., Hsu C. C., Raihany U., Peng C. Y.: Robotic arm combined with the visual images in a transparent object recognition. In *Proceedings of the 2020 International Symposium on Computer, Consumer and Control (IS3C)* (2020), pp. 126–129.
- [DSB*12] Darabi S., Shechtman E., Barnes C., Goldman D. B., Sen P.: Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–10.
- [FWZ05] Fitzgibbon A., Wexler Y., Zisserman A.: Image-based rendering using image-based priors. *International Journal of Computer Vision* 63, 2 (2005), 141–151.
- [GEB15] Gatys, L. A.; Ecker, A. S.; Bethge, M. Texture synthesis using convolutional neural networks. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 262–270, 2015.
- [GSY*17] Gardner, M.A., Sunkavalli, K., Yumer, E., Shen, X., Gambaretto, E., Gagné, C., Lalonde, J.F.: Learning to predict indoor illumination from a single image. *ACM Transactions on Graphics* 36(6), 1–14, 2017.
- [HK18] Hedman P., Kopf J.: Instant 3D photography. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.
- [HKAK14] Huang J. B., Kang S. B., Ahuja N., Kopf J.: Image completion using planar structure guidance. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.
- [HPP*18] Hedman P., Philip J., Price T., Frahm J. M., Drettakis G., Brostow G.: Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–15.
- [HWL17] Han K., Wong K., Liu M.: Dense reconstruction of transparent objects by altering incident light paths through refraction. *International Journal of Computer Vision* 126 (2017), 460–475.
- [KLS*13] Kopf J., Langguth F., Scharstein D., Szeliski R., Goesele M.: Image-based rendering in the gradient domain. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–9.
- [KTR*20] Kalra A., Taamazyan V., Rao S. K., Venkataraman K., Raskar R., Kadambi A.: Deep polarization cues for transparent object segmentation. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 8599–8608.
- [LSR*20] Li Z., Shafiei M., Ramamoorthi R., Sunkavalli K., Chandraker M.: 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.
- [LYC20] 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* (2020), pp. 1262–1271.
- [MNSiT13] Maeno K., Nagahara H., Shimada A., Ichiro Taniguchi R.: Light field distortion feature for transparent object recognition. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 2786–2793.
- [MSOC*19] Mildenhall B., Srinivasan P. P., Ortiz-Cayon R., Kalantari N. K., Ramamoorthi R., Ng R., Kar A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–14.
- [MST*20] 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 *Proceedings of the European Conference on Computer Vision* (2020), Springer, pp. 405–421.

- [NMYL19] NIKLAUS S., MAI L., YANG J., LIU F.: 3D Ken Burns effect from a single image. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–15.
- [PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., DESMAISON A., KÖPF A., YANG E., DEVITO Z., RAISON M., TEJANI A., CHILAMKURTHY S., STEINER B., FANG L., BAI J., CHINTALA S.: PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32. Curran Associates, 2019: 8026–8037.
- [PKD*16] PATHAK D., KRAHENBUHL P., DONAHUE J., DARRELL T., EFROS A. A.: Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2536–2544.
- [PYY*17] PARK E., YANG J., YUMER E., CEYLAN D., BERG A. C.: Transformation-grounded image generation network for novel 3D view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 3500–3509.
- [RFJ21] ROCKWELL C., FOUHEY D. F., JOHNSON J.: PixelSynth: Generating a 3D-consistent experience from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14104–14113.
- [RPE21] ROMBACH, R., ESSER, P and OMMER, B (2021). Geometry-Free View Synthesis: Transformers and no 3D Priors. In *Proceedings of the Intl. Conf. on Computer Vision (ICCV)*.
- [SMP*20] 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 2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), IEEE, pp. 3634–3642.
- [SRSF19] SHIN D., REN Z., SUDDERTH E. B., FOWLKES C. C.: 3D scene reconstruction with multi-layer depth and epipolar transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 2172–2182.
- [SSKH20] SHIH M. L., SU S. Y., KOPF J., HUANG J. B.: 3D photography using context-aware layered depth inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 8028–8038.
- [STB*19] SRINIVASAN P. P., TUCKER R., BARRON J. T., RAMAMOORTHY R., NG R., SNAVELY N.: Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 175–184.
- [SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations (ICLR2015)* (2014).
- [TGF*18] TULSIANI S., GUPTA S., FOUHEY D. F., EFROS A. A., MALIK J.: Factoring shape, pose, and layout from the 2D image of a 3d scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 302–310.
- [TS20] TUCKER R., SNAVELY N.: Single-view view synthesis with multiplane images. *arXiv preprint arXiv:2004.11364* (2020).
- [WGL*18] WHELAN T., GOESELE M., LOVEGROVE S. J., STRAUB J., GREEN S., SZELISKI R., BUTTERFIELD S., VERMA S., NEWCOMBE R. A.: Reconstructing scenes with mirror and glass surfaces. *ACM Transactions on Graphics* 37, 4 (2018), 102.
- [WGSJ20] WILES O., GKIOXARI G., SZELISKI R., JOHNSON J.: SynSin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 7467–7477.
- [XBS*19] XU Z., BI S., SUNKAVALLI K., HADAP S., SU H., RAMAMOORTHY R.: Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–13.
- [XYL*19] XIONG W., YU J., LIN Z., YANG J., LU X., BARNES C., LUO J.: Foreground-aware image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 5840–5848.
- [ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 586–595.
- [ZTF*18] ZHOU T., TUCKER R., FLYNN J., FYFFE G., SNAVELY N.: Stereo magnification: Learning view synthesis using multiplane images. In *Proceedings of the ACM SIGGRAPH* 2018.
- [ZTS*16] ZHOU T., TULSIANI S., SUN W., MALIK J., EFROS A. A.: View synthesis by appearance flow. In *Proceedings of the European Conference on Computer Vision* (2016), Springer, pp. 286–301.
- [ZWLQ19] ZHANG Z., WANG Z., LIN Z., QI H.: Image super-resolution by neural texture transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 7982–7991.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Data S1