Consistent Depth Prediction for Transparent Object Reconstruction from RGB-D Camera
Supplemental Material

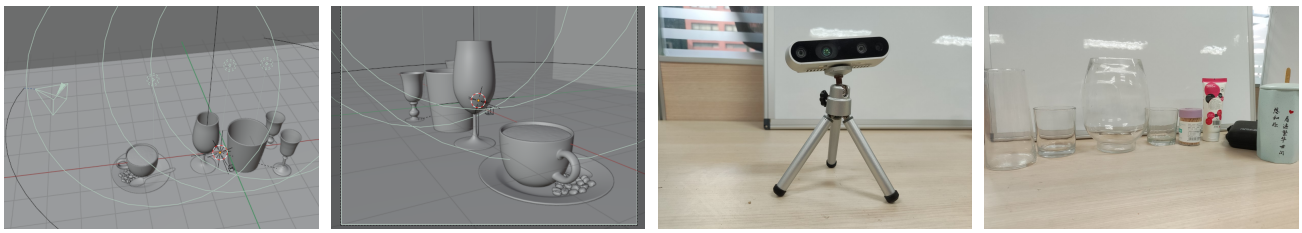
Yuxiang Cai Yifan Zhu Haiwei Zhang Bo Ren *
Nankai University
{caiyuxiang, zhuyifan}@mail.nankai.edu.cn, {zhhaiwei, rb}@nankai.edu.cn

This is the supplementary material for our proposed real-time transparent object reconstruction method. In this material, we will introduce some details about our datasets and show more evaluation results. Besides, we provide a video that shows our results compared with the previous methods introduced in our paper.

A. Real-time Performance

The implementation of our Pseudo-SLAM system is based on C++ and Libtorch framework. The mask prediction result is provided individually using [2]. We query the result and send it to our depth prediction module. The average time for constructing a sequence with 300 frames is 31.1s. Note that the initialization time is about 1.4s. The GPU warm-up at the early step costs about 9.3s on average. We only start calculating FPS when the running time is stable for each frame. When stabilizes, the depth prediction module takes 0.0346s and RGB-D reconstruction takes 0.0317s on average per frame. The FPS of our method is 14.5 on average.

B. Data Creation



(a) Synthetic Scene (b) Synthetic Scene from Camera View (c) RGB-D Camera (d) Transparent and Opaque Objects

Figure 1. **Relevant software and tools used to build our dataset.**

B.1. Synthetic Data Creation using Blender

We have discussed in our paper that the previous datasets are not suitable for our depth prediction model. Since obtaining ground truth depth data of transparent objects in real scenes is hard, we use the free software *Blender* to simulate real scenes that contain the transparent objects. We use the Python script to render these scenes and set a circle camera trajectory to obtain a sequence of ground truth depth data, mask, and RGB. We create 15 scenes for training and 5 scenes for validation. Our training scenes contain glass cups, glass goblets, wine cups, and square transparent boxes. Each scene contains about 400-600 frames. These data do not contain Kinect-like noise but to simulate real-world data captured by the RGB-Dcamera, we add Gaussian noise and controllable motion blur to depth and RGB data at the training stage. For validation scenes, we do not include new transparent object models. Instead, we set different placements of the glass cup model to create scenes: ”Glass Cup” and ”Glass Cup(down)” and different trajectories of the glass goblet model to create scenes: ”Glass Goblet” and ”Glass

*Bo Ren is the corresponding author

Goblet(part)". The scene "Snack Plate" contains multiple opaque boxes and a transparent one. The number of frames that each validation scene contains is listed in Tab. 1.

B.2. Capture Real Scene Data using RGB-D Camera

We use a hand-hold RealSense D435i RGB-D camera at a frame rate of 30 Hz and 640x480 resolution in the real scene data capturing. We use *Kalibr* toolbox [1] and the *April tag* for camera calibration. The specific calibration information is stored in the "camera_info" file in our real captured dataset. We select four daily used glass objects: "Angular Cup", "Long-Necked Cup", "Vase", and "Cylindric Cup" as our test transparent objects. The transparent and opaque objects used to build our real scene dataset are shown in Fig. 1 and the number of frames is shown in Tab. 1. Because we capture scenes with hand-holding camera, the video we get occasionally shakes. We discard these fragments as they will greatly impact the ElasticFusion reconstruction method.

Synthetic Scene				
Glass Cup	Glass Cup(down)	Goblet	Goblet(part)	Snack Plate
300	300	450	300	400
Real Scene				
Angular Cup	Long-Necked Cup	Vase	Cylindric Cup	Cylindric Cup(floor)
515	490	448	320	379

Table 1. **Number of frames in different scenes.**

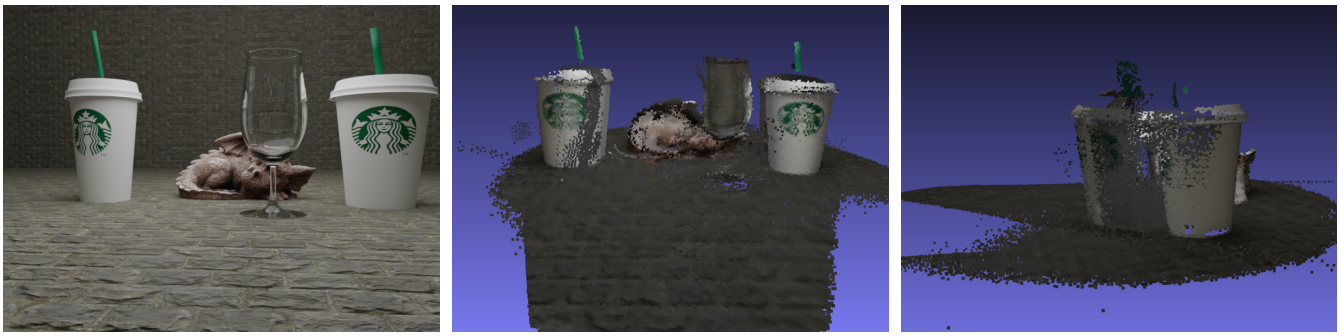
	Glass Cup			
Method	Chamfer ↓	Prec ↑	Recall ↑	F-Score ↑
ClearGrasp	0.068	0.462	0.368	0.410
TransCG	0.061	0.611	0.443	0.500
Ours	**0.010**	**0.670**	**0.664**	**0.667**
	Glass Cup(down)			
Method	Chamfer ↓	Prec ↑	Recall ↑	F-Score ↑
ClearGrasp	0.094	**0.529**	0.437	**0.479**
TransCG	0.083	0.485	**0.455**	0.470
Ours	**0.075**	0.421	0.387	0.403
	Goblet(part)			
Method	Chamfer ↓	Prec ↑	Recall ↑	F-Score ↑
ClearGrasp	0.038	0.540	0.518	0.517
TransCG	0.026	0.627	0.829	0.765
Ours	**0.015**	**0.847**	**0.908**	**0.815**
	Goblet			
Method	Chamfer ↓	Prec ↑	Recall ↑	F-Score ↑
ClearGrasp	0.043	0.540	0.459	**0.600**
TransCG	0.029	**0.627**	0.437	0.580
Ours	**0.019**	0.402	**0.785**	0.557
	Snack Plate			
Method	Chamfer ↓	Prec ↑	Recall ↑	F-Score ↑
ClearGrasp	0.055	0.731	0.771	0.751
TransCG	0.046	0.771	0.779	0.775
Ours	**0.018**	**0.913**	**0.863**	**0.887**

Table 2. **Detailed reconstruction evaluation result.**

C. Transparent Object Reconstruction

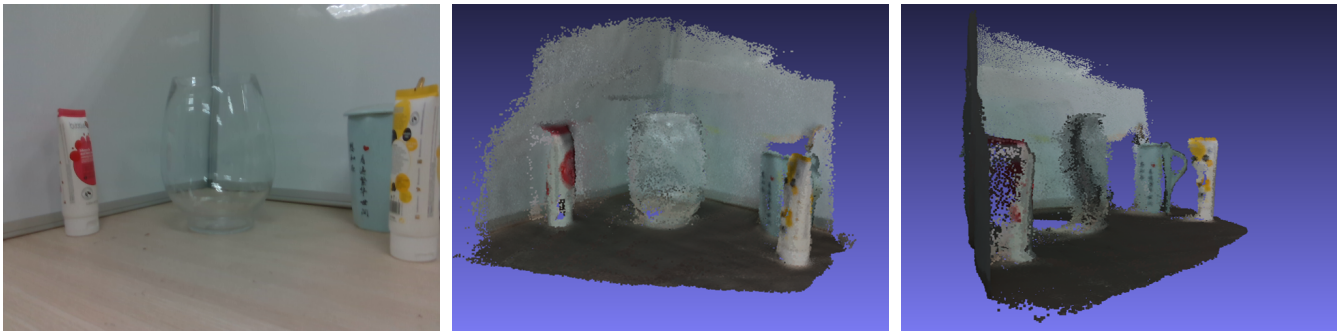
Our geometry reconstruction result is formatted as "surfel" proposed in ElasticFusion, which mainly contains RGB color, location, surface normal, and radius. We provide a video that displays the process of the surfel reconstruction. This video contains detailed visualization of geometry results in the triangle mesh.

Besides, we show our reconstruction results of the above scenes in Fig. 5 and Fig. 4 with a higher resolution. We also show the detailed reconstruction evaluation metrics used in our paper in Tab. 2.



(a) RGB (b) View 1 (c) View 2

Figure 2. **Bad cases in our synthetic dataset.**



(a) RGB (b) View 1 (c) View 2

Figure 3. **Bad cases in our real dataset.**

Bad cases analysis. We find that for the "Goblet" scene, our method fails to predict the consistent depth value on different views. As shown in Fig. 2, the reconstruction result is not complete. The surface of the transparent object can be reconstructed only from the displayed view of the image, while from another view, the result is distorted. We argue that the incomplete reconstruction is due to the accumulated error of camera trajectory prediction. This error is caused by the highlight on the surface of the transparent object and leads to the unmatching of neighboring images in our model. Besides, in the real captured scene, the material of the transparent object is different from the one in the synthetic scene. It is hard to simulate the material of real transparent objects. In the "Vase" scene shown in Fig. 3, due to the large shape different from the transparent objects in our dataset, our method can not restore the curved surface of the transparent vase. For future work, to resolve these problems, we will include more transparent objects with different shapes in our dataset and improve the accuracy of camera trajectory estimation in high-light scenes.

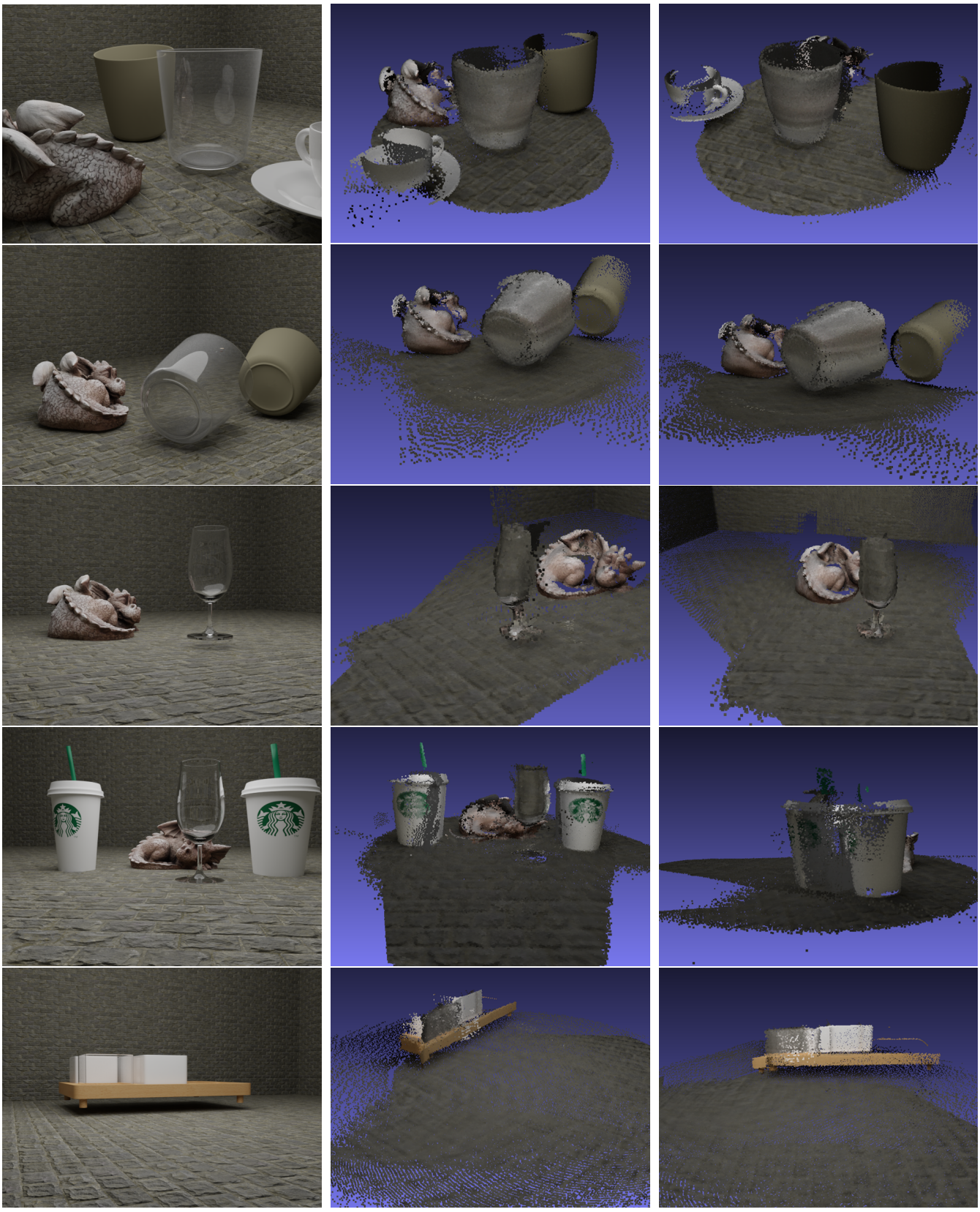
References

[1] Jérôme Maye, Paul Furgale, and Roland Siegwart. Self-supervised calibration for robotic systems. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 473–480. IEEE, 2013. 2

[2] Yifan Zhu, Jiaxiong Qiu, and Bo Ren. Transfusion: A novel slam method focused on transparent objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6019–6028, 2021. 1

(a) RGB (b) View 1 (c) View 2

Figure 4. **Reconstruction result in the real scene.** (b) and (c) are our reconstruction results in different views. From top to bottom are: scene ”Angular Cup”, scene ”Long-Necked Cup”, scene ”Vase”, scene ”Cylindric Cup” and scene ”Cylindric Cup(floor)”.



(a) RGB (b) View 1 (c) View 2

Figure 5. **Reconstruction result in the synthetic scene.** (b) and (c) are our reconstruction results in different views. From top to bottom are scene ”Glass Cup”, scene ”Glass Cup(down)”, scene ”Goblet(part)”, scene ”Goblet” and scene ”Snack Plate”.