Supplementary Materials of GCRayDiffusion: Pose-Free Surface Reconstruction via Geometric Consistent Ray Diffusion

Anonymous ICCV submission

Paper ID 11950

1. More Details of Camera Estimation from Neural Ray Bundles

Given a collection of neural bundle rays $\mathcal{R}_i$ associated with 2D pixels $\{u_k\}_M$, here we can give the derivation details to recover the camera pose T by solving the intersection of all rays in $\mathcal{R}_i$.

Specifically, we begin by defining each ray in the bundle that each ray $\mathbf{r}_k^i$ is represented as a 7-dimension vector including a unit directional vector $\mathbf{v}_k^i \in R^3$ through any point $\mathbf{p}_k^i \in R^3$ and depth $d_k^i \in R$ following Plücker coordinates [1] as:

$$\mathbf{r}_k^i = (\mathbf{v}_k^i, \mathbf{m}_k^i, \mathbf{d}_k^i) \in R^7, \tag{1}$$

For each ray, we have the corresponding 2D pixel u_k in the image plane, which we can relate to the 3D scene via the camera's intrinsic and extrinsic parameters.

Next, we aim to recover the camera's pose, represented by a transformation matrix T. This matrix consists of the rotation and translation of the camera, mapping 3D world coordinates to 2D image coordinates. We solve for the camera pose by finding the intersection of the rays in the bundle $\mathcal{R}_i$. In practice, this involves solving the system of equations resulting from the projection of the 3D rays onto the 2D image plane, using the camera's intrinsic parameters.

Let the projection matrix P be defined as:

$$P = K[R|t]$$

where K is the intrinsic matrix of the camera, R is the rotation matrix, and t is the translation vector. The goal is to estimate R and t such that the projected rays from the world coordinates align with the observed rays in the image. This is achieved by minimizing the geometric error between the projected 3D points and the corresponding 2D points.

To perform this optimization, we can use methods such as bundle adjustment or direct least squares optimization, which iteratively adjusts R and t to minimize the projection error across all rays in $\mathcal{R}_i$. The solution is the camera pose T that best explains the observed rays.

Once the camera pose is recovered, we can refine it by considering the depth information from the rays and adjusting the parameters to improve the alignment of the rays with the 3D scene. This approach provides a robust way to estimate camera poses from neural ray bundles, leveraging the geometry of the rays and the 3D structure captured in the scene.

2. More Details on Experiments

More Details of Image Feature Extractor $F_{\mathcal{I}}$. We use the DINOV2 transformer-based image encoder for feature extraction.

The image encoder is responsible for extracting the feature maps from the image input, which are then processed further by the downstream models.

More Details of Triplane-based SDF F_θ. We formulate F_θ as a triplane-based signed distance field (SDF), which consists of a Transformer-based image encoder Φ to extract triplane feature maps from image inputs and a MLP-based decoder $\mathcal{D}$ to regress the SDF prediction.

The triplane's initial size is 32×32 with 512 output channels. After up-sampling, a Multihead MLP (Multilayer Perceptron) is used to decode the Signed Distance Function (SDF). The up-sampling operation increases the spatial resolution, while the Multihead MLP helps to refine the feature map and interpret the SDF, allowing the model to better capture the underlying geometry and surface details.

Time and Efficiency Analysis. Here we give more details of our GCRayDiffusion in terms of inference time and GPU memory storage for each main components, including the image feature extraction (IFE), geometric consistent ray diffuser (GCRD) and triplane-based SDF (SDF) respectively.

As shown in Table 1, on average the IFE module takes about 50ms for image feature extraction, GCRD takes about 200ms and SDF takes about 300ms for ray diffuser and SDF learning respectively. As for the GPU memory storage, on average IFE module costs about 1G for image feature ex-

	IFE	GCRD	SDF
Time	50ms	200ms	300ms
GPU Storage	1G	1G	2G

Table 1. The inference time for each component of our GCRayDiffusion

traction, GCRD costs about 1G and SDF cost about 3G for ray diffuser and SDF learning respectively.

More Visual Comparison Results. Fig. 1 to Fig. 4 show more visual surface reconstruction and camera pose estimation results evaluated on Objaverse and GSO dataset for different comparing approaches, including RelPose++, FORGE, DUSt3R and our GCRayDiffusion rescptively.

3. Limitation and Discussion

One main limitation of our current solution in GCRayDiffusion is that the geometric consistent ray diffuser is mainly pre-trained on single object dataset, i.e., Objaverse, therefore our GCRayDiffusion couldn't achieve satisfied camera pose estimation and surface reconstruction results for the scene scenario. One possible solution is to use large scene dataset to train our GCRayDiffusion, but there still remains challenge for scene data annotation. Moreover, it is also a challenging problem to the joint learning of camera pose estimation and surface reconstruction for large scene images, especially given sparse image input. The second limitation comes from the neural surface learning, since our current triplane-based SDF learning would also fail for extremely sparse inputs. One interesting future work is formulating both the camera pose estimation and neural surface learning as a unit diffusion processing, thus enabling very robust camera pose estimation and surface reconstruction through the diffusion even in extreme sparse inputs.

References

[1] Julius Plücker. *Analytisch-Geometrische Entwicklungen*. GD Baedeker, 1828. 1

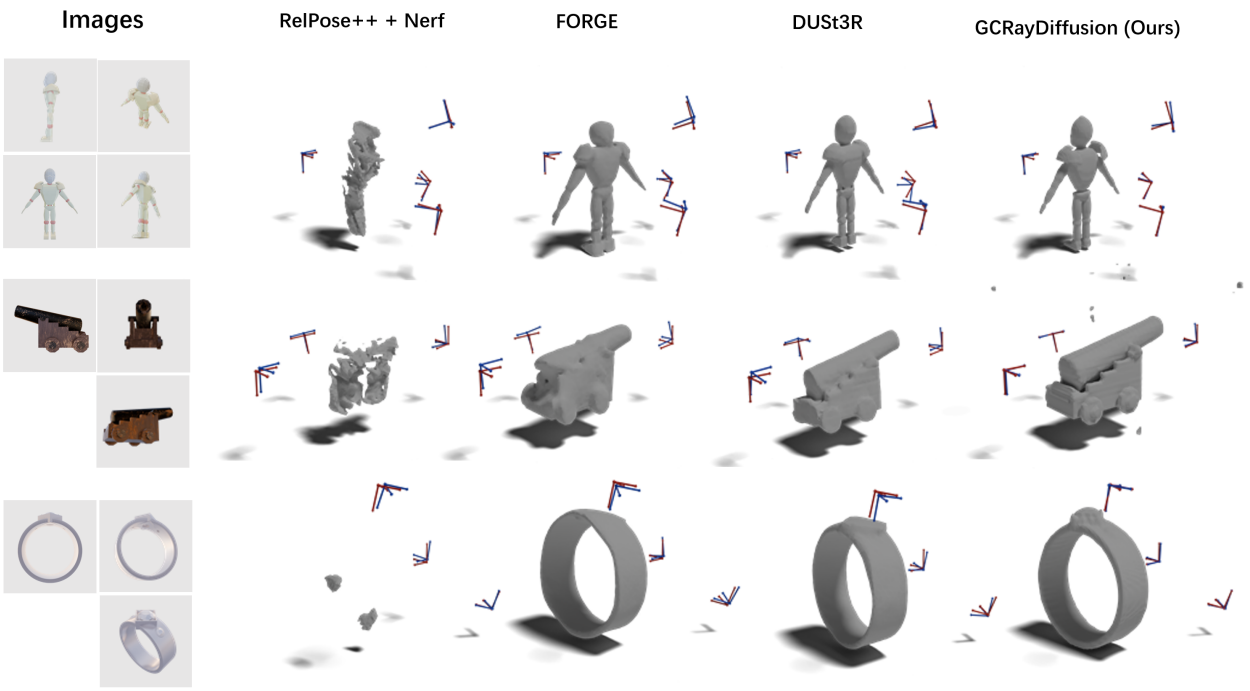

Figure 1. Qualitative surface reconstruction comparison evaluated on Objaverse dataset for different comparing approaches, including RelPose++, FORGE, DUSt3R and our GCRayDiffusion (from left to right column) respectively.

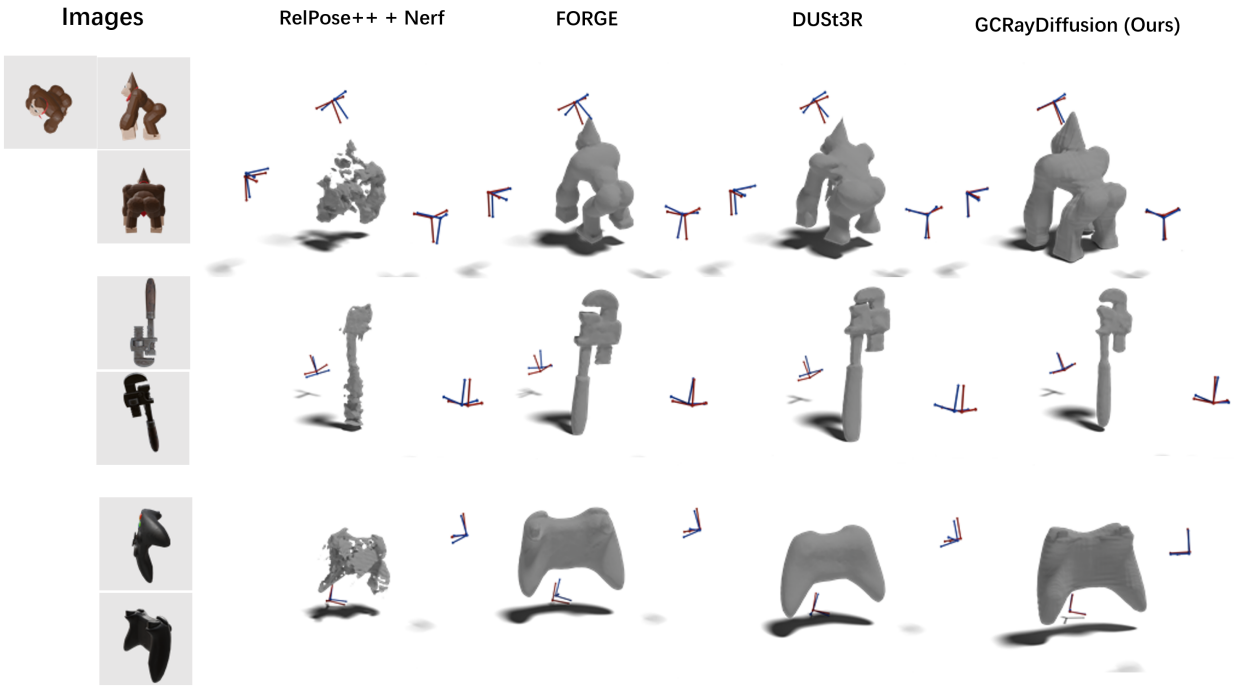

Figure 2. Qualitative surface reconstruction comparison evaluated on Objaverse dataset for different comparing approaches, including RelPose++, FORGE, DUSt3R and our GCRayDiffusion (from left to right column) respectively.

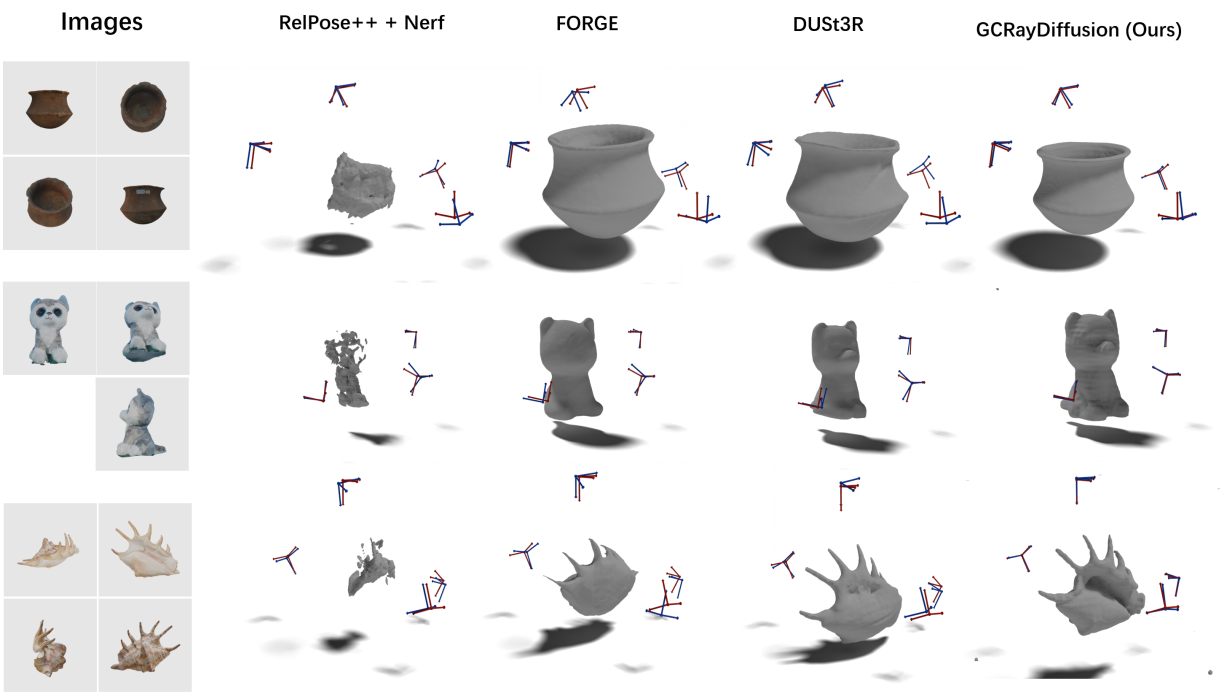

Figure 3. Qualitative surface reconstruction comparison evaluated on Objaverse dataset for different comparing approaches, including RelPose++, FORGE, DUSt3R and our GCRayDiffusion (from left to right column) respectively.

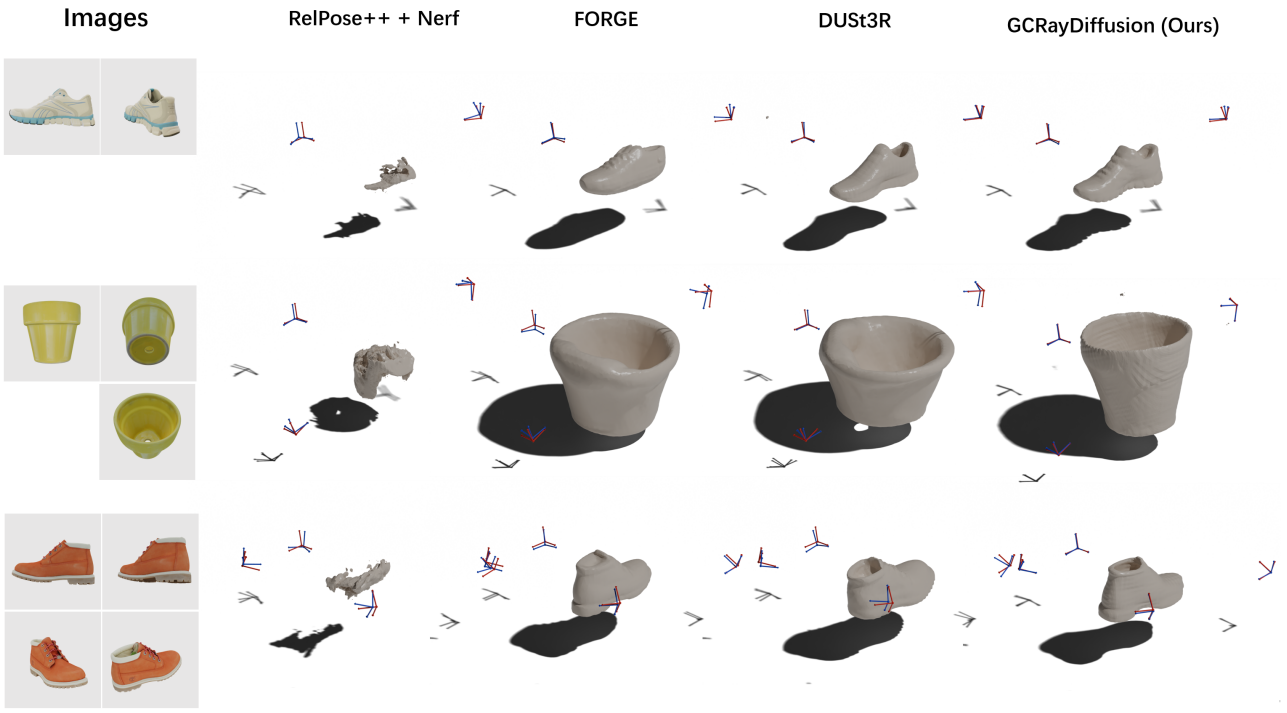

Figure 4. Qualitative surface reconstruction comparison evaluated on GSO dataset for different comparing approaches, including RelPose++, FORGE, DUSt3R and our GCRayDiffusion (from left to right column) respectively.