

Twiner: Shining Light on Digital Twins in a Few Snaps

Supplementary Material

6. Supplementary Material

6.1. Network Details

Tricolumn Encoder We predict a tricolumn grid with spatial dimensions of $R = 128$, and a feature dimension of $D = 1024$ from our transformer network. The transformer network consists of 16 cross attention layers, interlaced with splatting layers before cross attention layers number 0, 8, and 15.

Illumination Module The illumination module consists of a transformer with 16 cross-attention layers using a feature dimension of $D = 1024$. The predicted cubemap has a resolution of 128×128 for each face. We use as positional encoding the 3D directional vectors corresponding to each pixel in the cubemap. These directional vectors are then passed to a harmonic encoding in order to produce a D -dimensional positional encoding.

6.2. Training Details

As mentioned in the main script, we use a combination of multiple loss functions to supervise Twiner during training. Specifically, we use the losses \mathcal{L}_S to supervise material properties and illumination with synthetic data, \mathcal{L}_N to supervise predicted normals using our pseudo-gt normals from density, \mathcal{L}_{TV} to smoothen albedo, normals, and depth, \mathcal{L}_d to supervise the rendered depth, \mathcal{L}_M to supervise the foreground object's opacity mask, and \mathcal{L}_R to supervise the whole model using the composite rendered images. The final loss we use is calculated as follows:

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_R + \mathcal{L}_S + \mathcal{L}_M + 0.1\mathcal{L}_N + 0.1\mathcal{L}_{TV} + 0.1\mathcal{L}_d. \quad (11)$$

We finetune our model until convergence using 32 A100 GPUs, which takes approximately two days.

6.3. BRDF details

As mention in the main script, we are able to compute the outgoing radiance at any point according to the reflectance equation Eq. (1). We utilize the disney BRDF parameterized by albedo, metalicity and roughness together with the split-sum approximation as per [22]. That is, we compute the diffuse and specular components of the reflectance equation $\hat{\mathbf{L}}_d$ and $\hat{\mathbf{L}}_s$ as

$$\hat{\mathbf{L}}_d = \frac{\hat{\mathbf{k}}_d \hat{\mathbf{a}}}{\pi} \hat{\mathbf{L}}_d^i, \quad \hat{\mathbf{L}}_s = (\hat{\mathbf{F}}_r * F_1 + F_2) \hat{\mathbf{L}}_s^i, \quad (12)$$

where F_1 and F_2 are precomputed and stored in a 2D lookup table, $\hat{\mathbf{L}}_s^i$ and $\hat{\mathbf{L}}_d^i$ are pre-integrated illumination maps

which we obtain from our predicted illumination cubemap in a differentiable manner following [34], and where

$$\begin{aligned} \hat{\mathbf{k}}_d &= (1 - \hat{m}) * (1 - \hat{\mathbf{F}}_r). \\ \hat{\mathbf{F}}_r &= \hat{\mathbf{F}}_0 + (1 - \hat{\rho} - \hat{\mathbf{F}}_0) * (1 - \langle \hat{\mathbf{n}}, \omega_o \rangle)^5, \\ \hat{\mathbf{F}}_0 &= (1 - \hat{m}) * 0.04 + \hat{m} * \hat{\mathbf{a}}. \end{aligned} \quad (13)$$

6.4. Meshing Details

We first estimate a point cloud covering the object by rendering a set of depth images from fixed viewpoints and projecting them into 3D space. This point cloud is used to initialize a tetrahedral grid with a resolution of 128^3 . We then sample density values from our implicit representation composed of a predicted tricolumn and the learnt light-plane MLP at the tetrahedral vertex locations. In order to remove some of the noise from the density estimates we apply a graph-based smoothing operator along the edges of the tetrahedral grid. Finally, we apply marching tetrahedra on the density grid to extract a mesh.

Once a mesh has been extracted we are then required to extract PBR textures. We first simplify the mesh through edge contraction with a target of 40K faces. Since we are targeting single object reconstruction we then filter the mesh to keep only the single largest cluster of faces thus removing any floating artifacts which could have been created due to noisy density. Given this simplified mesh, we then perform UV mapping and extract albedo, metalicity, and roughness for each uv index using its corresponding 3D point on the mesh. Given the 3D surface point and corresponding normal, we move a fixed distance along the normal direction and perform volume rendering to extract the albedo, metalicity, and roughness in order to account for distributed density along the surface of the object.

After extracting a PBR textured mesh, we then apply the texture enhancement proposed in [45] in order to refine the predicted PBR textures.

6.5. Visualization Videos

We provide additional visualization videos of reconstructions generated with Twiner together with this supplementary material. We visualize the predictions from our model under direct sunlight for four different instances of a wide range of object categories in CO3Dv2 ranging from simple shapes like balls to complex objects like motorcycles. Our model is able to provide good quality relightable reconstructions thanks to the combination of real and synthetic data used during training.