Supplementary Material for GOEmbed: Gradient Origin Embeddings for Representation Agnostic 3D Feature Learning

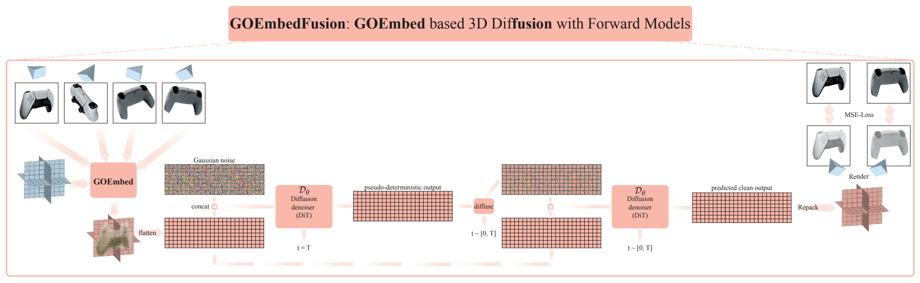
GOEmbedFusion: GOEmbed based 3D Diffusion with Forward Models
GOEmbed
Gaussian noise
concat
flatten
$\mathcal{D}_\theta$ Diffusion denoiser (DiT)
t = T
pseudo-deterministic output
diffuse
t ~ [0, T]
$\mathcal{D}_\theta$ Diffusion denoiser (DiT)
t ~ [0, T]
predicted clean output
Repack
Render
MSE-Loss

Fig. 1: Illustration diagram for the GOEmbedFusion method. Best viewed zoomed in.

1 Preliminaries

We start with a summary of the required preliminaries for **GONs**: Gradient Origin Networks in sec. 1.1 and **Diffusion with Forward** models in sec. 1.2. Please refer to the papers of Bond-Taylor and Willcocks [8], and Tewari et al. [69] for more details about these methods.

1.1 Preliminaries: GONs (Gradient Origin Networks)

Given a dataset of observed samples $x \sim p_{\text{data}}$, where $x \in \mathbb{R}^m$, a Gradient Origin Network [8] (GON) auto-encodes the input x into reconstructed $\hat{x}$ via a much smaller latent space $z \in \mathbb{R}^k$, where $k \ll m$, without requiring an explicit encoder network. The encoder mapping of the input x to latent z, $F_{\text{enc}} : \mathbb{R}^m \rightarrow \mathbb{R}^k$, is defined as the gradient of the log-likelihood of x under the decoder network $F_{\text{dec}} : \mathbb{R}^k \rightarrow \mathbb{R}^m$ wrt. a known fixed origin latent z_0. In practice, the z_0 is always set to zeros, i.e., $z_0 \in \mathbb{R}^k_{[0]}$ and the mean squared error between $F_{\text{dec}}(z_0)$ and x is used as an estimate of the log-likelihood as follows:

$$\begin{aligned} z &= F_{\text{enc}}(x) \\ &= -\nabla_{z_0} \|x - F_{\text{dec}}(z_0)\|_2^2. \end{aligned} \tag{1}$$

The decoding of z can now simply be obtained as a forward pass of the decoder network F_{dec}:

$$\begin{aligned} \hat{x} &= F_{\text{dec}}(z) \\ &= F_{\text{dec}}(-\nabla_{z_0} \|x - F_{\text{dec}}(z_0)\|_2^2). \end{aligned} \tag{2}$$

Using this encoder-less auto-encoding mechanism, the decoder network can be trained using the standard log-likelihood maximization objective (again estimated via mean squared error):

$$\begin{aligned} L^{\text{MSE}}(x, \hat{x}) &= \|x - \hat{x}\|_2^2 \\ &= \|x - F_{\text{dec}}(-\nabla_{z_0}\|x - F_{\text{dec}}(z_0)\|_2^2)\|_2^2. \end{aligned} \tag{3}$$

In the GON paper [8], Bond-Taylor and Willcocks also proposed a variational version of the GON autoencoder to enable generative sampling of the latent space, a coordinate-based implicit version of the GON as a secondary application, and a multi-step generalisation of the Gradient Origin Networks mechanism. But we only describe the auto-encoder version here, because it is the most relevant one to our proposed GOEmbed encodings (sec 3 of main paper).

1.2 Preliminaries: Diffusion with Forward models

Diffusion models define the generative process through two markov chains, namely forward (diffusion) and reverse (denoising), over the observed data distribution p_{data} [1]. The most popular variant DDPM [27] defines the forward distribution $q(x_{0:T}) = q(x_0, x_1, ..., x_t, ..., x_T)$ as the markov chain of T Gaussian transitions:

$$\begin{aligned} q(x_{0:T}) &= q(x_0)\prod_1^T q(x_t|x_{t-1}) \\ &\text{where,} \\ q(x_t|x_{t-1}) &= \mathcal{N}(\sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I), \end{aligned} \tag{4}$$

where the sequence $(\alpha_{1:T})$ instantiates a monotonically decreasing noise schedule such that the marginal over the last variable approximately equals a standard Gaussian distribution, i.e. $q(x_T|x_{T-1}) \approx q(x_T) \approx \mathcal{N}(0, I)$. Similar to the forward process, the reverse process is also defined as another markov chain of T Gaussian transitions:

$$\begin{aligned} p(x_{T:0}) &= p(x_T)\prod_1^T p_\theta(x_{t-1}|x_t) \\ &\text{where,} \\ p(x_0) &= q(x_0) = p_{\text{data}}(x) \\ p(x_T) &= q(x_T) = \mathcal{N}(0, I) \\ p_\theta(x_{t-1}|x_t) &= \mathcal{N}(\sqrt{\alpha_{t-1}}\mathcal{D}_\theta(x_t, t), (1-\alpha_{t-1})I), \end{aligned} \tag{5}$$

where the mean of the Gaussian is given by a learned timestep-conditioned denoiser network $\mathcal{D}_\theta : \mathcal{R}^m \rightarrow \mathcal{R}^m$. The network $\mathcal{D}_\theta$ can be trained by minimizing

[1] please note that the forward diffusion process here is not to be confused with the forward setting of **Diffusion with Forward** models.

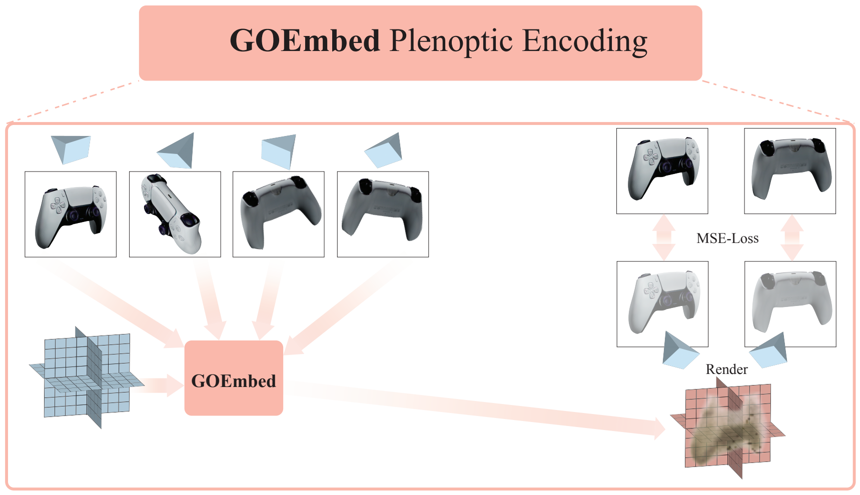

Fig. 2: Illustration diagram for the GOEmbed Plenoptic Encoding experimental setup.

the expectation of the KL divergence between [44] the forward and the reverse transition distributions of a given particular latent x_t over the timesteps t as

$$\mathcal{L}_{\text{diffusion}} = \mathbb{E}_t[D_{KL}(q(x_t|x_{t-1})||p_\theta(x_t|x_{t+1}))], \tag{6}$$

and synthetic samples can be drawn from the trained model by iterating T-times over eq. 5 starting from samples x_T drawn from standard Gaussian distribution $\mathcal{N}(0, I)$. In practice, assuming that the network $D_\theta(x_t, t)$ outputs an estimate of the clean data sample $\hat{x}_0 \approx x_0$ instead of the mean of the Gaussian transition over the previous latent x_{t-1}, the KL divergence objective simplifies to the mean squared error between $\hat{x}_0$ and x_0 [44]

$$\mathcal{L}_{\text{simple}}(x_0, \hat{x}_0) = \mathbb{E}_t[||x_0 - \mathcal{D}_\theta(x_t, t)||_2^2]. \tag{7}$$

DFM models [69] consider the class of *stochastic-inverse* problems, like inverse 3D graphics, which pose a unique challenge where we do not have direct access to the samples $x \sim p_{\text{data}}$, but only have access to partial observations of x obtained through a differentiable forward function $o = \texttt{forward}(x, \phi)$, where ϕ are the parameters for obtaining the observation. The mathematical framework of DFM

learns the conditional distribution $p(x|o, \phi)$ over the unobserved data x given the partial observations (o, ϕ), by modeling the *pushforward* distribution

$$p_\theta(o^{\text{trgt}}|o^{\text{ctxt}}, \phi^{\text{ctxt}}, \phi^{\text{trgt}}). \tag{8}$$

The observations $(o^{\text{ctxt}}, \phi^{\text{ctxt}})$ form the context while $(o^{\text{trgt}}, \phi^{\text{trgt}})$ are the target observations.

$$p_\theta(o^{\text{trgt}}_{0:T} | o^{\text{ctxt}}, \phi^{\text{ctxt}}, \phi^{\text{trgt}}) = \\ p(o^{\text{trgt}}_T) \prod_1^T p_\theta(o^{\text{trgt}}_{t-1} | o^{\text{trgt}}_t, o^{\text{ctxt}}, \phi^{\text{ctxt}}, \phi^{\text{trgt}}) \tag{9}$$

In order to implicitly model the conditional distribution over the data x from the pushforward, each of the learned reverse transition $p_\theta(o^{\text{trgt}}_{t-1} | o^{\text{trgt}}_t, o^{\text{ctxt}}, \phi^{\text{ctxt}}, \phi^{\text{trgt}})$ is defined using the following three steps:

$$x_t = \texttt{denoise}(o^{\text{ctxt}}, o^{\text{trgt}}_t, t, \phi^{\text{ctxt}}, \phi^{\text{trgt}}) \tag{10}$$
$$\hat{o}^{\text{trgt}}_t = \texttt{forward}(x_t, \phi^{\text{trgt}}) \tag{11}$$
$$o^{\text{trgt}}_{t-1} \sim \mathcal{N}(\sqrt{\alpha_{t-1}} \hat{o}^{\text{trgt}}_t, (1 - \alpha_{t-1}) I). \tag{12}$$

Equations 10 and 11 define the same functionality as that of the $\mathcal{D}_\theta$ denoiser network, but also integrate the differentiable `forward` function in the process. The noisy versions of the observed target views o^{trgt}_t can be obtained by the straightforward diffusion process as defined in eq. 4 without any changes. The Forward-Diffusion model can be trained using a conditional version of the KL divergence of eq. 6 as follows:

$$\mathcal{L}_{\text{forward-diffusion}} \\ = \mathbb{E}_t[D_{KL}(q(o^{\text{trgt}}_t | o^{\text{trgt}}_{t-1}) || p_\theta(o^{\text{trgt}}_t | o^{\text{trgt}}_{t+1}, o^{\text{ctxt}}, \phi^{\text{ctxt}}), \phi^{\text{trgt}})].$$

And lastly, synthetic samples of the unobserved underlying variable x can be generated in a auto-regressive manner. First we draw a sample o^{trgt} starting from a given set of $(o^{\text{ctxt}}, \phi^{\text{ctxt}})$ context observations by iterating over equations 10, 11, and 12 T times. A partial estimate of the unobserved underlying variable x^0_0 is obtained from the last denoising step. The complete sample x can be generated by merging n different partial estimates $x^{0:n-1}_0$ by repeating the former process n-times and accumulating the drawn o^{trgt}s together as a new context each time.

2 GOEmbedFusion implementation details

Fig. 1 illustrates the pipeline of the proposed GOEmbedFusion method. Although eqs. 5-5 of the main paper describe the model mathematically, more algorithmic details are provided as follows: We do all the experiments with the Triplane 3D representation. For the GOEmbedFusion model, we use 4 source views and 2 target views for training. We train on the OmniObject3D dataset with a batch size of 16 for $1M$ iterations. Of all the data and metadata provided in the dataset, we make use of the RGB, depth and normal-map renders of the textured-meshes available as part of the scans and the 3D camera parameters of the rendered views. All the rendered maps in the dataset are natively rendered

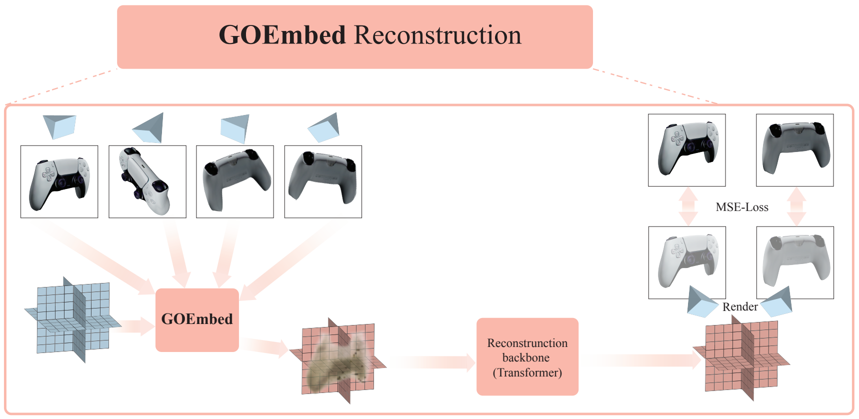

Fig. 3: Illustration diagram for the GOEmbed 3D reconstruction experimental setup.

at the resolution of `[800 x 800]` in Blender [9]. Each of the 3D scan contains ~100 renders from random viewpoints on the upper hemisphere of the 3D centered objects. We use *only* the RGB maps for training the model, while use the depth-maps and normal-maps for computing metrics.

For the diffusion details, we base our code on the guided-diffusion code [1], and use the default values of 0.0001 and 0.02 for the `beta-start` and `beta-end` respectively. For the remaining, we use $T = 1000$ timesteps, `cosine` noise schedule and the `x-start` as the model output. Lastly, we use a single learning rate `5e-5` for the entire span of the training.

3 Discussion on 3D reconstruction

Fig. 2 details the setup of the Plenoptic Encoding experiments, while Fig 3 depicts the setup for the sparse-view 3D reconstruction. As we can see, the only difference between these two setups is The presence of a deep learning reconstruction backbone between the two. For this baseline, we train a regression based network which reconstructs the 3D scene using only the GOEmbed encoded 3D representation as input. Note that this baseline doesn't apply noise anywhere, and hence gives purely deterministic output. As can be seen from the scores of table 3 of the main paper, the reconstruction only (regression) model achieves 27.650 dB, while the GOEmbedFusion gets as close as 26.447 dB PSNR by itself. Lastly, the Transformer architecture used by us is based on the `DiT-XL/2` model of Peebles et al. [54].

4 Checkerboard artifacts in generated 3D samples

Fig. 4 depicts the presence of checkerboard artifacts in the renders of the generated samples by our GOEmbedFusion model. The norm-heatmap of the flattened Triplane features sheds deeper light on the nature of these artifacts. These

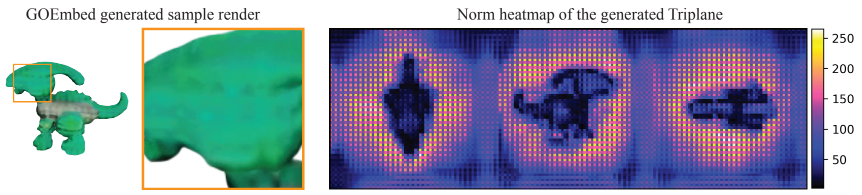
GOEmbed generated sample render
Norm heatmap of the generated Triplane
250
200
150
100
50

Fig. 4: Presence of checkerboard artifacts in the GOEmbedFusion samples.

checkerboard patterns are highly reminiscent of the checkerboard patterns displayed by the recent work of Denoising-VIT [84], which we believe can be rectified using better positional encodings in the DIT network.