PoseDiffusion: Solving Pose Estimation via Diffusion-aided Bundle Adjustment – *Supplementary Material* –

Jianyuan Wang[1,2] jianyuan@robots.ox.ac.uk
Christian Rupprecht[1] chrisr@robots.ox.ac.uk
David Novotny[2] dnovotny@meta.com

[1]Visual Geometry Group, University of Oxford [2]Meta AI

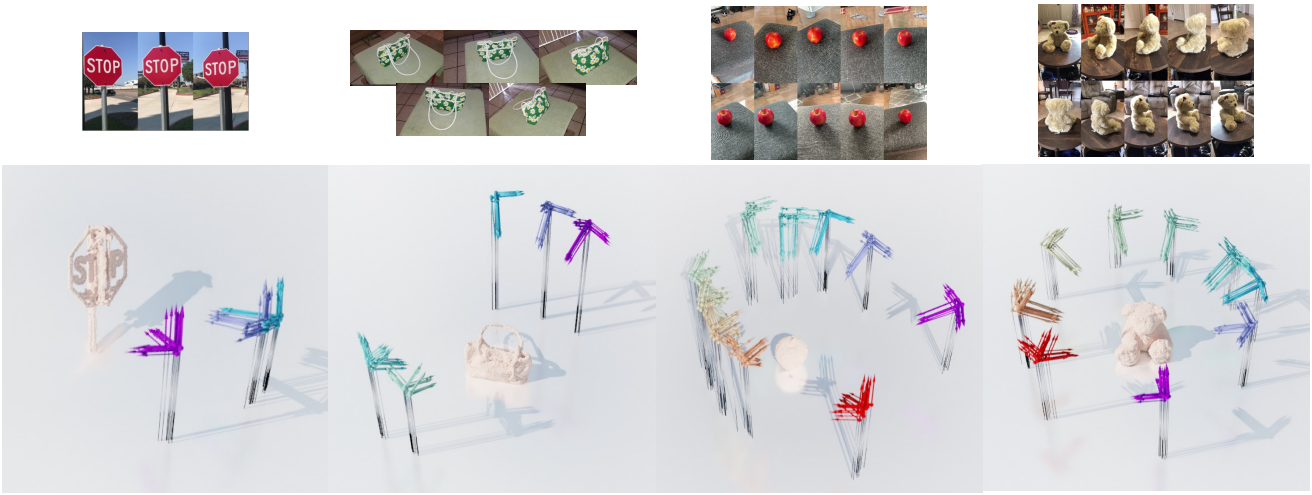

Figure 1: **Pose uncertainty** visualizing multiple samples from $p(x|\mathtt{I})$ conditioned on the same set of input images $\mathtt{I}$. The cameras predicted for the same frame are indicated with identical colors.

Abstract

In this supplementary material, we include (1) more implementation details, (2) the visualization of the uncertainty in the predicted camera poses, (3) additional ablation studies to justify our technical choices, (4) potential avenues for future work, and (5) a video that offers a comprehensive visualization of our method.

1. Implementation Details

In this section, we provide more method details. Additionally, Fig. 2 illustrates a single training-mode forward pass of PoseDiffusion.

Feature Extraction. We use the pretrained DINO ViT-S16 model [1] as our feature extraction backbone. The model and the weight are available in its public repository. We first center-crop the input images and resize them to a resolution of 224×224. Similar to [1], we then respectively resize the images to 1, $\frac{1}{2}$, and $\frac{1}{3}$ of the input resolution (224), and average their features to achieve a multi-scale understanding. The weights of the DINO model are optimized during our training.

Representation and Canonicalization. We represent the camera poses with $\left((\hat{f}^i, \mathbf{q}^i, \mathbf{t}^i)\right)_{i=1}^N$. This representation has a dimensionality of 8: 1 for focal length $\hat{f}$, 4 for quaternion rotation $\mathbf{q}$, and 3 for translation $\mathbf{t}$. As mentioned in the main paper, for each sequence, we randomly chose one input frame as the 'canonical' (pivot) one. Specifically, we reorient the coordinate system of the sequence so that it is centered at the pivot camera. This transformation results in the pivot camera being positioned at the origin with no translation, and with an identity rotation matrix. We explicitly provide this information to the network by utilizing a one-hot pivot flag. Furthermore, in order to prevent over-

fitting to scene-specific translation scales, we normalize the translation vectors by the median norm.

More specifically, given a batch of scene-specific training SfM extrinsics $\{\hat{g}^1, ...\hat{g}^N\} = \mathcal{T}_j \in \mathcal{T}$, the transformer T ingests normalized extrinsics $g^i = s((\hat{g}^\star)^{-1}\hat{g}^i)$ which are expressed relative to a randomly selected pivot camera $\hat{g}^\star \in \mathcal{T}_j$, where $s(\cdot)$ denotes scale normalization which divides the translation component $\mathbf{t}$ of the input $\mathbb{SE}(3)$ transformation by the median of the norms of the pivot-normalized translations. Focal lengths and principal points remain unchanged in the whole process. To avoid the extreme cases brought by canonicalization of outliers, we clamp the input and estimated translation vectors at a maximum absolute value of 100. We also clamp the predicted focal lengths by a maximum value of 20.

Architecture. For the input of the denoiser $\mathcal{D}_\theta$, we concatenate the input poses x_t^i, the diffusion time t, and the feature embeddings $\psi(I^i)$ of the input images I^i. Specifically, we first project the concatenated input poses $x_t^i \in \mathbb{R}^8$ and steps $t \in \mathbb{R}$ to a feature vector with 96 dimensions (dim) by a linear transformation. Next, we concatenate the 96-dim feature vector with x_t^i, t, and 385-dimensional image features $\psi(I^i)$, fed into the denoiser. The image feature embedding $\psi(I^i)$ comprises 384-dim DINO features and the one-dimensional binary pivot camera flag $p^i_{\text{pivot}} \in \{0, 1\}$.

The denoiser $\mathcal{D}_\theta$ adopts a classic Transformer architecture. We use the built-in implementation of PyTorch. Our denoiser has 8 encoder layers and does not use decoder layers. The number of heads is set to 4, and the dimension of the feedforward network is 1024. The output features of the transformer are passed into a two-layer MLP to give the final prediction. The hidden dimension of the final MLP is 128 and the output dimension is 8.

Diffusion Model. We use the PyTorch implementation of DDPM [4]. We set the total number of diffusion sampling steps T as 100. Following the default setting of DDPM, the forward process variance (β_t) increases linearly from 10^{-3} to 0.2. We empirically chose the "x_0 formulation" of DDPM because it exhibits a more stable training and marginally better performance than predicting the noise profile. The other hyperparameters were kept at their default values as per the utilized DDPM codebase.

GGS. The guidance strength s is set adaptively to $s = \min(\alpha \frac{\|\mu_t\|}{\|\nabla p(\mathtt{I}|x)\|}, 1.0)$, with $\alpha = 0.0001$ to stabilize training. We skip the GGS process if no matches were discovered between any pair of input frames.

Training. We train our model on 8 NVIDIA Tesla V100 GPUs, each with 192 images. For each sequence, we randomly conduct color-jitter augmentation to all images in each batch. Additionally, with a probability of 0.15, we randomly turn each training image to its gray-scale form. To ensure stable training, we rescale the optimization gradient so that its norm does not exceed 1.0. The whole training pipeline is implemented using PyTorch3D.

Evaluation. As mentioned, we align the predicted camera poses to the ground truth ones by a single optimal similarity before evaluation, which is implemented by Umeyama's algorithm [5]. The latter aligns the 3D locations of the optical centers of the predicted cameras to the centers of the corresponding ground truth cameras.

NeRF. The training and evaluation of our NeRF experiments leverage the Implicitron framework. Each NeRF model was trained using the default parameters of the framework. We empirically verified that using a single focal length comprising the average over all frame-specific focal length predictions provides better performance. To ensure reconstructibility of the evaluation sequences, we first train NeRF with ground-truth camera poses and, select only the ones where training/evaluation with 8/2 views gives PSNR of 25 or better.

Fundamental Matrix Derivation. Epipolar geometry, *i.e.* the relationship between points and lines of two cameras observing the same scene, can be algebraically represented via the fundamental matrix $F \in \mathbb{R}^{3\times3}$. In more detail, denote (x^i, x^j) the parameters of the camera pair, where $x = (K, g)$ consists of intrinsics $K \subset \mathbb{R}^{3\times3}$ and extrinsics $g \in \mathbb{SE}(3)$. The extrinsics g can be further expressed as a rotation matrix and the translation vector $(R \in \mathbb{SO}(3), \mathbf{t} \in \mathbb{R}^3)$. Using the latter, we define a 3×4 projection matrix $M = K\,[R \mid \mathbf{t}]$.

Assume a point $\tilde{\mathbf{p}}$ in the camera plane defined by M^i. The ray back-projected from $\tilde{\mathbf{p}}$ by M^i can be written as $[M^i]^+\tilde{\mathbf{p}} + \lambda C$, where $[M^i]^+$ is the pseudo-inverse of M^i, and C is the camera center so that $M^i C = \mathbf{0}$. The scalar $\lambda \in \mathbb{R}$ parametrizes the ray. Setting $\lambda = 0$ and $\lambda = \infty$ yields $[M^i]^+\tilde{\mathbf{p}}$ and the camera center C respectively. These two points will be imaged at the second image plane M^j as $M^j[M^i]^+\tilde{\mathbf{p}}$ and M^jC. The epipolar line l^j is defined as the line connecting these two points, *i.e.*, $l^j = (M^jC) \times M^j[M^i]^+\tilde{\mathbf{p}}$. The fundamental matrix F is defined as the mapping from a point in the first image plane to its corresponding epipolar line in the second plane, *i.e.* $l^j = F\tilde{\mathbf{p}}$. Therefore, we obtain $F = (M^jC) \times M^j[M^i]^+$. It is worth noting that the point $\tilde{\mathbf{p}}$ is removed from the formulation of F, because F is the relationship between two image planes, and is constant for all the points in one image plane. For more details, please refer to [2].

2. Ablation Studies and Analysis

Unless otherwise stated, all ablation studies are conducted on CO3Dv2.

Camera Pose Uncertainty. One inherent advantage of utilizing the diffusion model for camera pose estimation is its probabilistic nature. It is well-known that few-view cam-

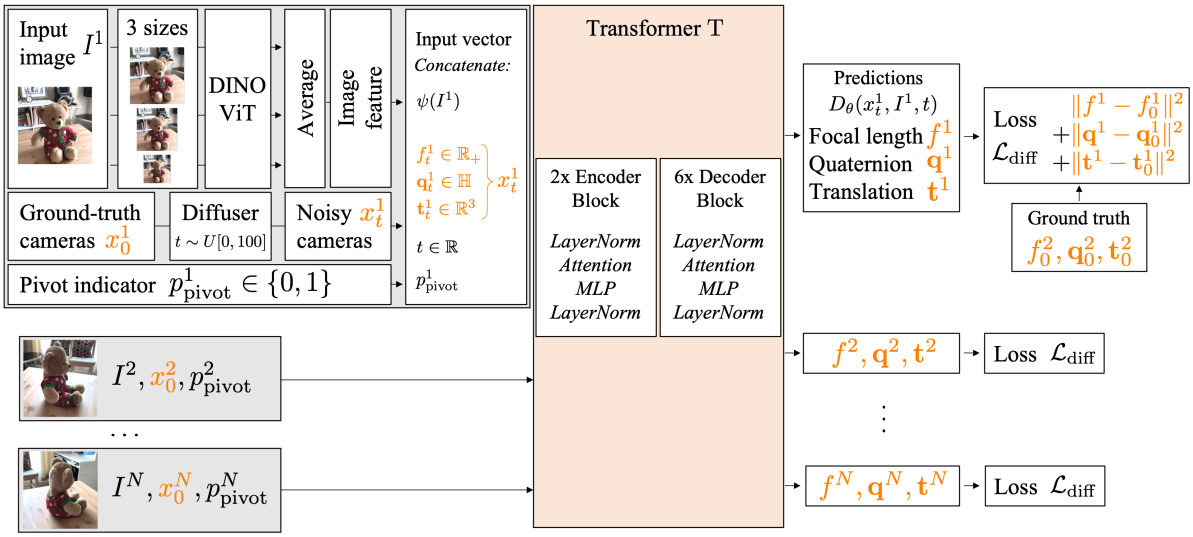

Figure 2: **Overview of our architecture** depicting a single training-mode forward pass.

Backbone	ResNet50 (sup. [3])	ResNet50 (DINO [1])	ViT-S16 (DINO [1])
mAA(30)	63.1	64.3	**66.5**

Table 1: **Performance of different feature backbones**. With other settings unchanged, we evaluate different feature extraction backbones on CO3Dv2.

# diffusion steps T	30	50	100	500
mAA(30)	62.5	66.1	**66.5**	65.3

Table 2: **The effect of the number of sampling steps** T. We evaluate the value of the diffusion sampling steps T from 30 to 500.

	w/o background	w background
mAA(30)	57.0	**66.5**

Table 3: **Effect of background pixels on CO3Dv2**. We compare camera accuracy attained when letting PoseDiffusion observe background pixels (w background) and when using the foreground masks to mask-out the background (w/o background).

era pose estimation is a non-deterministic problem, where multiple pose combinations may be all reasonable for a set of images. We provide a visualization in Fig. 1 to verify that our method can provide several reasonable pose sets x for the same input frames `I`.

Backbone. To explore the effect of upstream feature quality, we try different feature extraction backbones as shown in Tab. 1. ResNet50 trained in a self-supervised manner (DINO ResNet50 [1]) performs better than ResNet50 trained by supervised image classification [3]. The DINO ViT model [1] shows the best performance.

Diffusion Steps. Differently from the original application in image generation (which requires 1000 diffusion steps), the results in Tab. 2 show that a moderate number of sampling steps ($T = 100$) suffice. Therefore, we use $T = 100$ for all the experiments if not further specified.

Importance of Background. We have observed that our method can produce favorable results even when the object is nearly symmetrical. One plausible explanation for this is that the model utilizes cues from the textured background to estimate relative poses (which is valid and desired in SfM). In order to test this hypothesis, we conducted an experiment where we mask the background. In Tab. 3 the performance of the model declines significantly when we replace the background pixels with black, which supports our intuition.

3. Future Work

Looking ahead, we plan to extend the current framework to a self-supervised manner, which would eliminate the need for high-quality ground truth camera poses. This would enable the model to take advantage of numerous Internet data and expand its applicability to a wider range of data distributions. Additionally, our method can serve as a robust initialization for classic Bundle Adjustment frameworks like COLMAP, which could further enhance the accuracy of the pose estimates without the need for the costly and complex iterative SfM process.

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Pro-*

ceedings of the IEEE/CVF international conference on computer vision, pages 9650–9660, 2021. 1, 3

[2] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 2

[5] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991. 2