Appendix

The appendix includes sections as follows:

- **Section A**: Implementation details of both large-scale training and ablation studies.
- **Section B**: Flow evaluation.
- **Section C**: Additional quantitative results.
- **Section D**: Additional qualitative comparisons.

A. Implementation Details

In this section, we provide implementation and training details for all experiments of `Flow3r`.

A.1. Large-scale Training Settings

Full Model – `Flow3r`. Our primary model is obtained by fine-tuning the π^3 [42] backbone and adding a dense flow head. Given N input images, a DINOv2 [29] encoder extracts patch tokens which are processed by a 36-layer multi-view transformer to produce per-image latent representations. These representations are then decoded by two task-specific transformer branches into geometry features and camera features. A point head predicts local pointmaps from the geometry features, while a camera head predicts camera poses from the camera features. In π^3, the predicted local pointmaps are supervised in a permutation-equivariant manner without selecting a reference view. For image I_i in the N-view input, the network predicts a pixel-aligned pointmap $\hat{\mathbf{P}}_i \in \mathbb{R}^{H\times W\times 3}$ in its own camera coordinate system. Due to scale ambiguity, the predicted pointmaps across all views share a single global scale factor s. During training, an optimal scale s^* is estimated from the ground-truth geometry and used to compute the reconstruction error:

$$\mathcal{L}_{\text{points}} = \frac{1}{3NHW}\sum_{i=1}^{N}\sum_{u}\frac{1}{\mathbf{D}_i(u)}\|s^*\hat{\mathbf{P}}_i(u) - \mathbf{P}_i(u)\|_1, \quad (9)$$

where $\hat{\mathbf{P}}_i(u)$ and $\mathbf{P}_i(u)$ denote the predicted and ground-truth 3D points at pixel u, and $\mathbf{D}_i(u)$ is its ground-truth depth. Camera supervision in π^3 is defined on relative poses. Given the predicted camera poses $\hat{\mathbf{T}}_i$ and $\hat{\mathbf{T}}_j$, the relative pose from view j to i is defined as $\hat{\mathbf{T}}_{i\leftarrow j} = \hat{\mathbf{T}}_i^{-1}\hat{\mathbf{T}}_j$. The camera loss averages over all ordered pairs and is a weighted sum of a rotation term and a translation term :

$$\mathcal{L}_{\text{cam}} = \frac{1}{N(N-1)}\sum_{i\neq j}\Big(\mathcal{L}_{\text{rot}}(i,j) + \lambda_{\text{trans}}\mathcal{L}_{\text{trans}}(i,j)\Big), \quad (10)$$

where $\mathcal{L}_{\text{rot}}(i,j)$ is the geodesic distance between relative rotations,

$$\mathcal{L}_{\text{rot}}(i,j) = \arccos\left(\frac{\text{Tr}\big(\mathbf{R}_{i\leftarrow j}^{\top}\hat{\mathbf{R}}_{i\leftarrow j}\big) - 1}{2}\right), \quad (11)$$

and $\mathcal{L}_{\text{trans}}(i,j)$ is a Huber loss on relative translation after resolving scale ambiguity using the same s^* computed from pointmap alignment. We refer the reader to π^3 [42] for further details. Building on this backbone and objectives, `Flow3r` introduces an additional dense factored flow supervision term via the added flow head, while keeping the original π^3 geometry and camera supervision unchanged.

Full Model – `Flow3r-VGGT`. We also instantiate a model variant by fine-tuning the VGGT [38] backbone. It similarly employs a DINOv2 [29] encoder, followed by a 48-layer multi-view transformer with alternating frame-wise and global self-attention blocks that produces camera features and geometry features. These features are then processed by task-specific heads to predict camera poses and scene geometry.

Unlike π^3 [42], VGGT predicts geometry and poses in a global frame – the coordinate frame of the first image in the input. Following the insight of π^3, we supervise the camera and geometry using relative quantities to remove the dependence on an absolute frame. We first normalize the predicted 3D pointmaps using their mean distance to the centroid, and this value serves as the scene scale implicitly learned by the model. For camera supervision, we adopt the same relative rotation loss $\mathcal{L}_{\text{rot}}$ as π^3 (Eq. 11), but replace their relative translation loss with a center loss $\mathcal{L}_{\text{center}}$ directly on the predicted camera centers after computing an optimal alignment, which we empirically find to perform better. Specifically, for the predicted camera centers $\{\hat{\mathbf{c}}_i\}$, we compute an optimal rigid transform $(\mathbf{R}^*, \mathbf{t}^*)$ that minimizes $\sum_i \|\mathbf{R}^*\hat{\mathbf{c}}_i + \mathbf{t}^* - \mathbf{c}_i\|_2^2$, and aligns the predicted centers as $\hat{\mathbf{c}}_i^{\text{aligned}} = \mathbf{R}^*\hat{\mathbf{c}}_i + \mathbf{t}^*$. The camera center loss is defined as:

$$\mathcal{L}_{\text{center}} = \frac{1}{N}\sum_i \big\|\mathbf{c}_i^{\text{aligned}} - \mathbf{c}_i\big\|_1,$$

where N denotes the number of views in the input. For supervising the predicted global pointmaps, we introduce a permutation-equivariant training objective – we estimate the best rigid transform that aligns the predicted pointmap $\hat{\mathbf{P}}_i$ to the ground truth $\mathbf{P}_i$ and computes an ℓ_2 loss on the aligned points:

$$\mathcal{L}_{\text{point}} = \frac{1}{N}\min_{\mathbf{R}^*\in SO(3),\,\mathbf{t}^*\in\mathbb{R}^3}\sum_i \|\mathbf{R}^*\hat{\mathbf{P}}_i + \mathbf{t}^* - \mathbf{P}_i\|_2.$$

To prevent coordinate drift over training, we regularize the mean position of all predicted points to stay centered around the origin:

$$\mathcal{L}_{\text{reg}} = \left\|\frac{1}{N}\sum_i \hat{\mathbf{P}}_i\right\|_2. \quad (12)$$

The depth head is supervised using the confidence-weighted regression loss introduced in DUSt3R [41]. Specifically, for each pixel u, we regress the predicted depth $\hat{\mathbf{D}}_i[u]$ to the ground-truth depth $\mathbf{D}_i[u]$ using the predicted uncertainty $\mathbf{\Sigma}_i[u]$ as a per-pixel confidence weight. The depth loss is:

$$\mathcal{L}_{\text{depth}} = \frac{1}{N}\sum_{i=1}^{N}\left(\Big\|\mathbf{\Sigma}_i \odot (\hat{\mathbf{D}}_i - \mathbf{D}_i)\Big\|_2 - \sum_u \alpha\log\mathbf{\Sigma}_i[u]\right), \quad (13)$$

The total loss for supervising camera and geometric predictions is therefore:

$$\mathcal{L}_{\text{supervised}} = \mathcal{L}_{\text{rot}} + \mathcal{L}_{\text{center}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{point}} + \beta\mathcal{L}_{\text{reg}}. \quad (14)$$

Flow Head. We predict dense flow between a pair of images by fusing source-view geometry features with target-view camera features and decoding the fused representation using a DPT

head [31]. For `Flow3r` (π^3 backbone), the geometry features describe local structure within each view. Consequently, in addition to combining the source-view geometry features with target-view camera features, we also incorporate source-view camera features to transform local point representations into a consistent global geometry. Specifically, we first average the target-view camera features to obtain a global camera token, concatenate it with the source-view geometry features and source-view camera features, and fuse them using a lightweight MLP. We aggregate features from multiple transformer layers and decode the fused representation with a DPT head to predict dense flow. For `Flow3r-VGGT` (VGGT backbone), the predicted geometric elements are defined directly in a global coordinate frame. We therefore concatenate only the source-view geometry features with the target-view camera features and decode them using the same DPT head to produce dense flow.

Training Procedure. We adopt the dynamic batch sizing strategy of π^3. Each GPU processes 24 images, with the batch size varying from 2 to 24 images. Image resolutions are randomly sampled such that the total pixel count lies between 100k and 255k, and the aspect ratio is drawn from $[0.5, 2.0]$. We train the models in multiple stages. For the VGGT backbone, we first fine-tune the model on labeled 3D data without flow supervision to adapt it to the permutation-equivariant formulation described above. For both `Flow3r` and `Flow3r-VGGT`, we initialize the model from the pretrained backbone and train only the factored flow head for 50k steps while freezing the backbone, using 8 H100 GPUs with a learning rate of 5×10^{-5}. After this warm-up stage, we unfreeze the entire model and train end-to-end for another 100k steps on both labeled and unlabeled data, using 8 H100 GPUs with a learning rate of 2×10^{-5}. This two-stage procedure stabilizes geometry learning under full supervision before introducing large-scale flow supervision from unlabeled videos.

A.2. Ablation Studies

For the ablation studies in Sec. 4.1, we use a compact model that preserves the overall architecture of `Flow3r-VGGT` but with reduced capacity for efficiency. Concretely, we adopt a 224×224 DINOv2 backbone and a 24-layer multi-view transformer, while keeping the prediction heads and training objectives unchanged. Despite being significantly smaller, this model remains architecturally compatible with the full variant and supports all ablation analyses. All ablation models are trained from scratch for 160k iterations on 4 H100 GPUs. We use Adam with a learning rate of 2×10^{-5}.

A.3. Evaluating Dynamic Scene Reconstruction

For dynamic datasets such as Kinetics-700 and EPIC-KITCHENS, which lack annotated geometry and camera poses, we generate pseudo labels using MegaSAM [24]. For Kinetics-700, evaluation is performed on a curated subset of 60 sequences. For EPIC-KITCHENS, we process 100 frames per sequence to obtain pseudo annotations.

To fairly assess temporal reasoning under varying motion magnitudes, we adopt a stride-based sampling protocol. For each dataset, we predefine a set of temporal strides, and at evaluation time randomly sample one stride per sequence. The input consists of 8 ordered frames selected according to the sampled stride. While the frames preserve their original temporal order, they are not necessarily temporally adjacent. As a result, smaller strides emphasize short-term motion handling, whereas larger strides challenge the model's ability to maintain long-range dynamic consistency.

B. Flow Evaluation

While our method leverages flow supervision, accurate dense correspondence prediction is not its primary objective. Instead, flow serves as an intermediate supervisory signal to improve visual geometry learning. Owing to the factored formulation, correspondences are predicted by combining geometry and camera representations, rather than by directly matching dense features across views as in tracking-based flow heads. Consequently, the model does not explicitly optimize for pixel-level matching accuracy.

Tab. 6 reports standard flow metrics (AEPE and EPE@5px) on SpatialVID. We observe that the factored formulation achieves lower flow accuracy than tracking-based predictors, yet consistently yields better camera pose estimation and geometry reconstruction. This indicates that the benefit of flow supervision does not stem from precise correspondence prediction per se, but from the geometric constraints it imposes during training.

Model Variant	Data (# Seqs)	RRA↑	RTA↑	CD↓	MSE↓	AEPE↓	EPE@5px↑
`flow-projective`	+ SpatialVID (3K)	61.23	56.12	0.158	0.710	30.24	24.15
`flow-tracking`	+ SpatialVID (3K)	68.56	62.95	0.107	0.628	**18.82**	**43.26**
`flow-factored`	+ SpatialVID (3K)	76.26	68.84	0.103	0.598	23.29	36.42

Table 6. **Pose, geometry, and flow evaluation.**

C. Additional Quantitative Results

Beyond the main results, we provide additional quantitative evaluations on several more datasets (see Tables 8, 9, 10, 11, 12, 13, 14, 15). These include the datasets from the main text, for which we report all relevant metrics, as well as several representative static-scene benchmarks such as 7-Scenes [36], ScanNet [8], CO3Dv2 [32], and NRGBD [2]. For camera evaluation, we follow the standard protocols for each setting: static-scene datasets are evaluated using Relative Rotation Accuracy (RRA), Relative Translation Accuracy (RTA), and the Area Under the Curve which combines the first two metrics, while dynamic-scene datasets are evaluated using Absolute Trajectory Error (ATE), Relative Pose Error for translation (RPE trans), and Relative Pose Error for rotation (RPE rot). For geometry evaluation, we adopt the same metrics for both static and dynamic datasets, reporting Accuracy, Completeness, Chamfer Distance, Mean-Squared Error, and F-score at 2% and 5% thresholds. Across these results, we observe that `Flow3r` shows a general improvement over the baselines, including methods trained with substantially more labeled data (*e.g.*, the non-public synthetic data for training π^3 [42]).

Efficacy of Pseudo-labeled 3D Supervision. A natural alternative for leveraging unlabeled video data is to directly generate pseudo ground-truth 3D geometry and camera poses using off-the-shelf optimization-based systems such as MegaSAM [24]. To examine this possibility, we conduct an ablation study in which

we generate pseudo 3D and pose labels for 3K SpatialVID sequences using MegaSAM. As shown in Tab. 7, training with these MegaSAM-generated pseudo-labels – together with 1K labeled OmniWorld sequences – performs worse than our proposed factored flow supervision. This performance gap suggests that directly regressing 3D geometry and poses from optimization-based 'teachers' may introduce substantial noise and artifacts. In contrast, our formulation provides a cleaner and more effective supervisory signal for leveraging unlabeled video data. Moreover, generating pseudo-labels with MegaSAM is computationally expensive and difficult to scale. In comparison, our flow-based supervision is not only more effective but also easier to deploy across diverse unlabeled datasets.

Data		Metrics			
Labeled (# Seqs)	**Unlabeled (# Seqs)**	**RRA↑**	**RTA↑**	**CD↓**	**MSE↓**
OmniWorld (1K)	-	66.01	62.37	0.105	0.637
OmniWorld (1K)	SpatialVID (3K)	76.26	68.84	0.103	0.598
OmniWorld (1K)+SpatialVID (3K)	-	75.99	65.30	0.108	0.625
OmniWorld (4K)	-	78.68	70.26	0.080	0.565

Table 7. **Ablating pseudo-labeled 3D as supervision.**

D. Additional Qualitative Comparisons

We evaluate our model on a diverse collection of in-the-wild videos spanning both static and dynamic scenes, including everyday scenarios with people, animals, vehicles, and complex background clutter. As shown in Fig. 7 and Fig. 8, across these varied examples, our method consistently produces competitive or superior 3D geometry compared to baseline methods. We additionally provide full point-cloud visualizations in video format on the project website.

Methods	Kinetics700								
	ATE↓	RPE trans↓	RPE rot↓	Acc.↓	Comp.↓	CD↓	MSE↓	f-score@2%↑	f-score@5%↑
DUSt3R	0.045	0.063	9.343	0.083	0.106	0.095	0.366	0.317	0.533
CUT3R	0.019	0.027	1.988	0.070	**0.076**	**0.073**	0.303	0.352	0.573
VGGT	0.027	0.038	1.392	0.088	0.120	0.104	0.347	0.258	0.479
π^3	0.016	0.023	1.006	**0.059**	0.097	0.078	0.267	0.347	0.585
Flow3r	**0.013**	**0.018**	**0.830**	0.062	0.092	0.077	**0.256**	**0.371**	**0.599**

Table 8. **Comparison on Kinetics700.**

Methods	EPIC-KITCHENS								
	ATE↓	RPE trans↓	RPE rot↓	Acc.↓	Comp.↓	CD↓	MSE↓	f-score@2%↑	f-score@5%↑
DUSt3R	0.077	0.110	8.492	0.100	0.092	0.096	0.312	0.385	0.528
CUT3R	0.056	0.081	4.709	0.085	0.095	0.090	0.338	0.297	0.493
VGGT	**0.032**	0.049	3.025	**0.061**	0.069	0.065	0.220	0.415	0.617
π^3	**0.032**	0.043	3.025	0.069	0.058	**0.059**	0.200	**0.459**	0.620
Flow3r	**0.032**	**0.037**	**2.729**	0.066	**0.056**	**0.059**	**0.199**	0.339	**0.622**

Table 9. **Comparison on EPIC-KITCHENS.**

Methods	Sintel								
	ATE↓	RPE trans↓	RPE rot↓	Acc.↓	Comp.↓	CD↓	MSE↓	f-score@2%↑	f-score@5%↑
DUSt3R	0.152	0.179	15.166	0.255	0.224	0.240	0.622	0.124	0.271
CUT3R	0.111	0.128	1.998	0.221	0.269	0.245	0.676	0.111	0.217
VGGT	0.090	0.086	1.220	0.217	0.227	0.222	0.595	0.105	0.311
π^3	0.060	0.066	1.122	0.189	0.191	0.190	0.523	0.141	0.317
Flow3r	**0.048**	**0.058**	**0.920**	**0.153**	**0.104**	**0.129**	**0.426**	**0.201**	**0.404**

Table 10. **Comparison on Sintel.**

Methods	Bonn								
	ATE↓	RPE trans↓	RPE rot↓	Acc.↓	Comp.↓	CD↓	MSE↓	f-score@2%↑	f-score@5%↑
DUSt3R	0.055	0.113	6.384	0.029	0.037	0.033	0.116	0.546	0.800
CUT3R	0.021	0.095	6.349	0.018	0.023	0.021	0.088	0.658	0.899
VGGT	0.011	0.096	**6.021**	0.016	0.028	0.021	0.082	0.644	0.884
π^3	0.016	0.095	6.240	0.014	0.025	0.019	0.076	0.669	0.905
Flow3r	**0.009**	**0.094**	6.262	**0.009**	**0.018**	**0.013**	**0.052**	**0.801**	**0.954**

Table 11. **Comparison on Bonn.**

Methods	7-scenes								
	RRA@30↑	RTA@30↑	AUC@30↑	Acc.↓	Comp.↓	CD↓	MSE↓	f-score@2%↑	f-score@5%↑
DUSt3R	100.0	76.59	55.07	0.047	0.053	0.050	0.170	0.457	0.714
CUT3R	100.0	81.15	59.72	0.055	0.049	0.052	0.183	0.457	0.695
VGGT	100.0	86.51	69.77	0.052	0.056	0.054	0.182	0.395	0.665
π^3	100.0	87.69	71.80	0.039	0.045	0.042	0.169	0.473	0.737
Flow3r	100.0	**91.66**	**75.76**	**0.025**	**0.016**	**0.020**	**0.102**	**0.729**	**0.807**

Table 12. **Comparison on 7-scenes.**

Methods	NRGBD								
	RRA@30↑	RTA@30↑	AUC@30↑	Acc.↓	Comp.↓	CD↓	MSE↓	f-score@2%↑	f-score@5%↑
DUSt3R	100.0	93.25	76.04	0.031	0.024	0.027	0.063	0.662	0.865
CUT3R	100.0	95.63	76.90	0.042	0.019	0.030	0.075	0.549	0.808
VGGT	100.0	99.21	93.13	0.015	0.010	0.012	0.032	0.833	0.964
π^3	100.0	99.20	90.40	0.010	0.007	0.008	0.021	0.910	0.983
Flow3r	100.0	**99.60**	**94.40**	**0.008**	**0.005**	**0.007**	**0.018**	**0.938**	**0.992**

Table 13. **Comparison on NRGBD.**

Methods	Scannet								
	RRA@30↑	RTA@30↑	AUC@30↑	Acc.↓	Comp.↓	CD↓	MSE↓	f-score@2%↑	f-score@5%↑
DUSt3R	**100.0**	57.14	31.56	0.031	0.032	0.032	0.092	0.591	0.831
CUT3R	99.39	71.39	42.30	0.053	0.034	0.043	0.132	0.499	0.740
VGGT	**100.0**	**93.71**	**71.37**	**0.015**	0.019	0.017	**0.053**	0.769	0.931
π^3	99.75	91.14	69.39	0.016	0.022	0.019	0.058	0.762	0.930
Flow3r	**100.0**	92.89	71.00	**0.015**	**0.018**	**0.016**	**0.053**	**0.780**	**0.943**

Table 14. **Comparison on Scannet.**

Methods	Co3Dv2								
	RRA@30↑	RTA@30↑	AUC@30↑	Acc.↓	Comp.↓	CD↓	MSE↓	f-score@2%↑	f-score@5%↑
DUSt3R	97.07	90.91	66.83	0.036	0.069	**0.033**	0.223	0.567	0.783
CUT3R	93.85	90.68	68.11	0.047	0.082	0.064	0.278	0.507	0.737
VGGT	98.41	97.27	87.62	0.022	**0.051**	0.036	0.151	0.707	0.874
π^3	98.82	97.49	**90.53**	0.022	0.052	0.037	0.151	0.707	0.874
Flow3r	**98.84**	**97.62**	90.41	**0.020**	0.052	0.037	**0.150**	**0.709**	**0.876**

Table 15. **Comparison on Co3Dv2.**

Figure 7. **More qualitative results on dynamic scenes.** `Flow3r` effectively handles dynamic components, yielding cleaner geometry than prior feed-forward methods.

Figure 8. **More qualitative results on static scenes and interaction videos.** The first six examples show static scenes and the last two examples feature interaction videos. `Flow3r` tends to produce more stable geometry under motion, though some challenging cases still show noticeable artifacts.