

Point4Cast: Streaming Dynamic Scene Reconstruction and Forecasting

Supplementary Material

We begin this supplementary document by providing details of the datasets used for training in Section A. In Section B, we present additional architectural specifications of our proposed model. Next, in Section C, we report inference speed comparisons between our approach and competing baselines for 3D point map reconstruction. We further include, in Section D, additional ablation studies that examine the influence of our time-conditioning strategy on overall performance. Finally, in Section E and Section F, we provide qualitative results for 3D point map reconstruction and forecasting, camera pose forecasting, and 3D point tracks across a diverse set of scenes. For the challenging task of 3D point map reconstruction and forecasting, we also present qualitative comparisons against state-of-the-art baselines.

The following summarizes the supplementary materials we present:

1. Training details, including those of the datasets used for training our model as well as loss plots.
2. Additional details about the network architecture of *Point4Cast* including discussion of adaptations needed to incorporate popular backbones into our framework.
3. Runtime comparison of different competing methods for 3D point map reconstruction.
4. Ablation studies comparing the different time conditioning mechanisms.
5. Qualitative results comparing *Point4Cast* with different competing methods on samples from an in-domain dataset as well as from unseen, online videos for the task of 3D point map reconstruction/forecasting.
6. Qualitative results of camera pose estimates and 3D point tracks obtained by our method.

In addition, in the supplementary bundle, we provide: a video showing examples of reconstruction/forecasting of 3D point maps and recovered camera poses on the chosen dataset as well as on videos captured in the wild, by *Point4Cast* and other competing methods, compiled together in *supplementary_video.mp4*.

A. Details of Training Datasets

Table A. Description of training datasets used for *Point4Cast*.

Dataset Name	# Videos	Total # Frames	Frame Resolution
Kubric [1]	1,000	~24,000	512 × 512
PointOdyssey [9]	100	~150,000	960 × 540
Stereo4D [2]	2,000	~1,600,000	512 × 512
Custom synthetic dataset	2,000	~240,000	960 × 540

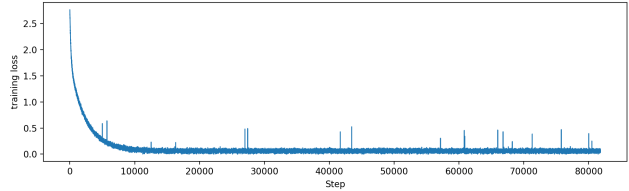


Figure A. Training Loss on the PointOdyssey dataset.

In order to equip, *Point4Cast* with strong inductive priors for online 3D point map reconstruction and forecasting of *dynamic scenes*, beginning with pre-trained modules of popular feed-forward 3D point map reconstruction frameworks [6, 7], we train our model on four diverse datasets of dynamic scenes: (i) Kubric [1], (ii) PointOdyssey [9], (iii) Stereo4D [2], and (iv) an additional synthetic dataset which we create by using Blender - coupling chosen Mixamo¹ motion assets with scene elements from BlenderKit².

The reuse of modules from popular backbones, such as CUT3R [7] or VGGT [6], provides *Point4Cast* with important priors about the geometry of the 3D world across several different types of scenes. Nonetheless, to further finetune our model with 3D priors from complex *dynamic scenes* and to also train the new, untrained modules introduced in our architecture, we leverage the datasets listed in Table A. In particular, we use: (i) The Kubric dataset [1], consisting of ~1,000 indoor, synthetic videos which have metric depth provided with each video. Every video in the dataset is 24 frames long and has frames of 512 x 512 resolution. (ii) The PointOdyssey dataset [9], which comprises of ~100 synthetic videos which also include metric depth per frame. Each video in the dataset is ~1500 frames long and has frames of 960 x 540 resolution. (iii) The Stereo4D dataset [2], containing 2,000 real-world clips, each approximately ~800 frames long, with frames of 512 x 512 resolution. (iv) Finally, we also construct an in-house, synthetic 3D-scene dataset which we construct by using Blender to render chosen Mixamo motion assets (such as human characters and animals) with scene elements (those that constitute the background) obtained from BlenderKit. Following this procedure, we obtain a dataset of 2k videos, where each video is ~120 frames long and has frames of 960 x 540 resolution. This process yields perfect ground-truth metric depth for all frames.

Point4Cast is trained end-to-end with gradients through successive updates of the spacetime representation, similar to Backpropagation Through Time (BPTT), but super-

¹<https://www.mixamo.com/>

²<https://www.blenderkit.com/>

vision after every update shortens gradient paths and avoids long-horizon error accumulation. Moreover, update/readout use attention-based transformers (not sequential RNNs), helping avoid vanishing gradients. Pretrained geometric backbones further stabilize optimization. *Point4Cast* is trained in a single-stage on $8 \times A100$ (80GB), takes ~ 107 h, and converges smoothly (See Fig. A for loss curve on the PointOdyssey dataset).

B. Additional Architectural Details

Several components of *Point4Cast* are initialized using pretrained modules from established backbones for 3D point-map reconstruction from images, such as CUT3R [7] and VGGT [6]. In particular, the image encoder is initialized from the ViT backbone of VGGT, producing 14×14 patches that are embedded into 1024-dimensional tokens. The attention blocks of both the *UpdateTransformer* and the *ReadoutTransformer* are initialized from the Dense Prediction Transformer (DPT) modules of VGGT. The *UpdateTransformer* and *ReadoutTransformer* each consist of 12 attention layers configured as transformer decoders, with 16 multi-head attention (MHA) heads per layer. The prediction heads, Head_{map} and Head_{cam} , follow the architecture of VGGT’s camera head and employ simple 1×1 convolutional layers to map the final 1024-dimensional tokens to the predicted 3D point map and camera parameters.

Incorporating different backbones: The flexible design of *Point4Cast*’s architecture allows for the integration of various backbone networks, enabling us to leverage their inductive biases for 3D point map reconstruction and forecasting. We experiment with two popular backbones: VGGT [6] and CUT3R [7]. When using VGGT as the backbone, we initialize the image encoder with the DINO encoder from VGGT. The *UpdateTransformer* and *ReadoutTransformer* modules are initialized using VGGT’s transformer blocks. In addition, both the Head_{map} and Head_{cam} modules are initialized with the weights of VGGT’s DPT heads. When using CUT3R [7] as the backbone, we initialize the image encoder with the pre-trained ViT encoder from CUT3R. The *UpdateTransformer* and *ReadoutTransformer* modules are both initialized using the transformer decoder from CUT3R.

C. Runtime Comparisons

In Table B, we compare the inference speed (FPS) of our method against several state-of-the-art baselines on the PointOdyssey dataset. Our approach achieves competitive runtime performance, demonstrating that the additional architectural components introduced in our framework do not impose any significant computational overhead.

Table B. **Runtime comparison of competing methods on the PointOdyssey dataset for 3D point map reconstruction.** Higher FPS indicates faster inference. The **best** result is highlighted.

Method	Inference speed (FPS) \uparrow
MonST3R [8]	4.29
VGGT [6]	1.13
CUT3R [7]	23.74
StreamVGGT [10]	17.18
Ours (VGGT backbone)	<u>20.83</u>

Table C. **Ablation study on the choice of Time Conditioning technique on the PointOdyssey dataset.** The design choices for our approach are highlighted.

Query Time Embedding	Conditioning	Acc. \downarrow	Comp. \downarrow
Sinusoidal	Cross-Attention	0.470	0.502
Learned	Cross-Attention	0.437	0.492
Learned	FiLM	0.428	0.472

D. Additional Ablation Results

In this section, we present some additional ablation results of our proposed approach.

D.1. Comparisons of Time-Conditioning Strategies

Table C summarizes the performances of plausible time-encoding and conditioning mechanisms on the PointOdyssey dataset, with the design choice of our model shown in the final row (highlighted in gray). Starting with the input time stamp, a sinusoidal embedding denotes a C -dimensional Positional Encoding (PE) of time. This maybe represented as:

$$e_t = [\sin(\omega_1 t), \cos(\omega_1 t), \dots, \sin(\omega_{C/2} t), \cos(\omega_{C/2} t)] \in \mathbb{R}^C,$$

while a learned embedding directly maps the scalar time to a learned C -dimensional embedding vector.

$$e_t = \text{Embed}(t) \in \mathbb{R}^C,$$

With the time embeddings e_t in place, we explore two approaches towards deriving the time conditioning vector. The first leverages the popular FiLM [3] conditioning technique which may be represented by the equations shown below:

$$\begin{aligned} \gamma &= W_\gamma e_t, \quad \beta = W_\beta e_t, \quad \gamma, \beta \in \mathbb{R}^C, \\ \mathbf{s}^{(t)}[i, :] &= \gamma \odot \frac{\mathbf{w}_k[i, :] - \mu_i}{\sigma_i} + \beta, \quad \forall i \in \{1, \dots, N\} \end{aligned} \quad (1)$$

Alternatively, we also explore the efficacy of a cross attention scheme as shown below:

$$\hat{\mathbf{s}} = \text{MHA}(Q = \mathbf{w}_k, K = e_t, V = e_t), \quad (2)$$

$$\mathbf{s}^{(t)} = \text{FFN}(\text{LN}(\hat{\mathbf{s}})) + \hat{\mathbf{s}}, \quad (3)$$

where MHA denotes Multi-head Attention, Q, K, V denote the query, key, and value of the attention module, FFN denotes a feed-forward network while LN denotes Layer Norm, as is common in transformer modules [5].

As we see from the results in Table C learned time embeddings outperform sinusoidal ones, and FiLM [3] conditioning provides the largest improvements.

D.2. Frame-Rate and Spacetime Representation Sensitivity

Rate	Acc.↓	Comp.↓	Config		Acc.↓	Comp.↓	FPS↑
			Backbone	Stream			
60 FPS	0.415	0.461	None	✓	0.719	0.897	20.83
30 FPS	0.427	0.469	CUT3R	×	0.539	0.662	5.75
			CUT3R	✓	0.484	0.516	22.05
10 FPS	0.474	0.493	VGGT	×	0.671	0.838	1.47
Uneven	0.433	0.473	VGGT	✓	0.438	0.465	<u>20.91</u>

Table D. Time-spacing sensitivity.

Table E. Ablations on streaming versus full video setting on the PointOdyssey dataset.

In order to study the impact of the video frame rate on the reconstruction, we synthesize a 3 s, 60 FPS clip and create variants by down-sampling the clip to 30 and 10 FPS, as well as an unevenly spaced video setting. Table D shows that down-sampling the frame rate leads to performance degradation due to larger inter-frame motions, albeit gracefully. *Point4Cast* remains largely robust to uneven sampling.

We also ablate the performance of *Point4Cast* on the type of backbone it is initialized with and the use of the proposed streaming update versus a per-frame inference without maintaining a persistent spacetime state. The results in Table E show that enabling the spacetime representation consistently improves reconstruction quality across both backbones, while also yielding a substantial efficiency gain in terms of FPS. The VGGT backbone combined with the streaming update achieves the best reconstruction accuracy with high throughput.

E. Qualitative 3D Reconstruction and Forecasting Results

Qualitative 3D point map reconstruction and forecasting results obtained by our method against competing, state-of-the-art baselines (MonST3R [8] and VGGT [6]) are shown in Figure E on the Point Odyssey dataset as well as on unseen, real-world, outdoor videos, as shown in Figure B and Figure C. Since, the competing baselines are not equipped for the novel task of 3D point map forecasting, we use optical flow [4] to first forecast the 2d image frames and then use the baseline approaches to predict the 3D point maps. We see that our proposed approach exhibits denser and more coherent 3D point maps with smoother motion, across both the reconstructed (yellow/blue) and forecasted (green) point maps, compared to the competing methods. Importantly, our proposed approach is able to forecast motions across both rigid (such as cars) and non-rigid (such as hu-

mans) objects, attesting to the effectiveness of our method. Additionally, Figure D shows that the estimated camera poses, obtained by our method, align well with human intuition - attesting to the effectiveness of our approach.

F. Qualitative 3D Point Track Results

One key feature of the design of our approach is that we obtain correspondences between point clouds across time steps, without any additional inference or training. We are thus able to compute tracks of 3D points over time. In Figure F, we show qualitative results for the same, across two different unseen sequences for videos captured in the wild. In both cases, we see that the point tracks are consistent with the motion of the vehicles (the car on the side of the street and the bus respectively.)

References

- [1] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanaprasgasm, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1
- [2] Linyi Jin, Richard Tucker, Zhengqi Li, David Fouhey, Noah Snavely, and Aleksander Holynski. Stereo4d: Learning how things move in 3d from internet stereo videos. *arXiv preprint arXiv:2412.09621*, 2024. 1
- [3] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 2, 3
- [4] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 3
- [5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3
- [6] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada, 2025. Computer Vision Foundation / IEEE. Open Access CVF version. 1, 2, 3
- [7] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10510–10522, 2025. 1, 2

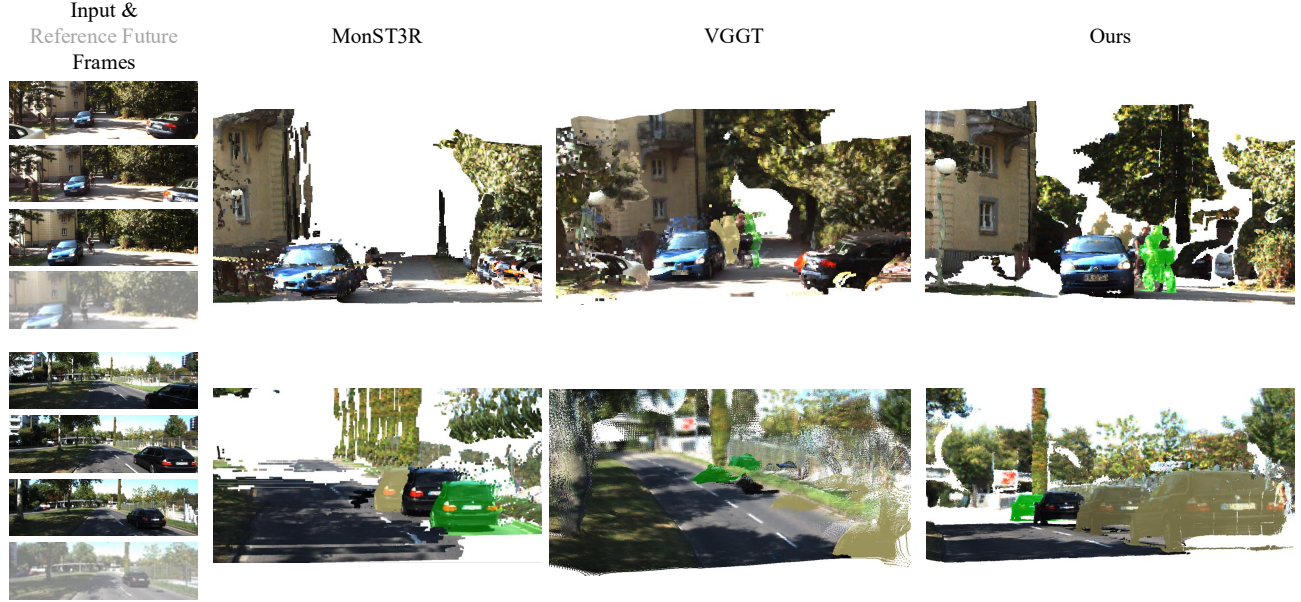


Figure B. **Qualitative comparison of dynamic-scene reconstruction (shown in blue) and forecasting (shown in green) on unseen videos, captured in the wild.** We compare *Point4Cast* (our proposed approach) with MonST3R and VGGT (adapted to use scene flow for forecasting) on challenging outdoor scenes. *Point4Cast* produces more complete and temporally consistent 3D point maps, with sharper geometry, fewer artifacts, and more reasonable future predictions.



Figure C. **Qualitative comparisons on challenging, dynamic-scene reconstruction (shown in blue) and forecasting (shown in green) on unseen videos of outdoor scenes, between *Point4Cast* and VGGT.**

- [8] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. In *The Thirteenth International Conference on Learning Representations*, 2025. 2, 3
- [9] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetstein, and Leonidas J Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19855–19865, 2023. 1, 5
- [10] Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4d visual geometry transformer.

arXiv preprint arXiv:2507.11539, 2025. 2



Figure D. **Qualitative visualization of camera trajectories, obtained by our method, across two unseen videos, captured in-the-wild.**

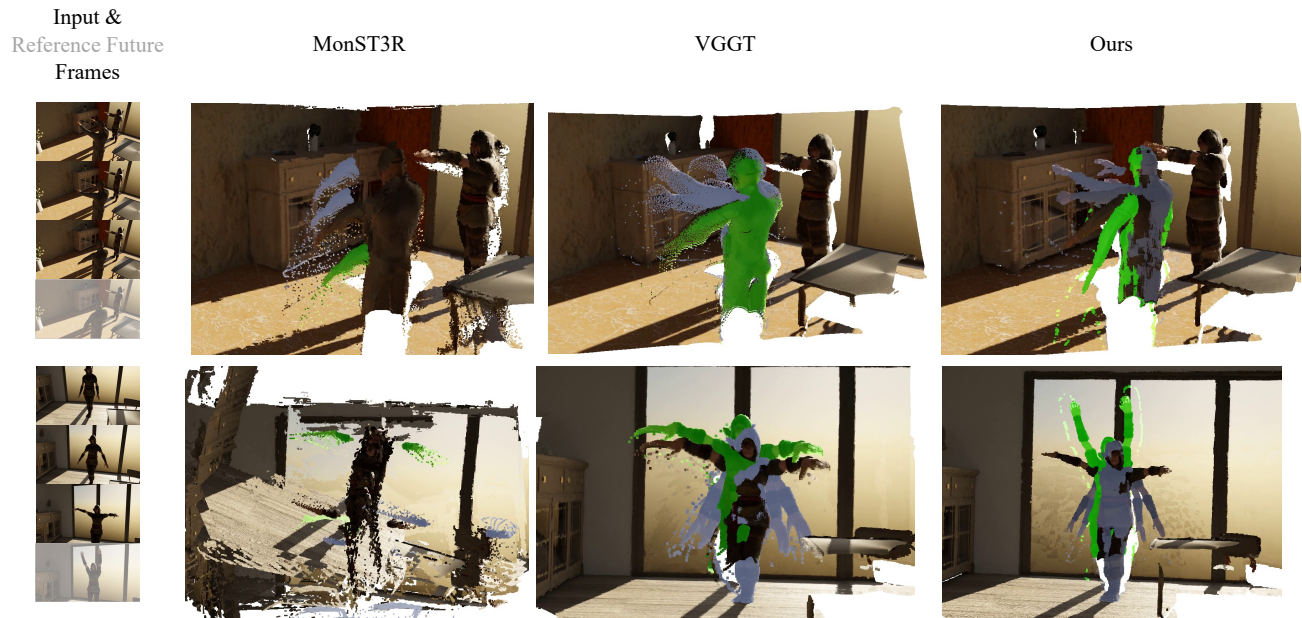


Figure E. **Qualitative comparison of 3D point map reconstruction (shown in blue) and forecasting (shown in green) on the Point Odyssey dataset [9].** We compare *Point4Cast* (our proposed approach) with MonST3R and VGGT (adapted to use scene flow for forecasting) on challenging scenes featuring non-rigid human motion. *Point4Cast* produces more complete and temporally consistent 3D point maps, with sharper geometry, fewer artifacts, and more reasonable future predictions.



Figure F. Qualitative visualization of 3D point tracks, obtained by our method, over time steps.