ChronoGS: Disentangling Invariants and Changes in Multi-Period Scenes

Supplementary Material

A. More Implementation Details

To help readers better understand our method and reproduce our results, we provide additional implementation details in this section. For clarity and completeness, we only summarize the most relevant aspects here. Please refer to our code for further details and full implementation.

A.1. Environment Setup

All experiments are conducted on a Linux workstation equipped with NVIDIA RTX A6000 48GB GPUs and Intel Xeon Gold 6330 (112) @ 3.1 GHz cpus. We use PyTorch 2.4.1 with CUDA 12.1 and Python 3.8.19. We use the gsplat[14] backend and differentiable splatting kernels provided therein. Training and evaluation on large images are performed with the default downsampling strategy of 3DGS[6].

A.2. Anchor Scaffold Construction

We follow the anchor-based scene parameterization introduced in Scaffold-GS[8]. Given a COLMAP[11] sparse reconstruction, let $\mathcal{P} = \{\mathbf{p}_j\}$ denote the sparse SfM point cloud. The goal is to construct a compact yet spatially uniform set of *anchors* $\mathcal{A} = \{a_i\}_{i=1}^{N_a}$ that provide stable geometric support for subsequent Gaussian decoding.

Spatial Grid Generation. The SfM point cloud is first enclosed by an axis-aligned bounding box. We discretize the bounding box into a 3D rectilinear grid with cell size $\Delta = 0.001$ unless otherwise specified. For each grid cell $\mathcal{G}_u$, we check whether it contains at least one SfM point:

$$\mathcal{G}_u \text{ is occupied } \iff \exists\, \mathbf{p}_j \in \mathcal{P} \text{ s.t. } \mathbf{p}_j \in \mathcal{G}_u.$$

Anchor Feature Initialization. Each anchor a_i maintains three types of learnable feature vectors: a period-invariant base feature $f_i^{\text{base}} \in \mathbb{R}^{d_b}$, a set of local period-varying features $f_i^{\text{var}}(t) \in \mathbb{R}^{T \times d_v}$. And a global period feature $g(t) \in \mathbb{R}^{T \times d_g}$ shared across anchors. Unless otherwise specified, all anchors' feature vectors are initialized with zero initialization, which allows the model to learn meaningful structure purely from data. At initialization, the anchor distribution mainly captures the coarse occupied volume of the scene, while the detailed geometry and appearance are progressively refined through the Gaussian clusters decoded from these features during training.

Anchor Filter. To accelerate rendering, we precompute a coarse visibility mask for each anchor. For every camera C_j, an anchor a_i is marked as potentially visible if it lies inside the camera frustum and in front of the camera center. This frustum-based filtering significantly reduces runtime culling overhead and is reused by our method without modification.

Anchor Growth and Pruning. We maintains a dynamic anchor set instead of a fixed scaffold. During training, anchors are periodically added or removed based on simple reconstruction statistics. In the growth step, regions with large residuals or strong gradients are identified in a coarse-to-fine hierarchy, and new anchors are inserted to increase local capacity. In the pruning step, anchors with consistently low visibility, opacity, or gradient magnitude are removed as redundant. This anchor-level adaptation keeps the scaffold compact while allocating more anchors to challenging regions.

A.3. Gaussian Cluster Prediction

Each anchor decodes a local cluster of Gaussian primitives, which collectively represent local shape and appearance. Instead of using a single large MLP to predict all Gaussian parameters, we use three lightweight MLPs to decode geometry, opacity, and appearance separately. We find this design more stable in practice. We use $K = 10$ Gaussians per anchor. Our MLPs employ a lightweight two-layer architecture: a fully connected layer with ReLU activation mapping the input to a hidden representation of dimension d_f, followed by an output layer producing task-specific parameters. The opacity MLP uses $\tanh$ activation, the color MLP uses sigmoid activation, and the covariance MLP has no final activation to allow unconstrained geometric parameters.

A.4. Training Strategy

We train our model using the Adam optimizer and employing differentiated learning rates for different parameter groups. Anchor positions remain fixed throughout training, while offset parameters use an exponential decay schedule from $0.01 \times$ spatial_lr_scale to $0.0001 \times$ spatial_lr_scale, where spatial_lr_scale corresponds to the scale of scene. MLP networks for opacity, covariance, and color prediction use exponential decay schedules with initial learning rates of 0.002, 0.004, and 0.008, decaying to 0.00002, 0.004, and 0.00005, respectively. Geometric parameters, including anchor features, scaling, and rotation use fixed learning rates of 0.0075, 0.007, and 0.002. Our training process consists of three distinct phases: during the initialization phase (iterations 0–500), we initialize model pa-

	Scene Street1			
Settings	PSNR↑	SSIM↑	LPIPS↓	Mem.↓
$\dim(g(t)) = 16 * T.$	24.55	**0.7991**	0.3333	0.42GB
$\dim(g(t)) = 24 * T.$	24.25	0.7948	0.3389	**0.36GB**
$\dim(g(t)) =$ **32** $* T.$	**24.57**	0.7972	**0.3241**	0.52GB
$\dim(g(t)) = 48 * T.$	23.94	0.7790	0.3623	0.58GB
$\dim(g(t)) = 64 * T.$	24.25	0.7921	0.3426	0.43GB

Table 1. **Ablation on the dimension $T \times d_g$ of the global feature $g(t)$.** We evaluate on Scene *Street1*. A small dimension limits scene-level change modeling, while a large one increases memory with no consistent quality gain. The best balance is achieved at $d_g = 32$.

	Scene Aerial1			
Settings	PSNR↑	SSIM↑	LPIPS↓	Mem.↓
$\dim(f_i^{\text{var}}(t)) = 8 * T.$	28.35	0.8478	0.2631	**0.49GB**
$\dim(f_i^{\text{var}}(t)) =$ **16** $* T.$	**28.80**	**0.8562**	**0.2509**	0.63GB
$\dim(f_i^{\text{var}}(t)) = 24 * T.$	28.62	0.8521	0.2573	0.90GB
$\dim(f_i^{\text{var}}(t)) = 32 * T.$	28.61	0.8512	0.2572	0.74GB

Table 2. **Ablation on the dimension $T \times d_v$ of the global feature $f_i^{\text{var}}(t)$.** We evaluate on Scene *Aerial1*. A small dimension limits anchor-level structure modeling, while a large one increases memory with no consistent quality gain. So we choose $d_v = 16 * T$.

rameters and begin standard gradient-based optimization; in the densification phase (iterations 500–20,000), we first collect gradient and opacity statistics from iterations 500–1,500, then perform active densification every 100 iterations from iteration 1,500 onwards, where we grow new anchors at offsets with gradient norms exceeding the threshold $\tau_g = 0.0002$ and visibility above $0.5 \times$ success_threshold $\times$ update_interval, while pruning anchors with accumulated opacity below min_opacity $\times$ anchor_demon and sufficient training statistics; finally, in the pure optimization phase (iterations 20,000–40,000), we disable densification, clean up accumulated statistics, and continue pure parameter optimization without structural changes to the anchor set.

B. More Ablation Studies

B.1. Ablation on Feature Dimensions

We further ablate the dimensions of the three feature components in our period-aware representation: the global period-varying feature $g(t)$, the local period-varying feature $f_i^{\text{var}}(t)$, and the base feature f_i^{base}. These dimensions control the capacity allocated to scene-level period-varying changes, local period-specific variations, and stable, period-invariant structure.

For $g(t)$, as shown in Tab. 1, we sweep $d_g \in \{16, 24, 32, 48, 64\}$ and observe that $d_g = 32$ offers the best quality. For the period-varying feature $f_i^{\text{var}}(t)$, as shown in Tab. 2, we test $d_v \in \{8, 16, 24, 32\}$, where small d restricts temporal expressiveness, and large d increases cost with diminishing returns. For the base feature f_i^{base}, as shown in Tab. 3, we evaluate $d_b \in \{8, 16, 24, 32\}$ and find that moderate dimensions (e.g., $d_b = 16$) provide the best trade-off between accuracy and memory cost.

Overall, the configuration $(d_b, d_v, d_g) = (16, 16, 32)$ provides the best balance between reconstruction quality and storage cost.

	Scene Aerial2			
Settings	PSNR↑	SSIM↑	LPIPS↓	Mem.↓
$\dim(f_i^{\text{base}}) = 8.$	26.00	0.8340	0.2011	**0.87GB**
$\dim(f_i^{\text{base}}) =$ **16**.	**26.50**	0.8467	0.1854	1.10GB
$\dim(f_i^{\text{base}}) = 24.$	26.41	**0.8478**	**0.1850**	1.20GB
$\dim(f_i^{\text{base}}) = 32.$	26.38	0.8452	0.1867	1.30GB

Table 3. **Ablation on the dimension d_b of the global feature f_i^{base}.** We evaluate on Scene *Aerial2*. A small dimension limits scene-level appearance modeling, while a large one exhibits no dominant quality gain but costs more storage. We use $d_b = 16$ for the trade-off.

	Scene Street2		
Settings	PSNR↑	SSIM↑	LPIPS↓
Uni. Init.	26.47	0.8182	0.3011
Gaussian Init.	26.11	0.8155	0.3024
Zero Init.	**26.56**	**0.8220**	**0.2946**

Table 4. **Ablation on anchor feature initialization.** Evaluated on *Street2*. We compare three initialization strategies for anchor features: uniform sampling, Gaussian sampling, and zero initialization. Zero initialization yields the best overall reconstruction quality, suggesting that our framework does not require pre-imposed feature diversity to achieve stable optimization.

B.2. Effect of Anchor Feature Initialization

To study the impact of anchor feature initialization, we evaluate three alternatives to the feature initialization: uniform sampling, Gaussian sampling, and zero initialization. As shown in Tab. 4, zero initialization achieves the best performance across all metrics, indicating that our model can effectively learn discriminative anchor features even when starting from a fully collapsed initialization. Uniform initialization provides competitive performance, while Gaussian initialization shows slightly degraded accuracy. These results demonstrate that our approach is robust to the choice of initialization and does not rely on carefully crafted feature priors.

Scene	Period 0		Period 1		Period 2		Period 3		Total	
	Images	Points	Images	Points	Images	Points	Images	Points	Images	Points
Lawncourt	234	85,655	193	57,884	277	117,595	N/A	N/A	704	261,134
Overstreet	83	35,513	295	235,537	208	132,957	N/A	N/A	586	404,007
BioBuilding	258	120,122	240	143,694	258	171,462	N/A	N/A	756	435,278
Canteen	340	101,523	183	99,915	279	159,250	N/A	N/A	802	360,688
Incubator	202	98,073	199	126,315	194	125,461	N/A	N/A	595	349,849
TriPlot	365	150,896	94	50,081	189	106,937	N/A	N/A	648	307,914
Street1	200	110,591	200	108,932	200	95,038	200	101,074	800	415,635
Street2	200	108,621	200	112,293	200	101,061	200	110,714	800	432,689
Aerial1	200	136,491	200	116,790	200	136,624	200	101,494	800	491,399
Aerial2	200	116,443	200	115,481	200	109,213	200	116,434	800	457,571
Aerial3	200	171,323	200	145,781	200	139,733	200	73,597	800	530,434
Aerial4	200	144,446	200	133,058	200	129,169	200	92,044	800	498,717
Total	**2,682**	**1,379,697**	**2,404**	**1,445,761**	**2,605**	**1,524,500**	**1,200**	**595,357**	**8,891**	**4,945,315**

Table 5. Point Cloud and Image Counts by Scene and Period

Method	PSNR↑	SSIM↑
NeRF[10]	21.36	0.709
MEIL-NeRF[5]	24.05	0.730
CLNeRF[2]	25.45	0.764
ChronoGS	**28.31**	**0.8934**

Table 6. Quantitative comparison on the WAT dataset.

Method	PSNR↑	SSIM↑	LPIPS↓
GsEditor[4]	24.133	0.867	0.143
CLNeRF[2]	24.541	0.658	0.373
CL-NeRF[12]	23.268	0.725	0.290
CL-Splats[1]	28.249	0.930	0.065
ChronoGS	**32.03**	**0.935**	**0.0431**

Table 7. Quantitative comparison on the CL-Splats real-world benchmark.

C. Additional Results on Other Benchmarks

Besides ChronoScene, we additionally evaluate ChronoGS on three public benchmarks used in prior long-term or continual neural rendering works: WAT[12], CL-Splats[1], and NeuSC[7] datasets. These results complement the main-paper evaluation and provide a broader view of generalization under different assumptions about temporal changes.

On the WAT and CL-Splats datasets (Tabs. 6, 7), ChronoGS significantly outperforms CLNeRF and CL-Splats, which primarily target incremental or relatively mild scene changes. In contrast, ChronoGS is designed for a **more challenging setting** with large-scale, long-term variations, explaining its stronger performance on these benchmarks. On the NeuSC benchmark (only the 5Pointz scene is publicly available, Tab. 8), NeuSC focuses on scenarios with **fixed geometry and frequent appearance changes**, while ChronoGS addresses a more complex setting where both geometry and appearance discretely change. As a result, **ChronoGS operates in a larger and more challenging solution space**, leading to competitive but slightly lower performance under NeuSC's specific assumptions.

Method	PSNR↑	SSIM↑	LPIPS↓
NeRFW[9]	17.52	0.545	0.500
HaNeRF[3]	16.82	0.539	0.508
NeRFW-T	19.41	0.611	0.418
HaNeRF-T	17.90	0.585	0.445
NeuSC	**21.32**	**0.745**	**0.274**
ChronoGS	19.35	0.717	0.2761

Table 8. Quantitative comparison on the NeuSC[7] dataset (5Pointz scene). The suffix "-T" denotes the variant that uses time as an additional model input.

D. More Qualitative Results

In this section, we provide additional qualitative comparisons that complement the results shown in the main paper. First, we include comparisons with 3DGS[6] and realtime4DGS[13], which are omitted from the qualitative figures in the main paper due to space constraints. These results further demonstrate the advantages of our method in handling multi-period geometry and appearance variations.

Image Demonstrations. We additionally present figures that contain representative qualitative results for every scene. These visualizations provide a complete overview of reconstruction quality and temporal consistency across the entire dataset. Please refer to Figure 1 and Figure 2 for the qualitative results.

Video Demonstrations. To provide a more intuitive comparison between our method and static or dynamic baselines, we also include video demonstrations in our supplementary material. These videos show how our method faithfully reconstructs complex geometry and appearance changes across periods. However, static methods, when trained on multi-period datasets that contain both geometric and appearance changes, tend to produce blurred reconstructions in the varying regions and fail to model the distinct scene states of each period. Dynamic methods, on the other hand, rely on assumptions of temporal continuity; when applied to discrete periods, they often yield poor intermediate states or introduce significant artifacts.

E. More Information About ChronoScene

To provide a comprehensive overview of the ChronoScene dataset, we present detailed statistics and visualizations that characterize the temporal distribution of both images and point clouds across different scenes. Tab. 5 summarizes the quantitative distribution of images and point clouds for each scene across different periods, revealing the scale and temporal distributions of our dataset.

To further illustrate the spatial-temporal structure of the point cloud data, we visualize the **sparse point cloud distributions** projected onto three orthogonal planes (XY, XZ, and YZ) for each scene, as shown in Figs. 3 to 14. These visualizations reveal how point clouds are distributed across different periods within the same coordinate system, with each period represented by a distinct color (red, yellow, blue, and green for periods 0, 1, 2, and 3, respectively). The projections highlight both the spatial coverage and temporal variations in the point cloud density. These figures demonstrate that our dataset encompasses a diverse range of scenes with varying temporal characteristics, from street scenes to aerial views.

These analyses collectively demonstrate that ChronoScene provides a rich and diverse dataset for temporal 3D scene understanding, with balanced temporal distributions and comprehensive spatial coverage across multiple scene types.

References

[1] Jan Ackermann, Jonas Kulhanek, Shengqu Cai, Haofei Xu, Marc Pollefeys, Gordon Wetzstein, Leonidas J Guibas, and Songyou Peng. Cl-splats: Continual learning of gaussian splatting with local optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7808–7817, 2025. 3

[2] Zhipeng Cai and Matthias Müller. Clnerf: Continual learning meets nerf. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23185–23194, 2023. 3

[3] Xingyu Chen, Qi Zhang, Xiaoyu Li, Yue Chen, Ying Feng, Xuan Wang, and Jue Wang. Hallucinated neural radiance fields in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12943–12952, 2022. 3

[4] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21476–21485, 2024. 3

[5] Jaeyoung Chung, Kanggeon Lee, Sungyong Baik, and Kyoung Mu Lee. Meil-nerf: Memory-efficient incremental learning of neural radiance fields. *IEEE Access*, 2025. 3

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 3

[7] Haotong Lin, Qianqian Wang, Ruojin Cai, Sida Peng, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. Neural scene chronology. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20752–20761, 2023. 3

[8] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 1

[9] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7210–7219, 2021. 3

[10] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 3

[11] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[12] Xiuzhe Wu, Peng Dai, Weipeng Deng, Handi Chen, Yang Wu, Yan-Pei Cao, Ying Shan, and Xiaojuan Qi. Cl-nerf: continual learning of neural radiance fields for evolving scene representation. *Advances in Neural Information Processing Systems*, 36:34426–34438, 2023. 3

[13] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 3

Figure 1. **Qualitative comparison on real scenes of ChronoScene.** We compare ChronoGS with representative baselines. Each row shows novel-view renderings of different methods, along with ground truth. Static models trained on mixed multi-period data produce ghosting and appearance blending due to inconsistent geometry, while dynamic methods that assume continuous motion fail under the large temporal gaps, causing incorrect geometry interpolation. **ChronoGS** faithfully reconstructs period-specific geometry and appearance, preserving sharp, temporally consistent details across long-term changes.

[14] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025. 1

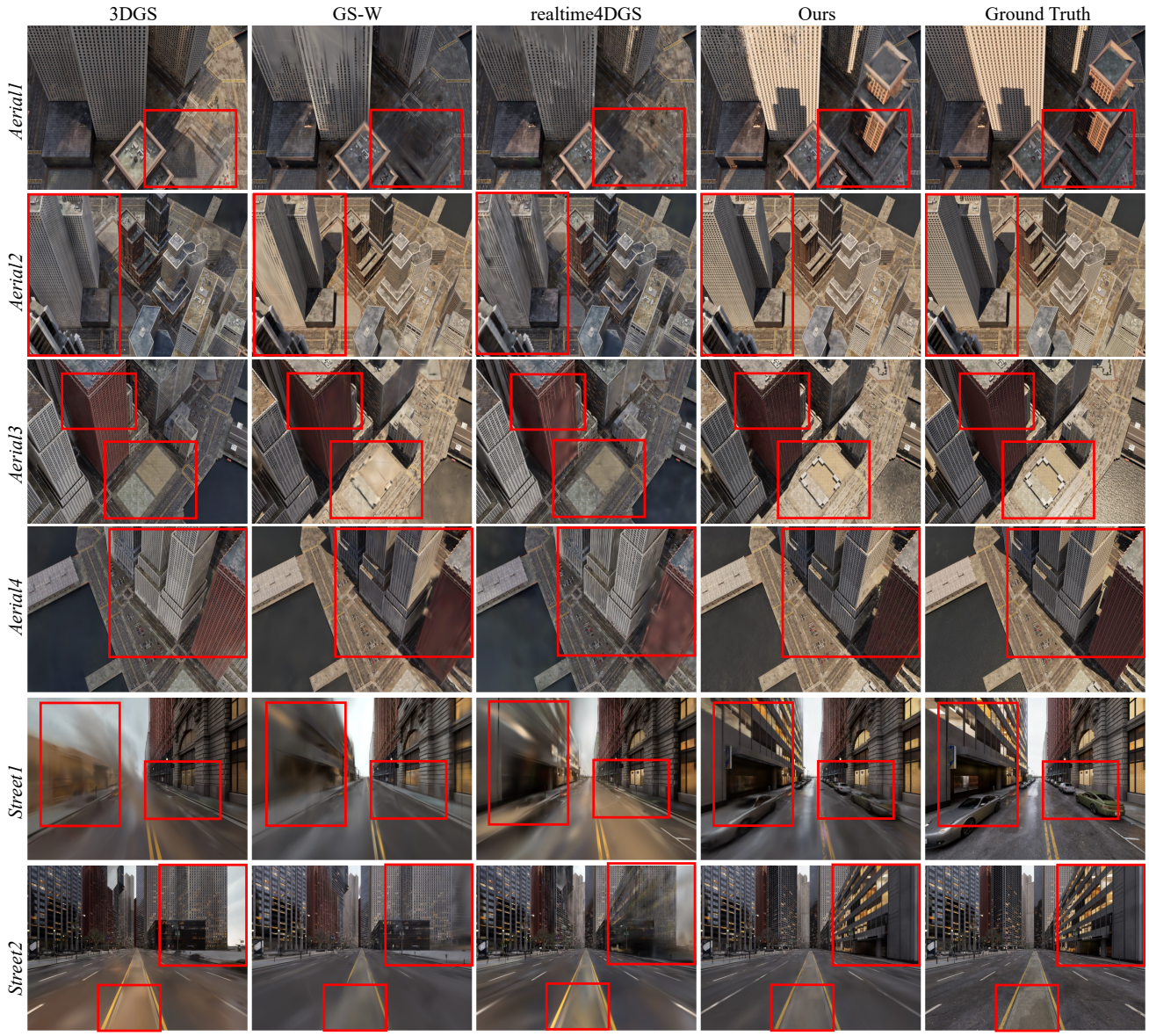

Figure 2. **Qualitative comparison on synthetic scenes of ChronoScene.** We compare ChronoGS with representative baselines. Each row shows novel-view renderings of different methods, along with ground truth. Static models trained on mixed multi-period data produce ghosting and appearance blending due to inconsistent geometry, while dynamic methods that assume continuous motion fail under the large temporal gaps, causing incorrect geometry interpolation. **ChronoGS** faithfully reconstructs period-specific geometry and appearance, preserving sharp, temporally consistent details across long-term changes.

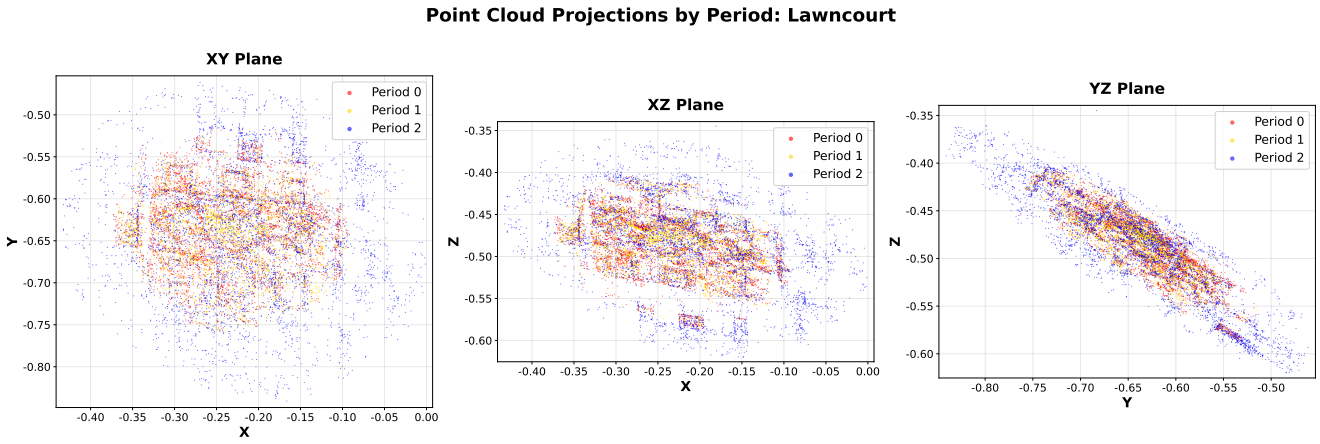

Figure 3. **Point cloud distribution of scene *Lawncourt*.**

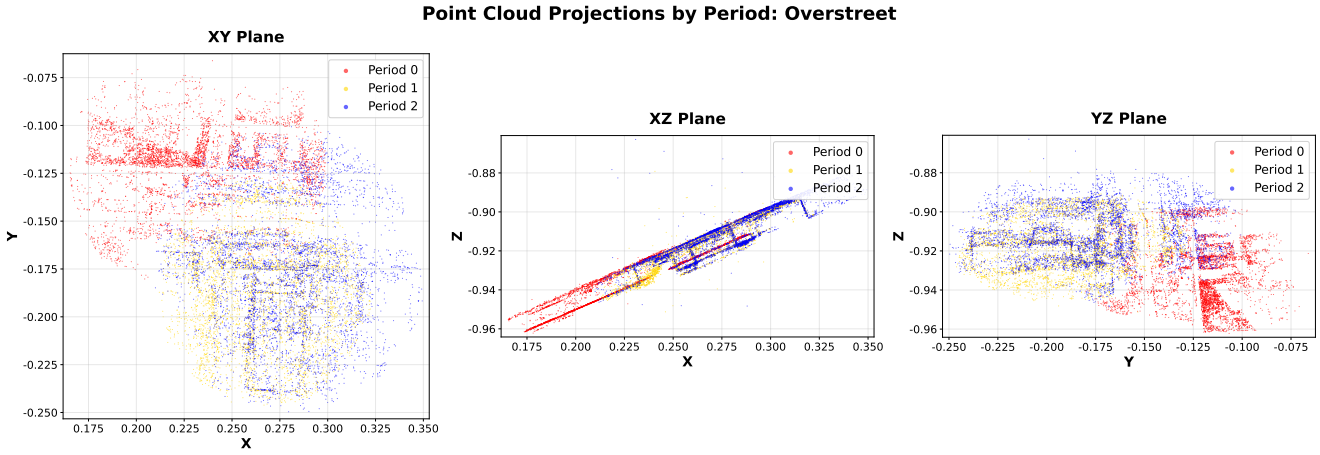

Figure 4. **Point cloud distribution of scene *Overstreet*.**

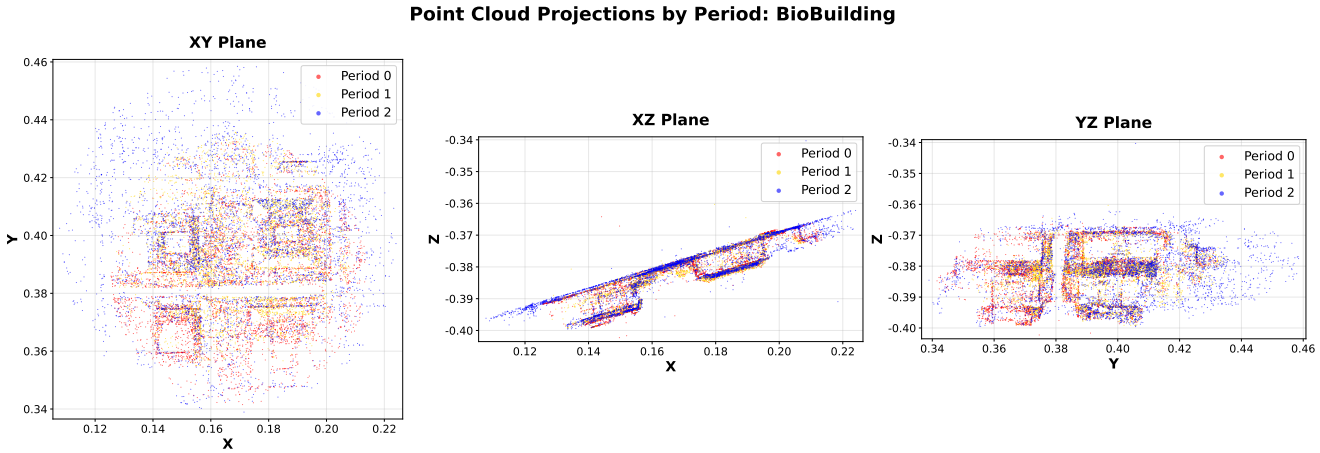

Figure 5. **Point cloud distribution of scene *BioBuilding*.**

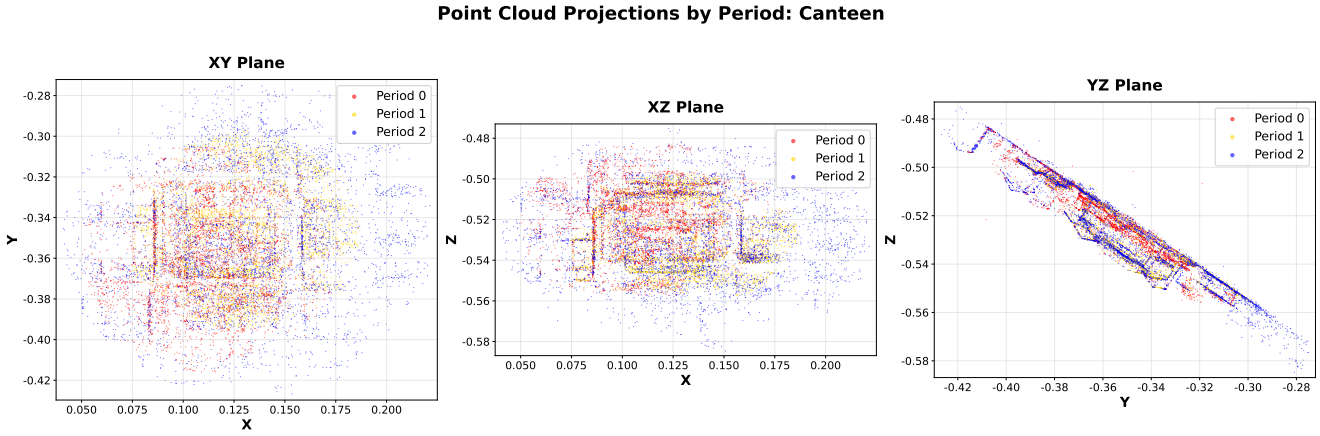

Figure 6. **Point cloud distribution of scene *Canteen*.**

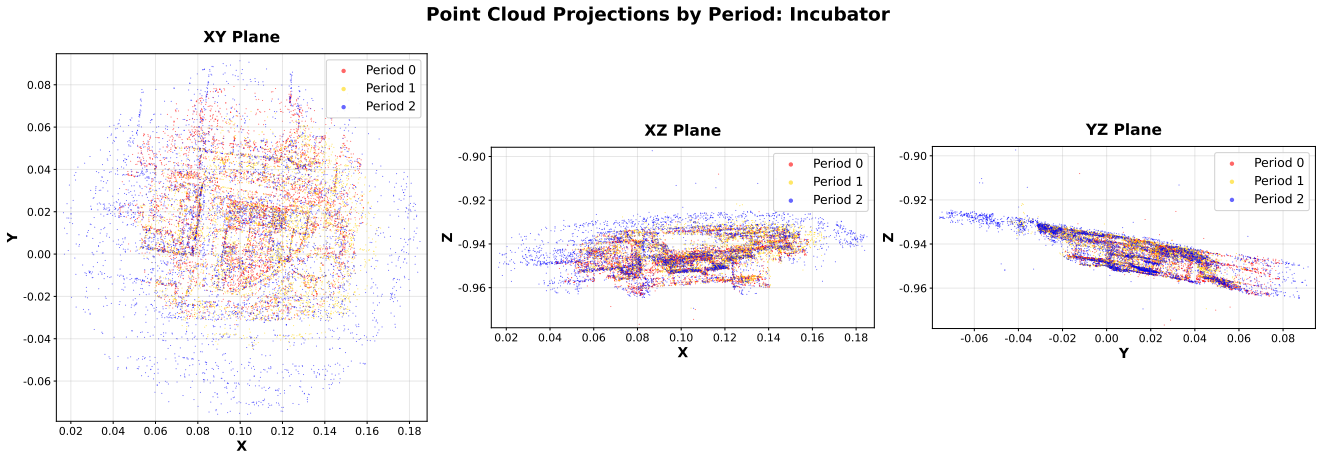

Figure 7. **Point cloud distribution of scene *Incubator*.**

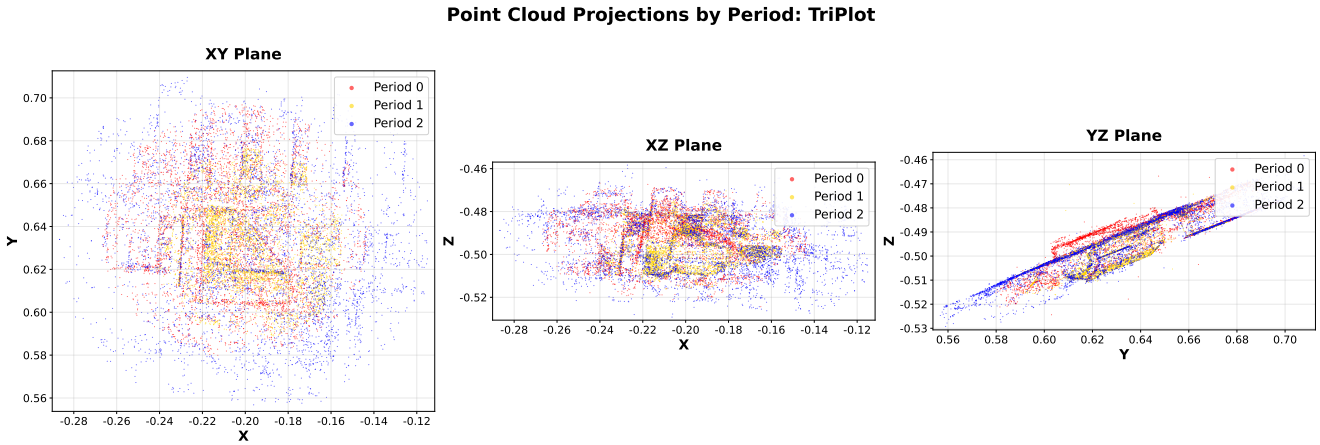

Figure 8. **Point cloud distribution of scene *TriPlot*.**

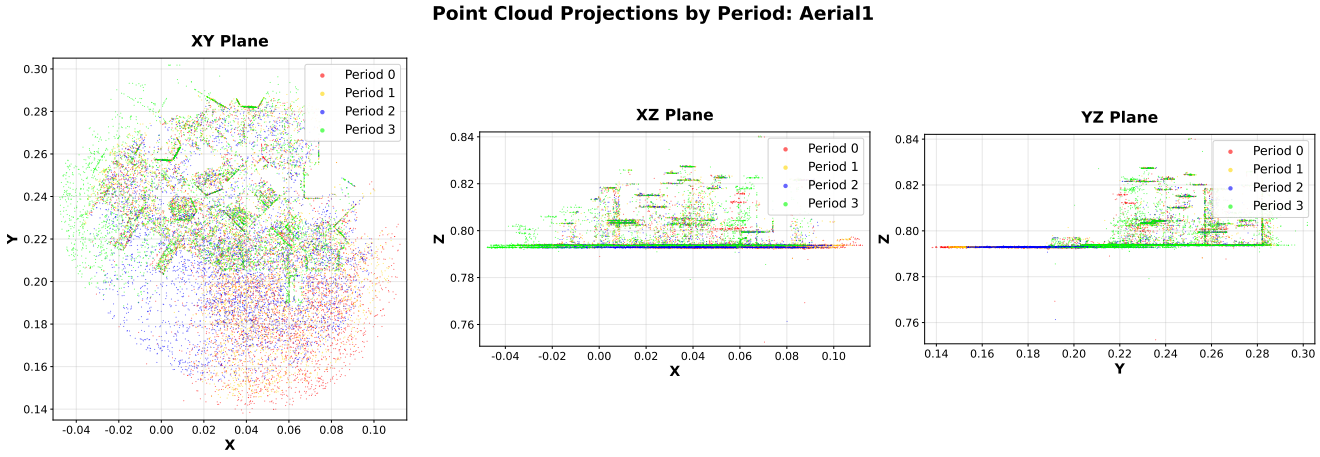

Figure 9. **Point cloud distribution of scene *Aerial1*.**

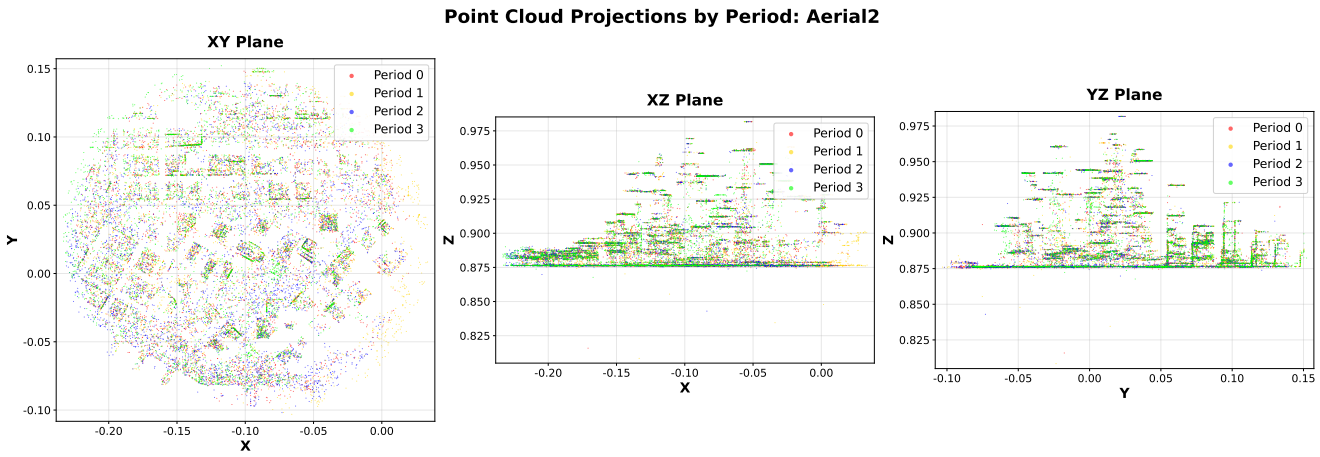

Figure 10. **Point cloud distribution of scene *Aerial2*.**

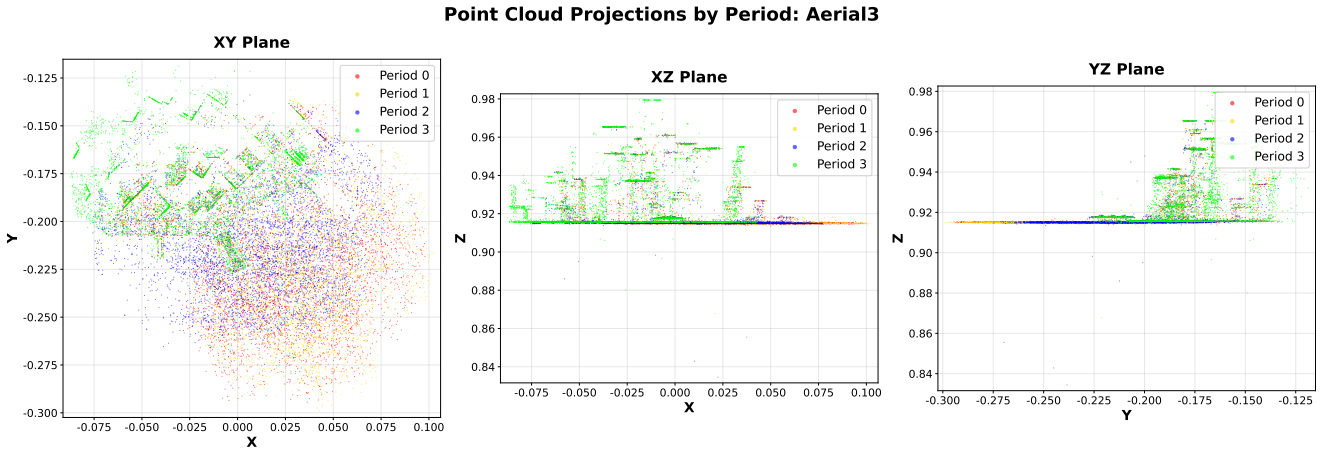

Figure 11. **Point cloud distribution of scene *Aerial3*.**

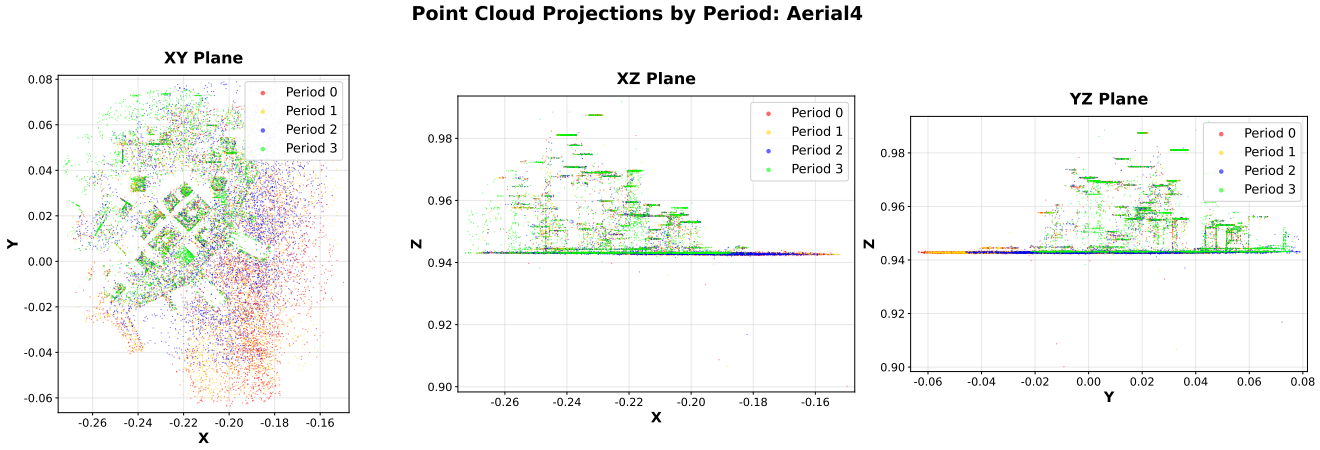

Figure 12. **Point cloud distribution of scene *Aerial4*.**

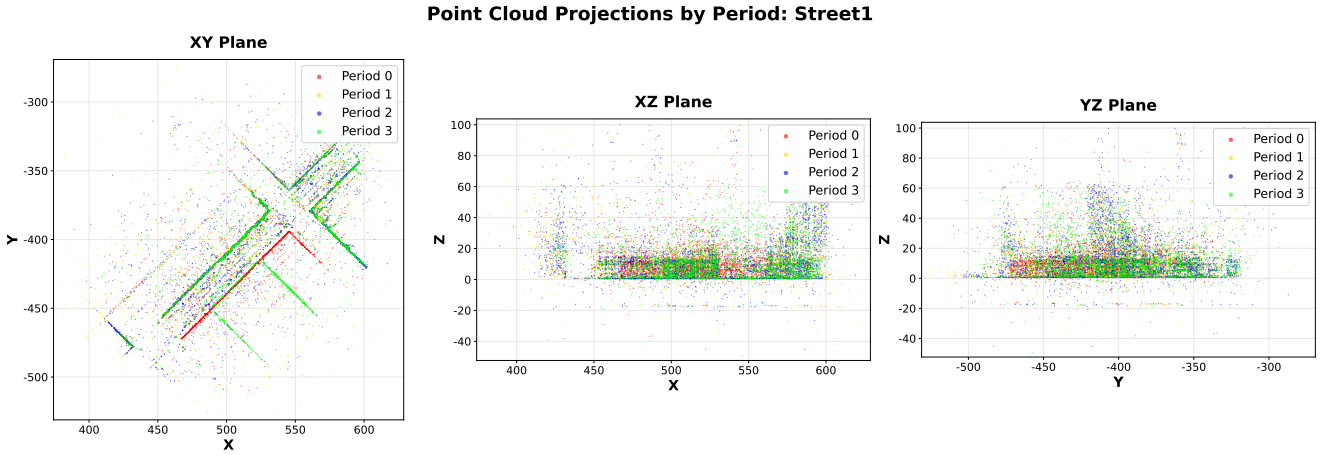

Figure 13. **Point cloud distribution of scene *Street1*.**

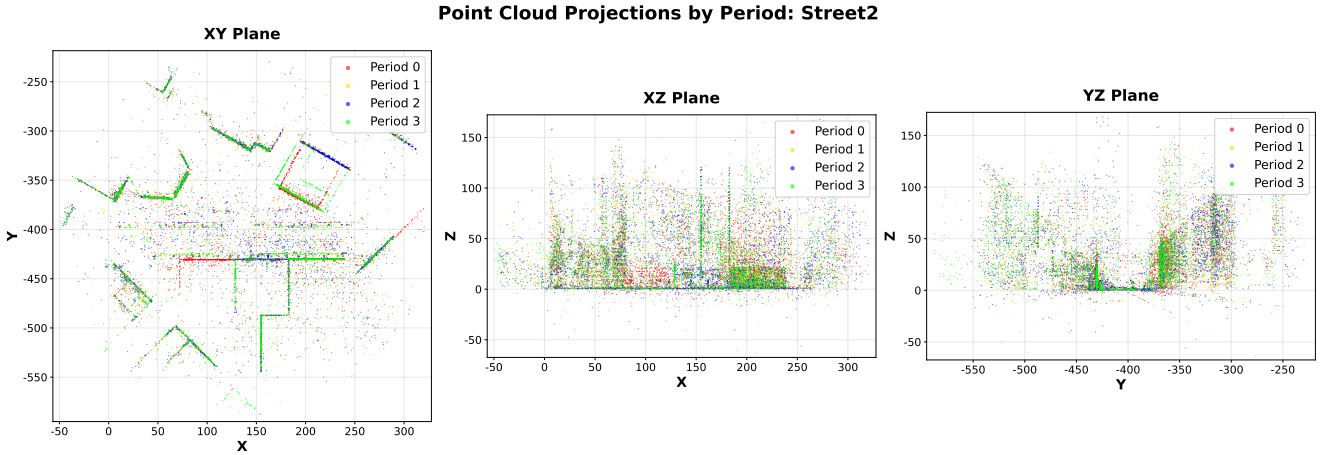

Figure 14. **Point cloud distribution of scene *Street2*.**