TemPCC: Completing Temporal Occlusions in Large Dynamic Point Clouds Captured by Multiple RGB-D Cameras

Andre Mühlenbrock[1], Rene Weller[1] and Gabriel Zachmann[1]

[1]Computer Graphics and Virtual Reality Research Lab, University of Bremen, Germany

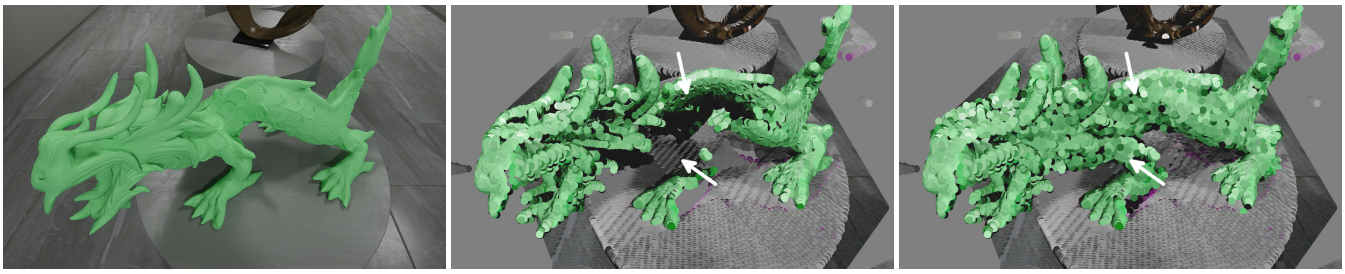

Figure 1: *Our temporal point cloud completion applied to a synthetic dynamic scene where a rotating dragon figure was captured by three virtual RGB-D cameras. On the left is the original scene created in Unreal Engine 5, in the center is the incomplete point cloud (notably due to self-occlusions), and on the right is the completed point cloud using our approach.*

Abstract

We present TemPCC, an approach to complete temporal occlusions in large dynamic point clouds. Our method manages a point set over time, integrates new observations into this set, and predicts the motion of occluded points based on the flow of surrounding visible ones. Unlike existing methods, our approach efficiently handles arbitrarily large point sets with linear complexity, does not reconstruct a canonical representation, and considers only local features. Our tests, performed on an Nvidia GeForce RTX 4090, demonstrate that our approach can complete a frame with 30,000 points in under 30 ms, while, in general, being able to handle point sets exceeding 1,000,000 points. This scalability enables the mitigation of temporal occlusions across entire scenes captured by multi-RGB-D camera setups. Our initial results demonstrate that self-occlusions are effectively completed and successfully generalized to unknown scenes despite limited training data.

CCS Concepts

• ***Computing methodologies*** → ***Point-based models;*** • ***Information systems*** → *Spatial-temporal systems;*

1. Introduction

The complete capture of dynamic 3D scenes in real-time is crucial for many applications, such as point cloud avatars in VR [GCC*20, YGE*21], VR tele-assistance [RYP*21,GJS*21,FMK*22], performance capture systems, or context-aware applications that rely on occlusion cues—such as autonomous surgical lighting systems that dynamically redistribute brightness across unoccluded light modules, where failing to detect transient occlusions leads to suboptimal illumination [MHU*25]. However, even with multiple RGB-D cameras, fully capturing such dynamic scenes in real-time remains challenging. Objects often obstruct each other, the backsides of objects are not captured, and pixels may be invalid.

Depth completion has achieved notable successes in filling holes across general scenes and practical settings, but fails to reconstruct occluded or rear parts of objects. Dynamic scene reconstruction methods, such as DynamicFusion [NFS15], VolumeDeform [IZN*16], and SobolevFusion [SBI18], build a canonical model continuously aligned to new observations, but continue to struggle with arbitrary topology changes in complex multi-person interactions. In contrast, Function4D [YZG*21] employs multi-RGBD-camera setups and fuses only three consecutive frames to mitigate topology-change-induced artifacts, but fills occlusions with learned implicit priors rather than true scene context. Point cloud completion, especially advanced by seminal contributions such as PCN [YKH*18], SnowflakeNet [XWL*23], and PointAttN [WCG*24], leveraging learned geometric priors. While these methods have shown remarkable results in reconstructing individual static ob-

DOI: 10.2312/egs.20251039

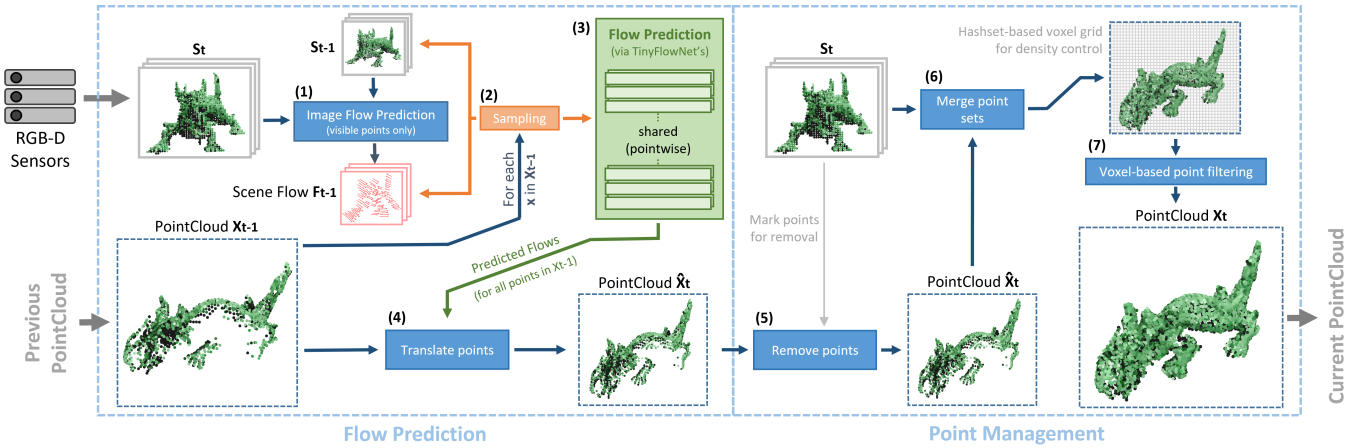

Figure 2: *Conceptual overview of our pipeline: Structured point clouds from multiple RGB-D cameras serve as input. An image flow algorithm (e.g., PDFlow [JSGJC15]) estimates motion for visible points (1). Occluded point flow is predicted using TinyFlowNet (TFN), which samples multiple visible points per occluded one (2) and is trained to predict the motion of occluded points based on visible ones (3), enabling the dynamic update of occluded points (4). On the right, the point set $\hat{X}_t$ is updated by removing duplicates and divergent points (5), integrating new observations S_t (6), and applying voxel grid filtering to ensure uniform density (7), yielding the final point cloud X_t.*

jects consisting of a few thousand points, they fall short in scenarios involving larger scenes with multiple objects that total several hundred thousand points. This highlights the need for methodologies that can scale more effectively to manage more complex point clouds that represent entire dynamic scenes.

In this paper, we introduce a novel, fully temporal approach to point cloud completion that works without learning object geometry and without reconstructing a canonical representation. Our methodology employs tiny, point-wise neural networks that dynamically predict the motion of occluded points based on visible neighboring points captured in camera images. Our pipeline manages a set of points representing the scene over time in which new points are inserted when observed or removed when density constraints are violated. Key innovations of our approach include the ability to manage point clouds of arbitrary sizes and complexities, even extremely large ones, by individually predicting the flow of each hidden point. We achieve linear complexity relative to the number of points, natively preserve color information throughout the pipeline, and allow flexible control over the point cloud density.

2. Our Approach

Pipeline Overview

Our pipeline processes structured point clouds from multiple cameras as input at each time t, denoted as S_t. In addition to managing the structured point clouds of all currently visible points, our pipeline also handles an unordered set X_t that represents all scene points at frame t, including temporally completed ones. To obtain X_t, the unordered set X_{t-1} is transformed according to the flow of the visible points from S_{t-1} to S_t, and then combined with S_t. By avoiding the explicit reconstruction of a canonical model, our approach aims to mitigate artifacts resulting from dynamic topology changes. An overview of the pipeline is illustrated in Figure 2.

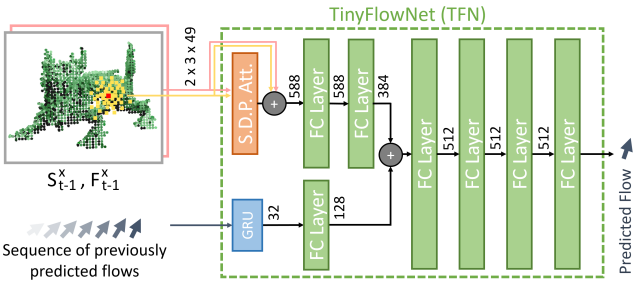

Figure 3: *Our TinyFlowNet (TFN) estimates the flow of a visible or occluded point p using Gaussian-distributed sample points (yellow) from its local neighborhood in the structured point clouds S_{t-1} along with the associated flow F_{t-1}. The sampled points are input relative to point p in camera space, allowing the TFN to be trained independently of camera perspective. Spatial inputs are processed through a Scaled-Dot-Product-Attention (S.D.P. Att.) layer (input and embedding size = 6), while temporal inputs are handled by a Gated Recurrent Unit (GRU). Finally, the data is processed by Fully Connected layers to predict the flow vector.*

Flow Prediction

To facilitate the transformation of occluded points alongside visible ones, our TinyFlowNet (TFN) plays a crucial role, depicted in Figure 3. In order to make our approach suitable for the GPU, each point x is processed independently by a TFN; the inputs are the visible points S_t and their image flows. Each point x from X_{t-1} is projected into the 2D image of the previous structured point clouds S_{t-1}, and around each projected point, n points are randomly sampled from a Gaussian distribution centered at the current point (we used $n = 49$), thereby introducing a spatial bias toward nearby regions. TFN processes not only the sampled relative positions and flow vectors of nearby visible points, but also the last m flow vectors previously predicted (we used $m = 30$), which are passed into

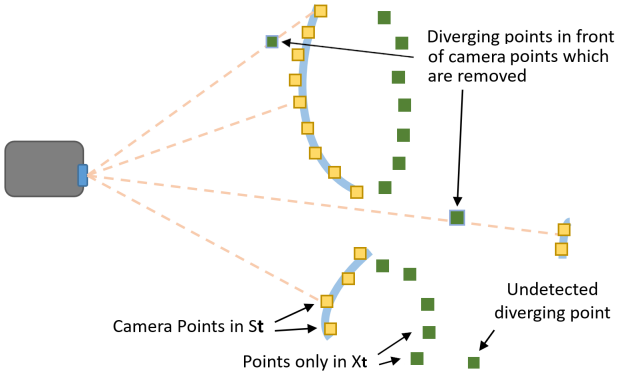

Figure 4: *Removal of diverging points in X_t based on the reliable representation S_t.*

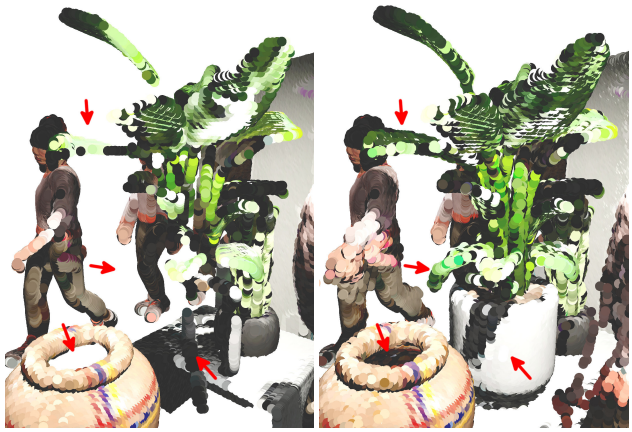

Figure 5: *Plant from Validation Scene B captured by three cameras, shown partly occluded on the left and temporally completed on the right. Static objects are well completed during occlusion.*

a GRU layer. This method ensures that we can track motion even when a point is temporarily occluded and lacks neighboring points for prediction. For every $x \in X_{t-1}$, we generate a flow, updating the point set to $\hat{X}_t$. By merging $\hat{X}_t$ and the current observations S_t, we obtain X_t.

In scenarios involving multiple cameras, each point $x \in X_{t-1}$ is projected into each camera-specific structured point cloud S^c_{t-1}, and TFN runs independently for each camera. Each camera c produces a flow vector f^c_{t-1}, which is averaged using a weighting function $w = 2^{-r \cdot d}$ where r is a constant (we used 20, resulting in the exponential function halving every 0.05 m), and d represents the average distance of the occluded point x to surrounding points. This weighting favors flow vectors from cameras closer to the occluded point, enhancing accuracy in predictions from those views.

Point Management

To ensure that the size of point set X remains manageable, we replace old points that are very close to new points and remove points when the density becomes too high or the points diverge significantly.

To maintain a specific point density in X, we employ a hashset-based voxel grid to record whether a voxel is already occupied (we used a voxel side length of 0.4 cm). Using CUDA, we iterate in parallel over all points x in X, generate a unique hash code for each voxel, and check if this hash code is already recorded in the hashset. If not, the hash code is added; if it is already present, the point x is removed. This ensures that each voxel cell contains at most one point. Additionally, by assuming that S_t generally provides a reliable representation of a visible surface, we can eliminate diverging points: If a point in X_t lies directly in front of a point in S_t from the camera's perspective, it is deemed non-existent as it would have otherwise been detected by the camera (see Figure 4).

3. Experiments

To train our TinyFlowNet (TFN), we require the ground truth flow of occluded points. Since well-known datasets such as the KITTI Scene Flow dataset [MG15] lack occluded points in their ground truth data, we created our own synthetic dataset to effectively train our TFN. Our dataset comprises three different scenes (one training scene and two test scenes), each featuring three virtual RGB-D cameras at a resolution of 640×576 at 30 Hz. In addition to the images from the RGB-D cameras, ground truth points of visible meshes including their corresponding flow vectors are sampled, ensuring that point information of occluded points is also available even if they are not captured by any of the three virtual cameras, so we can train and evaluate the TFN.

In our experiments, we trained the network using the L1 loss function and the Adam optimizer. We evaluated our pipeline on the scenes of our synthetic dataset (e.g., Figure 5) and on real point cloud recordings by multiple Azure Kinects using the CWIPC-SXR dataset [RAJ*21]; see Figure 6. In all datasets, our pipeline was able to temporally complete the scenes at different levels of detail. In our synthetic training scenario, occluded points traveled on average 30.6 cm with an error of 4.5 cm in the Training Scene, 15 cm with an error of 2.9 cm in Validation Scene A, and 34.6 cm with an error of 7.7 cm in Validation Scene B. These correspond to relative errors of 14.8 %, 19.15 %, and 22.33 %, respectively, compared to the traveled ground truth distances after 30 frames. These results indicate that TFN is capable of efficiently learning and generalizing motion patterns using only adjacent visible points. In our Supplemental Material, we compare the usage of PDFlow [JSGJC15] with that of optimal image flow, and also provide a visual ablation study, analyze runtime performance, and give dataset information.

4. Conclusion

We presented a method for temporally completing dynamic point clouds using small neural networks that infer the motion of occluded points from visible point flows. Our approach handles short-term occlusions effectively, without requiring canonical scene reconstructions, and generalizes robustly to unseen scenes. The method supports arbitrary point counts, scales linearly, is architecturally compatible with multi-GPU inference, and allows for control over point cloud density. Our LibTorch implementation

Figure 6: *"Completion results on the Scarf scene from the CWIPC-SXR dataset [RAJ*21], which contains dynamic real-world point clouds captured by seven synchronized Azure Kinects. In our experiments, we used data from only three widely spaced cameras, resulting in large occluded regions. Temporal self-occlusions are mostly completed effectively, although reconstructed areas may still exhibit some visual noise.*

achieves real-time performance for up to 30,000 points, with room for further optimization.

While our approach effectively completes temporal holes in real-world data—which is particularly beneficial for non-visual applications relying on occlusion cues—some visual artifacts, such as noise and minor inconsistencies, still remain in the reconstructed regions for visual use cases. To mitigate these artifacts, future work will explore integrating alternative neural network architectures into our pipeline, leveraging color and connectivity cues to enhance robustness and reduce drift, and developing a native CUDA implementation to further improve performance.

Our datasets and C++ implementation (utilizing LibTorch and CUDA) are available at Github[†].

Acknowledgements

This work was partially supported by BMBF grant 16SV9239.

References

[FMK*22] FISCHER R., MÜHLENBROCK A., KULAPICHITR F., USLAR V. N., WEYHE D., ZACHMANN G.: Evaluation of point cloud streaming and rendering for vr-based telepresence in the or. In *Proc. EuroXR* (2022), pp. 89–110. 1

[GCC*20] GAMELIN G., CHELLALI A., CHEIKH S., RICCA A., DUMAS C., OTMANE S.: Point-cloud avatars to improve spatial communication in immersive collaborative virtual environments. *Pers. Ubiquitous Comput. 25*, 3 (2020), 467–484. 1

[GJS*21] GASQUES D., JOHNSON J. G., SHARKEY T., FENG Y., WANG R., XU Z., ZAVALA E., ZHANG Y., XIE W., ZHANG X., DAVIS K., YIP M., WEIBEL N.: Artemis: A collaborative mixed-reality system for immersive surgical telementoring. In *Proc. ACM CHI* (2021). 1

[IZN*16] INNMANN M., ZOLLHÖFER M., NIESSNER M., THEOBALT C., STAMMINGER M.: Volumedeform: Real-time volumetric non-rigid reconstruction. In *Proc. ECCV* (2016), pp. 362–379. 1

[JSGJC15] JAIMEZ M., SOUIAI M., GONZALEZ-JIMENEZ J., CREMERS D.: A primal-dual framework for real-time dense rgb-d scene flow. In *Proc. ICRA* (2015), pp. 98–104. 2, 3

[MG15] MENZE M., GEIGER A.: Object scene flow for autonomous vehicles. In *Proc. CVPR* (2015). 3

[MHU*25] MÜHLENBROCK A., HUSCHER H., USLAR V. N., CETIN T., WELLER R., WEYHE D., ZACHMANN G.: A novel, autonomous, module-based surgical lighting system. *ACM Trans. Comput. Healthcare* (2025). 1

[NFS15] NEWCOMBE R. A., FOX D., SEITZ S. M.: Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proc. CVPR* (2015). 1

[RAJ*21] REIMAT I., ALEXIOU E., JANSEN J., VIOLA I., SUBRAMANYAM S., CESAR P.: Cwipc-sxr: Point cloud dynamic human dataset for social xr. In *Proc. ACM MMSys* (2021), pp. 300–306. 3, 4

[RYP*21] ROTH D., YU K., PANKRATZ F., GORBACHEV G., KELLER A., LAZAROVICI M., WILHELM D., WEIDERT S., NAVAB N., ECK U.: Real-time mixed reality teleconsultation for intensive care units in pandemic situations. In *Proc. IEEE VRW* (2021), pp. 693–694. 1

[SBI18] SLAVCHEVA M., BAUST M., ILIC S.: Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *Proc. CVPR* (2018). 1

[WCG*24] WANG J., CUI Y., GUO D., LI J., LIU Q., SHEN C.: Pointattn: You only need attention for point cloud completion. *Proc. AAAI Conf. Artif. Intell. 38*, 6 (2024), 5472–5480. 1

[XWL*23] XIANG P., WEN X., LIU Y.-S., CAO Y.-P., WAN P., ZHENG W., HAN Z.: Snowflake point deconvolution for point cloud completion and generation with skip-transformer. *IEEE Trans. Pattern Anal. Mach. Intell. 45*, 5 (2023), 6320–6338. 1

[YGE*21] YU K., GORBACHEV G., ECK U., PANKRATZ F., NAVAB N., ROTH D.: Avatars for teleconsultation: Effects of avatar embodiment techniques on user perception in 3d asymmetric telepresence. *IEEE Trans. Vis. Comput. Graph. 27*, 11 (2021), 4129–4139. 1

[YKH*18] YUAN W., KHOT T., HELD D., MERTZ C., HEBERT M.: Pcn: Point completion network. In *Proc. 3DV* (2018), pp. 728–737. 1

[YZG*21] YU T., ZHENG Z., GUO K., LIU P., DAI Q., LIU Y.: Function4d: Real-time human volumetric capture from very sparse consumer rgbd sensors. In *Proc. CVPR* (2021). 1

[†] https://www.github.com/muehlenb/TemPCC