SUPPLEMENTARY MATERIAL FOR "A UNIFIED FRAMEWORK FOR DYNAMIC POINT CLOUD COMPRESSION"

1. INTRODUCTION

In this supplementary material, we provide more details about our architecture and the training strategy used for training the proposed unified model. In terms of experiments, we include more details about the training and test datasets, as well as more complete results for application to attribute coding.

2. DETAILS OF THE ARCHITECTURE

Below we provide additional details of the architecture omitted in the main text due to space constraints.

2.1. Key difference from prior-arts

We start by highlighting some crucial differences from the existing related approaches. First, unlike most existing Octree Coding approaches that use only already available ancestral information as context for occupancy probability estimation, we perform feature-assisted Octree Coding where a feature coded into the bitstream is used for context modeling. Our feature is constructed from the current level to be encoded, in addition to the already available ancestral information, and is thus comparatively richer in information. Moreover, with our design the dependency between the levels in terms of feature extraction is broken and replaced with dependency between levels in terms of coding: as we are using the *coded* features and extracting ancestral features from already decoded parent level.

Secondly, unlike SparsePCGC [1] which also performs Feature-assisted Octree Coding, our parameters for the CFC module are completely shared between the Octree Coding and Feature Coding for all levels.

Thirdly, unlike ALL existing Feature Coding approaches, our CFC module is hierarchical in nature as it utilizes the lossless/lossy reconstruction and the intermediate quantized feature from the previous level, instead of being completely independent for each level *e.g.*, like in SparsePCGC [1].

2.2. Feature Interpolation Module

The goal of the Feature Interpolation (FI) module is to align the coordinates of the input feature map $\mathbf{F}^i$ with the lossy occupancy reconstruction $\widehat{\mathbf{O}}^{i-1}$ from the previous level, as also shown in Fig. 4 in the main text. This FI module can be skipped in case of lossless reconstruction from the previous level. To achieve this alignment of coordinates, the FI module first utilizes a sparse 3D "target convolution" for which the locations of the output feature map can be specified as the lossy occupancy reconstruction $\widehat{\mathbf{O}}^{i-1}$ from the parent level. Afterwards, the FI module performs feature aggregation via a few sparse 3D convolution layers to update the output feature map of the sparse 3D "target convolution".

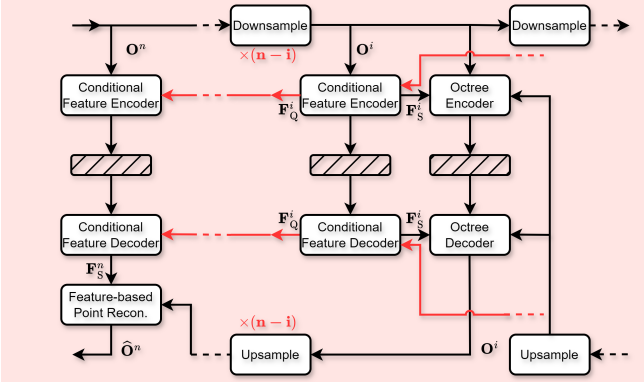

Fig. 1. The overall architecture of the main branch in our proposed method with point-based Feature Coding.

2.3. Point-based Architecture

The architecture design of the Feature Analysis (FA) and Feature Synthesis (FS) modules for Conditional Feature Coding (CFC) is motivated by the point cloud content being coded. For dense surface point clouds representing objects/scenes with uniform distribution of the points on the surface, a voxel-based architecture utilizing sparse 3D convolutions is used. However, for dense point clouds with non-uniform distribution of points and for sparse point clouds like those acquired from LiDAR, point-based architecture is more appropriate as it can have a larger receptive field without increasing the number of parameters to be learned, but at the cost of higher computational complexity. Moreover, with point-based architectures, hierarchical Feature Coding remains an open problem. We defer more details of this discussion to the Supplementary. Octree Coding for all content types is based on voxel-based architecture as lossless Octree Coding is deployed for

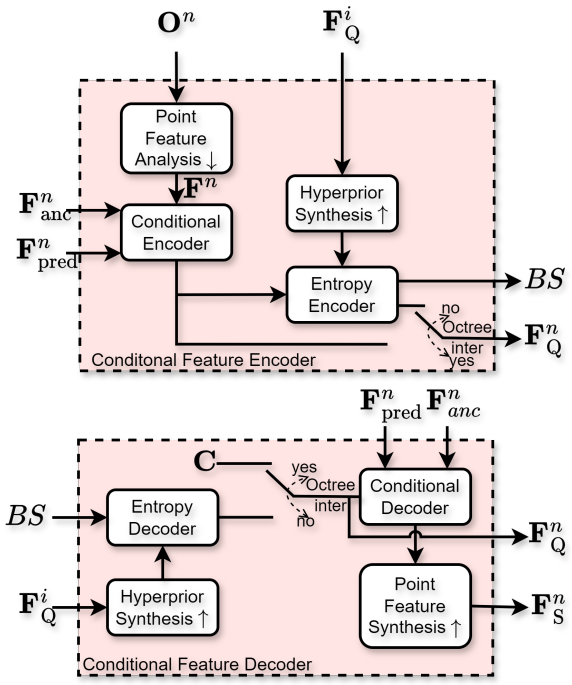

Fig. 2. Proposed conditional feature coding (CFC) module with point-based feature analysis and synthesis modules.

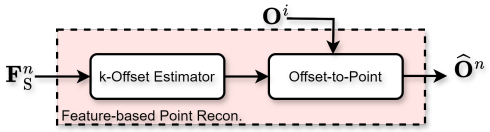

Fig. 3. Proposed Feature-based Point Reconstruction ($\mathrm{FR_P}$) module.

lower levels of the Octree where point distribution is mostly uniform, as observed in existing works [2].

Our proposed UniFHiD with a point-based architecture for Conditional Feature Coding is shown in Fig. 1. One can see that compared to Fig. 1 in the main text, the Octree Coding pipeline stays the same. However, the point-based Feature Coding pipeline is now performed in one-shot instead of hierarchical Feature Coding. The input to the point-based CFC module is directly the input point cloud occupancy $\mathbf{O}^n$, instead of $\mathbf{O}^j$ at an intermediate j-th level. Moreover, the Feature-based Point Reconstruction $\mathrm{FR_P}$ module shown in Fig. 3 also has a different architecture than the earlier Feature-based Reconstruction (FR) module.

The design of our proposed point-based CFC module is shown in Fig. 2, where the architecture is very similar to the convolution-based CFC from Fig. 3 in the main text. The difference is the Point Feature Analysis (PFA) and Point Feature Synthesis (PFS) modules, which are explained herein.

The PFA module is like the well-known PointNet++ [3]

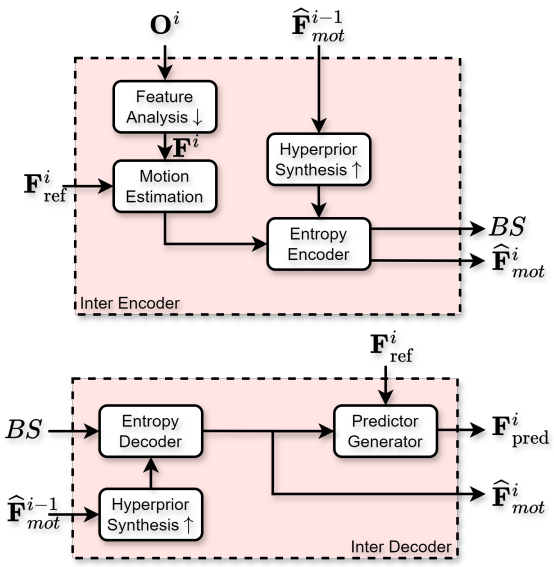

Fig. 4. The architecture of our inter coding branch.

but with a key difference. To avoid the computational complexity incurred by the neighborhood search, in our proposal, the neighborhood is constructed around each ancestor node at the last Octree Coding level i by only considering its children nodes at the leaf level. Since it is obvious which lead nodes belong to which ancestor at the i-th Octree level there is no need for a search. With the neighborhoods available, a PointNet [4] with shared parameters digests each neighborhood to produce a feature for each neighborhood, which along with the coordinates of the ancestor at the i-th Octree level composes the feature map $\mathbf{F}^n$.

On the decoder side, the PFS module takes $\widehat{\mathbf{F}}^n$ as input and uses an MLP module to increase the feature channel dimension from f to kf', where k is the number of offsets to be predicted within each neighborhood and f' is the new feature channel dimension for each offset to be predicted.

Finally, the $\mathrm{FR_P}$ module from Fig. 3 consist of two submodules: k-offset estimator and offset-to-point. The k-offset estimator module, implemented via MLPs, takes the feature map $\mathbf{F}^n_{\mathrm{S}}$ as input and produces k offsets for each feature vector. These k offsets per feature vector are then ingested by the offset-to-point module along with the reconstructed occupancy from the last octree level $\mathbf{O}^i$, to produce the final reconstructed points $\widehat{\mathbf{O}}^n$ by adding the offsets to the associated point locations from the last octree level.

2.4. Dynamic Coding: Inter Architecture

As pointed out in the main text, the inter coding branch shares a lot of similarities in terms of architecture design with the CFC pipeline. Specifically, the ancestral feature map $\mathbf{F}^i_{\mathrm{anc}}$ is replaced with a reference frame feature map $\mathbf{F}^i_{\mathrm{ref}}$ at the same Octree level obtained in the same way as feature map $\mathbf{F}^i$ is

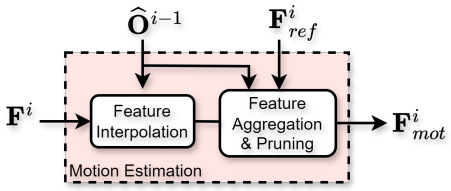

Fig. 5. The architecture of our motion estimation module

obtained using the FA module. The rest of the pipeline for the encoding/decoding of the inter feature remains similar to CFC as shown in Fig. 4. Just as before, having an architecture similar to CFC makes the inter coding branch be hierarchical in nature as well.

The motion estimation (ME) and predictor generator (PG) modules have architectures that are very similar to the conditional encoder (CE) and conditional decoder (CD) modules in the main coding branch, respectively. Since the ME module is on the encoder side it requires FI module for coordinate alignment just like the CE module. Moreover, an additional "Pruning" module is needed in ME and PG modules (in contrast to CE and CD modules) to align the coordinates of the feature aggregation to $\widehat{\mathbf{O}}^{i-1}$.

2.5. Style Control

Existing learning-based feature coding methods such as [5, 2, 1] lack proper rate control mechanisms. To achieve different bitrates during inferences, these methods need to reload a different neural network model optimized for a particular rate-distortion trade-off. However, this approach lacks flexibility: it requires the compression system to maintain a set of trained models, and it is unable to achieve fine-grain/continuous rate control. To overcome this issue, a variable-rate mechanism is needed. Additionally, to have a single unified model for coding in both intra and inter modes, a seamless switching between the intra-only coding and inter-coding is also desired. Both of these goals can be addressed via our proposed novel Style Control (SC) module, as shown in Fig. 6.

This SC module can be augmented to (the output of) each module in the network, taking a feature $\mathbf{F}^i$ as input and transforming its "style" in light of the control parameters: rate-distortion trade-off parameter λ and an inter-coding flag I. The main goal of the SC module is to transform the input features to the given control parameters through learnable modules but performing simple operations like scaling and shifting the mean of features. The SC module further consists of two sub-modules, also depicted in Fig. 6. The Style Encoding (SE) module takes the input control parameters (λ, and I) and embeds them to a higher dimensional space using an embedding, *e.g.*, sinusoidal embedding. Afterwards, the embedded vector is transformed into a style feature $\mathbf{f}_{\text{style}}$ through an MLP module. Next, the style feature $\mathbf{f}_{\text{style}}$ goes to the Adaptive Affine (AA) module which is tasked with updating the input feature $\mathbf{F}^i$ to a new feature $\mathbf{F}^i_{\text{new}}$. The AA module first normalizes the input feature to get a normalized input feature $\mathbf{F}^i_{\text{nm}}$ followed by an Affine module which scales and shifts the mean according to the parameters σ and m (respectively) predicted from the style feature $\mathbf{f}_{\text{style}}$ through another MLP module. With this architecture design, while having only one set of model parameters, the proposed model can achieve a fine-grain rate control by varying λ in a pre-defined range or switching the mode of operation (intra or inter mode).

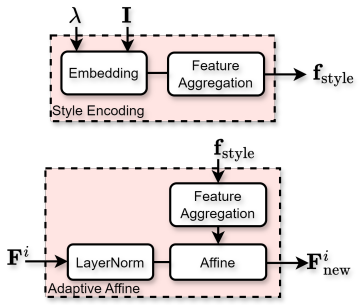

Fig. 6. Sub-modules of the proposed Style Control with Style Encoding (top) and Adaptive Affine (bottom).

2.6. Levelwise Stochastic Training

Since our proposed model is unified for both Octree Coding and Feature Coding, a special mechanism is applied during training. As pointed out in Sec. 2 in the main text, our method breaks the dependency between levels in terms of feature extraction (while maintaining the dependency in terms of coding). A motivation for this design is to enable the training to be performed in a level-wise manner rather than fully end-to-end, so as to reduce computational cost. Additionally, since our model is unified between Octree Coding and Feature Coding in terms of network parameters (majority of the parameters) and can seamlessly switch between inter and intra modes with the proposed Style Control technique, we can randomly switch between four coding configurations during training: intra Octree, intra Feature, inter Octree, and inter Feature. The ratio of our network getting trained in either configuration can be controlled by user-specified parameters.

To avoid our network getting stuck in local minimas that lead to poor inference performance, we employ a two-stage training procedure with a special schedule for rate-distortion trade-off parameter λ. In the first stage, we use a fixed low value λ which incurs a high bitrate during training but can achieve good reconstruction performance, *i.e.*, low distortion. This first stage lasts a few epochs, after which we use a normal schedule for λ where we randomly pick the value for λ within a specified range.

For the training of Octree coding the training loss consists of a binary cross-entropy loss (between the predicted prob-

Table 1. Training datasets used.

Class	Train (Sequence) Name	Fr.	Prc.
DS	Head_00039_vox12	1	12
	Frog_00067_vox12	1	12
	Egyptian_mask_vox12	1	12
	ULB_Unicorn_vox13	1	13
DS	RWTT Train Set	406	Float
DD	Queen	250	10
	8i VFB – Loot	300	10
	8i VFB – Red_and_Black	300	10
	8i VFB – Soldier	300	10
	8i VFB – Long_dress	300	10
SD	KITTI (00-10)	23201	18

Table 2. Test datasets used.

Class	Test (Sequence) Name	Fr.	Prc.
DS	Facade_00009_vox12	1	12
	House_without_roof_00057_vox12	1	12
	Arco_Valentino_Dense_vox12	1	12
	Statue_Klimt_vox12	1	12
	Shiva_00035_vox12	1	12
DS	RWTT_vishnu_156_vox10	1	10
	RWTT_foxstatue_211_vox10	1	10
	RWTT_tomb_059_vox10	1	10
DD	Exercise_vox10	300	10
	Model_vox10	300	10
	Dancer_vox11	300	11
	Basketball_player_vox11	300	11
	Thaidancer_viewdep_vox12	300	12
SD	KITTI (11-21)	3300	18

abilities and the ground truth occupancies) used for learning the occupancy probabilities and the rate loss of the features which is the bitrate estimated by the entropy bottleneck layer. The Feature Coding is trained using a rate distortion loss $L = L_D + \lambda L_R$, where the rate is again the bitrate of the feature estimated by the entropy bottleneck layer, and the distortion is computed using the binary cross-entropy loss as the Feature Coding produces a lossy utilizing the learned occupancy probabilities.

3. DATASET DETAILS

The datasets for our experimental evaluation consist of three main categories: Dense Static, Dense Dynamic and Sparse Dynamic, and follows the experimental guidelines outlined in the MPEG AI-PCC CfP [6]. The training set in the Dense Static category consists of a subset of the static point clouds from MPEG G-PCC CTC and some subset of static 3D textured models in the Real World Textured Things (RWTT) dataset [7] with permittive licenses. The corresponding test sets consist of another subset of the static point clouds from MPEG G-PCC CTC and three static 3D textured models in the RWTT dataset [6].

In the Dense Dynamic category, the training set is composed of 8iVFB dynamic point cloud sequences and the Queen sequence, while the test set consists of dynamic point cloud sequences from the V-PCC CTC. Finally, the training and testing data for Sparse Dynamic are the training and testing splits from the well-known spinning-LiDAR KITTI [8] dataset. Some additional details about each dataset category are provided in Supplementary. We provide some more details in Tab. 1 and Tab. 2 about the training and test datasets, respectively. These details include information like Class, number of frames (Fr.), and geometry precision (Prc.). For RWTT train set, 406 meshes among 568 were selected before processing them to point cloud training set. These mesh vertices are originally available in floating-point values, hence, quantization was performed on the vertices before point-sampling on top of the meshes. For KITTI train set we use sequences 00 to 10 with all their frames, while for the KITTI test set we use sequences 11 to 21 with the first 300 frames from each test sequence.

4. APPLICATION TO ATTRIBUTE CODING: FULL RESULTS

As mentioned in the main text, we paired our proposed geometry coding proposal with a traditional attribute coding named RAHT (Region Adaptive Hierarchical Transform) [10] to make a full Hybrid coding framework for point cloud compression (AI-based geometry coding + non-AI attribute coding). The full results of this hybrid framework are compared with existing approaches, as shown in Tab. 3. The traditional approaches for comparison are GeSTM-Octree and GeSTM-TriSoup, whereas the learning-based approaches are JPEG AI PCC, YOGA, and Unicorn. Our proposed approach is the only hybrid approach with learning-based geometry coding and traditional attribute coding. Additionally, in contrast to all other methods, JPEG AI PCC is a projection-based approach where it first projects the 3D point cloud geometry and attributes to 2D images followed by JPEG AI to compress the said 2D images. It should also be noted that the attribute compression part of GeSTM is also based on RAHT like our hybrid approach.

We can observe considerable gains across the board for all types of point clouds with our proposed Hybrid UniFHiD. Compared to GeSTM, this showcases the importance of better geometry coding that our learning-based method can provide. On the other hand, we can see considerable gains over all fully learning-based approaches which highlights the current performance gap between traditional and learning-based attribute compression. The learning-based attribute compression methods still have room to improve to catch up to and outperform the traditional attribute compression approaches for 3D point clouds.

Table 3. Comparison of our Hybrid UniFHiD method with other non-learning and learning based solutions, based on gains over G-PCC (for KITTI) and V-PCC (for Dense).

Dataset	Dense Static	Dense Dynamic	KITTI
Metric	Y,Cb,Cr	Y,Cb,Cr	R
JPEG AI PCC	+31.49%,+36.96%,+46.68%	—	—
GeSTM-Octree [9]	+63.79%,+59.21%,+46.55%	+80.78%,+15.89%,+11.41%	-0.90%
GeSTM-TriSoup [9]	-16.58%,-31.81%,-48.66%	-4.87%,-52.89%,-56.53%	—
YOGA	+15.78%,-36.24%,-63.03%	—	—
Unicorn	-25.30%,-40.23%,-40.80%	+6.96%,-19.71%,-25.27%	-59.22%
Hybrid UniFHiD	-25.7%,-36.0%,-48.8%	-22.8%,-54.6%,-56.5%	-77.5%

5. REFERENCES

[1] Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma, "Sparse tensor-based multiscale representation for point cloud geometry compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 7, pp. 9055–9071, 2022.

[2] Jiahao Pang, Kevin Bui, and Dong Tian, "Pivotnet: Heterogeneous point-voxel-tree-based framework for point cloud compression," in *2024 International Conference on 3D Vision (3DV)*. IEEE, 2024, pp. 1270–1279.

[3] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[4] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[5] Jiahao Pang, Muhammad Asad Lodhi, and Dong Tian, "Grasp-net: Geometric residual analysis and synthesis for point cloud compression," in *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, 2022, pp. 11–19.

[6] MPEG Technical Requirements (WG 02), "Updated call for proposals for ai-based point cloud coding," July, 2024, Accessed on Nov 14, 2024.

[7] Andrea Maggiordomo, Federico Ponchio, Paolo Cignoni, and Marco Tarini, "Real-world textured things: A repository of textured models generated with modern photo-reconstruction tools," *Computer Aided Geometric Design*, vol. 83, pp. 101943, 2020.

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[9] MPEG Coding of 3D Graphics and Haptics (WG 07), "An mpeg (ge)ometry based point cloud compression test model for (s)olid content," February, 2023, Accessed on Nov 14, 2024.

[10] Gustavo Sandri, Franck Thudor, Maja Krivokuća, and Bertrand Chupeau, "A motion-compensated interframe attribute coding scheme for dynamic dense point clouds," in *2023 IEEE 25th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2023, pp. 1–6.