Joint Learning of Point Clouds and Motion Vectors for Volumetric Video

Cheng-Tse Lee[1], Yuan-Chun Sun[1], Yuang Shi[2], Mufeng Zhu[3], Wei Tsang Ooi[2], Yao Liu[3], Chun-Ying Huang[4], and Cheng-Hsin Hsu[1]
[1]National Tsing Hua University, Taiwan
[2]National University of Singapore, Singapore
[3]Rutgers University, NJ, USA
[4]National Yang Ming Chiao Tung University, Taiwan

Abstract

Dynamic point clouds are simple and versatile representations for volumetric video. A challenge in processing and analyzing dynamic point clouds is the lack of explicitly defined structure and motion information across frames. This paper addresses a novel motion prediction problem that jointly learns point clouds and motion vectors by moving the points from a key frame to construct the following non-key frames while approximating the rendered 2D views of the input non-key frames. Our motion prediction algorithm is built upon augmented dynamic 3D Gaussian Splatting (3DGS) training algorithms with a lightweight point cloud converter and optimal parameter selector. Extensive experiments show that our resulting motion vectors lead to a PSNR (Peak Signal-to-Noise Ratio) gain of up to 8.71 dB and SSIM (Structural Similarity Index Measure) increase of up to 0.27, compared to the current practice. The computed motion vectors can be leveraged in multiple downstream applications, such as error concealment, temporal super-resolution, and source coding. Using the learned motion vectors for error concealment, we observe quality improvement by, at most, 3.93 dB in PSNR and 0.28 in SSIM, compared to a state-of-the-art end-to-end neural network.

CCS Concepts

• **Computing methodologies** → **Graphics systems and interfaces**; **Supervised learning**; • **Information systems** → **Multimedia information systems**.

Keywords

Dynamic point clouds, motion prediction, reconstruction

ACM Reference Format:

Cheng-Tse Lee, Yuan-Chun Sun, Yuang Shi, Mufeng Zhu, Wei Tsang Ooi, Yao Liu, Chun-Ying Huang, and Cheng-Hsin Hsu. 2025. Joint Learning of Point Clouds and Motion Vectors for Volumetric Video. In *The 17th International Workshop on IMmersive Mixed and Virtual Environment Systems (MMVE '25), March 31-April 4 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712677.3720458

MMVE '25, March 31-April 4 2025, Stellenbosch, South Africa

ACM ISBN 979-8-4007-1468-9/2025/03
https://doi.org/10.1145/3712677.3720458

1 Introduction

Dynamic point clouds have emerged as a versatile and efficient representation of volumetric video. Unlike other 3D representations such as meshes, 3D point clouds can flexibly capture complex, non-manifold shapes and represent a variety of attributes. Furthermore, point clouds can be produced directly from sensors such as Light Detection and Ranging (LiDAR) and RGB-D cameras. Without topological information, point clouds are more compute- and memory-efficient. MPEG has recently standardized two point cloud compression formats, allowing compatible volumetric video processing, streaming, and visualization, across applications and devices. These benefits of point clouds have driven its use in many applications, including: (i) eXtended Reality (XR) to enhance the immersive experience through real-time 3D capturing and rendering [7, 19]; (ii) networked collaboration systems to share 3D scenes [10, 21]; (iii) cultural heritage preservation to record detailed structures of historical sites [36]; and (iv) self-driving cars to model environments for object detection and path planning [18].

An underlying property that leads to the simplicity and versatility of dynamic point clouds is that each frame is just an unstructured list of 3D points with attributes. Such lack of structure, unfortunately, is a double-edged sword, especially when multiple frames of a dynamic point cloud sequence captured from a natural scene need to be jointly analyzed and processed. A typical processing pipeline produces each point cloud frame independently, optimized for accurate 3D representation of the objects being captured. As a result, there is no explicit relationship among the points across different frames in a sequence. For example, each frame from the well-known 8i Voxelized Full Bodies dataset [4] is just a list of points without any correspondence between two points with the same indices in adjacent frames. Such lack of structures complicates the analysis and processing of dynamic point cloud sequences along the temporal domain [31]. An important missing information is the motion vectors–the motion of a point between two frames, which are critical for many downstream applications, such as enabling interpolation for error concealment and temporal super-resolution and leveraging temporal redundancy for source coding.

To estimate the motion vectors between two unstructured point cloud frames, prior studies resorted to point-matching heuristics. For example, Hung et al. [13] proposed search algorithms to identify similar points between two frames to infer their motion vectors. Some other works chose to employ neural networks for point matching [14, 22, 23, 32, 35, 39, 40], which assumed a fixed number of

The work was partially supported by the NSTC grants of Taiwan (#112-2221-E-007-090-MY3 and #113-2221-E-A49-186-MY3).

points across the frames. Such a strong assumption, unfortunately, does not hold in dynamic point cloud sequences of natural scenes. These studies focus on estimating motion vectors between two *fixed* point cloud frames. This constraint severely limits the accuracy of motion vectors, as it is too restrictive–after all, the goal of a point cloud is to reproduce the visual appearance of a 3D scene when rendered, and many possible point clouds can lead to the same visual appearance. In this paper, we demonstrate that we can obtain more accurate motion vectors if we allow modification to the points in one of the frames, while retaining the visual appearance.

Fig. 1 illustrates our key idea. Given a sequence of point cloud frames, we designate the first frame as the *key frame* and the rest as non-key, *predicted frames*. For each predicted frame, we re-generate a point cloud so that: (i) the rendered views of this point cloud is as close to those of the corresponding input frame, and (ii) the motion vectors between this and its previous frame are explicitly defined. We refer to this problem as the *motion prediction* problem, and propose an algorithm to solve it.

Our motion prediction algorithm consists of the following steps. First, we render an input non-key frame (in dynamic point clouds) into multiple representative 2D views with different camera parameters. Next, inspired by neural networks for training dynamic 3DGS (3D Gaussian Splatting) representations [3], we construct a predicted non-key frame (in 3D Gaussian splats) with the best possible quality of their rendered 2D views. Last, we propose a lightweight procedure to convert this predicted non-key frame from 3D Gaussian splats to a point cloud, with minimal quality degradation of the rendered views while providing corresponding motion vectors.

This paper makes the following contributions:

- We develop a novel motion prediction algorithm for point matching in dynamic point cloud sequences. Instead of indirect and vague mappings, we derive the motion vectors from individual points of a non-key predicted frame to those of its key frame.
- We demonstrate the potential use of the derived motion vectors in a sample downstream application: error concealment, in which some point cloud frames are lost, late, or corrupted and need to be concealed at the receiver side using the successfully decoded frames.

We conducted extensive experiments to evaluate our motion prediction algorithm and the sample downstream application. Compared to the current practice, the motion vectors produced by our motion prediction algorithm lead to: (i) higher 2D video quality by up to 8.71 dB in PSNR (Peak Signal-to-Noise Ratio) and 0.27 in SSIM (Structural Similarity Index Measure) [11]; (ii) higher 3D point cloud quality with up to 61.31% reduction in average geometry distortion and 55.49% increase in average luminance PSNR; and (iii) reasonable running time. For error concealment, compared to the state-of-the-art heuristics [13] and end-to-end neural network [2], our motion vectors result in better 2D video quality: boosts by up to 1.29 dB and 3.93 dB in PSNR as well as 0.07 and 0.28 in SSIM are observed.

2 Related Work

Heuristic-based point matching. Several prior studies [13, 17, 26, 28, 29] proposed some heuristics to match points among different frames of a dynamic point cloud sequence. For instance, Li et al. [17] developed 3D-to-2D projection methods for improving the quality of 2D motion vectors in 2D video-based point cloud coding. Shi et al. [28] presented a source coder for LiDAR point clouds. For 3D motion vectors, Hung et al. [13] proposed three point matching heuristics: (i) point-to-point matching based on position and color similarity, (ii) triangular matching between a point and multiple neighboring points, and (iii) cube matching among fixed size cubes (instead of points). Different from our work, these prior works derived some point mapping, which are typically not good approximations of actual motion vectors.

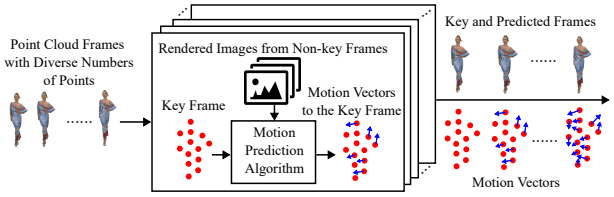

Figure 1: Overview of our proposed method for joint learning of point clouds and motion vectors.

Neural-network-based point matching. Neural networks were also used to match points across different frames [8, 14, 22, 23, 32, 35, 39, 40] for frame interpolation or extrapolation. Their core concept involved downsampling dynamic point clouds into a fixed, predefined number of points, and then using a neural network to compute the matching among these points. For example, Viola et al. [32] proposed a two-stage neural network that takes 2,048 points as input and minimizes the L_2 error between predicted and actual point positions. Similarly, Zeng et al. [39] proposed to match 1,024 points with a loss function based on Earth Mover's distance [5]. These neural networks came with some, if not all, of the following limitations: (i) fixed, and often too few, number of points per frame, (ii) high computational complexity, and (iii) no color supports, and are not well-suited to real-time applications with natural scenes.

End-to-end neural networks. Instead of explicit point matching, some existing neural networks [1, 2, 6, 38] solved downstream applications in an end-to-end fashion. For instance, Ye et al. [38] presented a two-stage neural network to model the relationships among features computed from points for temporal super-resklution. Akhtar et al. [2] transformed point cloud frames into a multi-scale feature space and searched for spatial and temporal patterns for temporal super-resolution. Their method was also generalized to source coding [1], which employed matching techniques in the feature space to reduce temporal redundancy across frames. Fan et al. [6] applied multi-scale feature analysis to identify matching among different frames to improve coding efficiency. These studies did not provide point matching or motion vectors, limiting their use in alternative downstream applications.

3 Motion Prediction Problem

3.1 Formulation

We first develop the formulation of our motion prediction problem. Each sequence of dynamic point clouds is divided into recurring Groups of Frames (GoFs), where a GoF consists of F input frames. We refer to the first frame, F_1, as the key frame and the following $N-1$ frames, $F_2, F_3, \cdots, F_N$, as the non-key frames. We denote

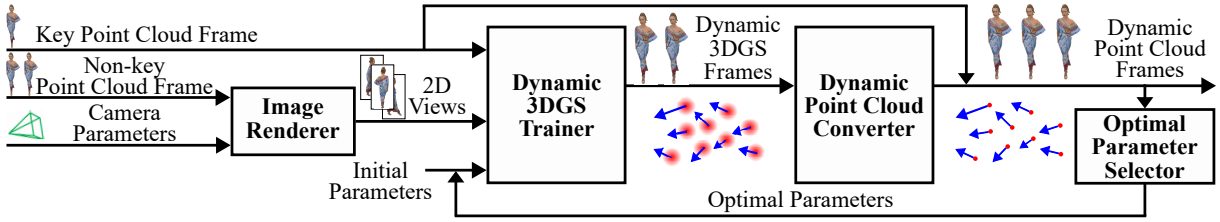

Figure 2: Our proposed motion prediction algorithm.

the number of points of F_n ($n \in \{1, 2, ..., N\}$) as M_n. In general, $M_{n_1} \neq M_{n_2}; \forall n_1, n_2 \in \{1, 2, \cdots, N\}$ and $n_1 \neq n_2$. Each point m in F_n ($n \in \{1, 2, \cdots, N\}$ and $m \in \{1, 2, \cdots, M_n\}$) is composed of a set of attributes, including the positions $P_{n,m}$, which are the $\{x, y, z\}$ coordinates and the colors $C_{n,m}$, which are the $\{r, g, b\}$ components.

Our motion prediction problem creates $N-1$ predicted non-key frames as output, where each of these frames has M_1 points, i.e., the same as the key frame. We use F'_n ($n \in \{1, 2, \cdots, N\}$) to denote the predicted frames, where $F'_1 = F_1$. Upon getting the predicted non-key frames, their "motion vectors" are defined with a generalized meaning of difference in various attributes, not limited to positions (x, y, z coordinates). In particular, we use $P'_{n,m}$ and $C'_{n,m}$ to denote the positions and colors of point m in predicted frame F'_n ($n \in \{2, 3, \cdots, N\}$ and $m \in \{1, 2, \cdots, M_1\}$). We then write motion vectors of positions and colors as $\Delta P_{n,m} = P'_{n,m} - P_{n,m}$ and $\Delta C_{n,m} = C'_{n,m} - C_{n,m}$, respectively. We collectively write all motion vectors of a frame F'_n as ΔP_n and ΔC_n for brevity.

The objective of our problem is to maximize the rendered view quality of predicted non-key frames F'_n ($n \in \{2, 3, \cdots, N\}$). Let $\mathbf{V}$ be a set of representative virtual camera parameters for rendering the point cloud frames. Each $V \in \mathbf{V}$ contains virtual camera positions, orientation, and intrinsic/extrinsic parameters. We use $R(F, V)$ to denote the rendered view of frame F, where $F \in \{F_1, F_2, \cdots, F_N\} \cup \{F'_1, F'_2, \cdots, F'_N\}$ and $V \in \mathbf{V}$. Last, we define $Q(R(F_n, V), R(F'_n, V))$ to be the visual quality of the predicted frame F'_n ($n \in \{2, 3, \cdots, N\}$), compared to the input frame F_n. The quality function could be based on any objective quality metric, such as PSNR and SSIM.

With all the notations developed thus far, our motion prediction problem can be written as:

$$\underset{\{\Delta P_n, \Delta C_n | \forall n=2,3,\cdots,N\}}{\textbf{argmax}} \quad \sum_{n=2}^{N} \sum_{V \in \mathbf{V}} Q(R(F_n, V), R(F'_n, V)). \tag{1}$$

This motion prediction problem takes $F_n, \forall n \in \{1, 2, \cdots, N\}$, as input and produces $F'_n, \forall n \in \{2, 3, \cdots, N\}$. The camera parameters $\mathbf{V}$, rendering function $R(\cdot)$, and quality function $Q(\cdot)$ are user-specified system parameters.

3.2 Dynamic 3DGS Training Algorithms

Constructing predicted non-key point cloud frames to maximize the objective function in Eq. (1) is no easy task. Therefore, we have decided to leverage recent advances in dynamic 3DGS training algorithms to develop our motion prediction algorithm. In addition to the position and color attributes of point clouds, Gaussian splats come with two more attribute categories: (i) transformation, including scale and rotation, to control their influence ranges and orientations, respectively; and (ii) opacity, to determine their contributions to rendered 2D views. Kerbl et al. [15] proposed the first static 3DGS training algorithm to construct a single 3DGS frame using: (i) a set of input 2D views, (ii) a set of corresponding camera parameters, and (iii) an initial (random or generated from SfM, structure from motion) point cloud. Their training algorithm [15] derives gradients to iteratively update all four categories of attributes to minimize the error between the input (ground truth) and rendered 2D views. Here, the gradients of position attributes can be thought of as moving points in the 3D space, while the gradients of color, transformation, and opacity attributes can be thought of as changing their values.

A naive way to apply this static 3DGS training algorithm on a dynamic sequence is to perform frame-by-frame training. Doing so, however, leads to flickering artifacts [27] along the time domain, which can be attributed to the randomness involved in the training algorithm and initial point clouds. Several dynamic 3DGS training algorithms have been proposed in the literature to tackle these artifacts [3]. Like our considered problem, these dynamic 3DGS training algorithms divide each sequence into several recurring GoFs, where each GoF consists of a key frame, followed by multiple non-key frames. They move Gaussian splats of a key frame to construct the following non-key frames under some local rigid constraints instead of independently building individual non-key frames from scratch. By doing so, they no longer suffer from flickering artifacts and converge faster. The resulting outputs are multiple 3DGS frames with the same number of Gaussian splats. Comparing each non-key frame F'_n ($n \in \{1, 2, \cdots, N\}$) with its key frame $F'_1 = F_1$, we get ΔP_n (position) and ΔC_n (color)[1], as well as: (i) ΔT_n for transformation that consists of scale a and orientation d and (ii) ΔO_n for opacity o.

3.3 Overview of Our Algorithm

Fig. 2 gives an overview of our proposed motion prediction algorithm, which consists of four components: (i) image renderer, (ii) dynamic 3DGS trainer, (iii) dynamic point cloud converter, and (iv) optimal parameter selector. Note that our motion prediction algorithm does not modify the key frame, which is directly taken from the dataset and remains unchanged throughout the process. As for non-key point cloud frames, the image renderer renders the input 3D frames into 2D views using representative camera parameters. These 2D views are then fed into the dynamic 3DGS trainer, serving as the ground truth to construct the non-key 3DGS

[1]We note that while 3DGS supports view-dependent colors through Spherical Harmonic (SH) coefficients, they are not widely used in point clouds. Hence, we only consider view-independent color in this work.

frames. In particular, the dynamic 3DGS trainer takes the key point cloud frame as input and generalizes it into the 3DGS key frame by setting all Gaussian splats' scale, orientation, and opacity to be pre-determined fixed values: A_0, D_0, and O_0, since they do not exist in point clouds.

The resulting key 3DGS frame is then passed into a modified dynamic 3DGS training algorithm, which adjusts the Gaussian splats of the given key 3DGS frame to construct non-key 3DGS frames. We note that, unlike existing dynamic 3DGS training algorithms, such as D3DGS [25], our dynamic 3DGS trainer does not train the key 3DGS frame. Similar to these algorithms, we adjust the 3DGS attributes of non-key frames using the differentiable properties of Gaussian rasterization to ensure their rendered views are as close to the input 2D views from the image renderer as possible. The given key and each constructed non-key 3DGS frame n ($n \in \{2, 3, \cdots, N\}$) define motion vectors ΔP_n, ΔC_n, ΔT_n, and ΔO_n.

The dynamic point cloud converter then turns the non-key 3DGS frames into point cloud frames by stripping off transformation and opacity attributes from individual Gaussian splats. Such a conversion could degrade the quality of the rendered views from point clouds compared to 3DGS frames. To mitigate this issue, we introduce an optimal parameter selector to choose the best parameters, such as the fixed scale A_0 (of 3DGS) and rendered point size s (of point cloud), to minimize the quality degradation. More specifically, the optimal parameter selector chooses the parameters through real experiments to maximize the visual quality (defined in Eq. (1)) of the dynamic point cloud frames. Last, our motion prediction algorithm returns dynamic point cloud frames along with corresponding motion vectors derived from the optimal parameters.

3.4 Design Decisions

Dynamic 3DGS trainer. A couple of design decisions are made. First, we need to determine the fixed 3DGS transformation and opacity attributes, which will be stripped when Gaussian splats are converted into points. For orientation D_0, we set it to get spherical rather than elliptical Gaussian splats, also for better approximating points. Similarly, for opacity O_0, we maximize its value because point clouds are typically rendered without transparency. As for scale A_0, we choose a small value for all Gaussian splats to better match the nature points. Different from D_0 and O_0, the best A_0 value is not obvious, and we empirically derive it later. Second, we fix these attributes throughout the training process, i.e., we changed the loss function by removing the terms related to transformation and opacity attributes. By doing so, our motion prediction algorithm focuses on the spatial and temporal structures of each dynamic scene and enjoys a reduced computational complexity.

Optimal parameter selector. Two core parameters need to be optimally selected: (i) the fixed scale A_0 of Gaussian splats (during training) and (ii) the point size s of point clouds (during rendering). To cover a wider range of A_0, we use a scale parameter $l = ln(A_0)$ as the control knob, following the original 3DGS paper [15]. Then, the task we have in hand is to determine the best l^* and s^* for the optimal visual quality of the renderer 2D views from predicted point cloud frames. Because the Gaussian splats are trained with the representative camera parameters $\mathbf{V}$, here, we use a different set of parameters $\mathbf{V'}$ to evaluate quality fairly.

Our pilot tests indicated that, compared to s, l incurs higher impacts on the visual quality of rendered views. Hence, our proposed algorithm selects l^* first and then determines s^*. The algorithm comes with a few parameters: (i) initial point size S_0, (ii) initial scale parameter L_0, (iii) point size step ΔS, and (iv) scale parameter step ΔL. We next give the sketch of our algorithm:

(1) We start from point size $s = S_0$, and execute the dynamic 3DGS trainer three times with $l \in \{L_0 - \Delta L, L_0, L_0 + \Delta L\}$. We then compute the visual quality of their rendered views using camera parameters $\mathbf{V'}$.
(2) If the visual quality of $l = L_0$ is not higher than both $l = L_0 \pm \Delta L_0$, we denote the $l = L_0 \pm \Delta L$ with higher visual quality as l'. We then run the dynamic 3DGS trainer again with another scale parameter: $l' + \Delta L$ if $l' = L_0 + \Delta L$; or $l' - \Delta L$ if $l' = L_0 - \Delta L$, to zoom into an interval with a better l. By repeating this exploration step a few times, eventually, we would reach a scale parameter L_b that leads to higher visual quality than $L_b \pm \Delta L$. Next, we let L_u be the scale parameter l that leads to higher visual quality between $L_b \pm \Delta L$. At this point, we know l^* falls between L_b and L_u.
(3) We next perform a binary search for l^* that maximizes the visual quality between L_b and L_u, with a step size of $\Delta L / 2^{k-1}$, where $k = 1, 2, \cdots, K$ is the number of iterations. After K iteration, we get l^*.
(4) With the Gaussian splats trained with l^*, we progressively increase the point size from S_0 to $S_0 + \Delta S$, $S_0 + 2\Delta S$, and so on, and render the views at each step. The process continues until no further improvement in visual quality is observed, at which point we determine the optimal point size, s^*.

If not otherwise specified, we empirically set $K = 3$, $L_0 = -10$, $\Delta L = 1$, and $S_0 = \Delta S = 1$. We note that our proposed optimal parameter selector terminates after a handful of executions of the dynamic 3DGS trainer (to search for l^* in steps 1–3) and 3–4 rendered views (to search for S^* in step 4), which are both manageable.

4 Experiments

4.1 Setup

Implementations. We have implemented our proposed algorithm in C++ and Python. Specifically, we use Open3D [41] in the image renderer and optimal parameter selector to generate 2D views from point cloud frames. The dynamic point cloud converter also employs Open3D to process (enriched) point clouds. To implement the dynamic 3DGS trainer, we leverage the differential Gaussian rasterization [15] with a modified loss function based on D3DGS [25] which is the first dynamic 3DGS training algorithm. We have also implemented an error concealment algorithm as a sample downstream application, along with evaluation scripts.

Motion vectors. As there exists no prior work, we implement a current, *traditional* practice by: (i) running D3DGS [25] for dynamic 3DGS frames and (ii) stripping off unnecessary attributes for point cloud frames for comparison.

Four dynamic point cloud sequences: RedandBlack (RNB), Loot (LOO), Soldier (SOL), and Longdress (LON), with different complexity levels from 8i Voxelized Full Bodies dataset [4] are considered for evaluation. We train each frame for 2,000 iterations and keep other settings the same as traditional D3DGS. We randomly take

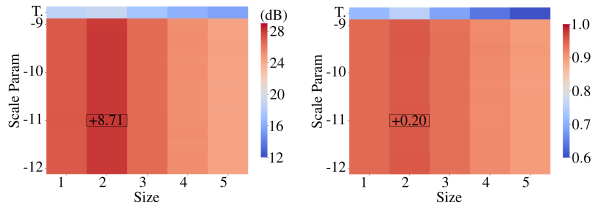

Figure 3: Sample overall quality with different parameters: (a) PSNR and (b) SSIM from Loot.

Table 1: Optimal Parameters and Per-Frame Overhead

Seq.	l^*	s^*	#T	#R	T. Time	R. Time
LOO	-11.00	2	6 (46%)	10 (15%)	97.35 s	5.34 s
SOL	-9.5	2	5 (38%)	9 (14%)	119.45 s	6.09 s
RNB	-11.00	2	6 (46%)	10 (15%)	96.24 s	5.32 s
LON	-10.25	2	5 (38%)	9 (14%)	97.56 s	5.34 s

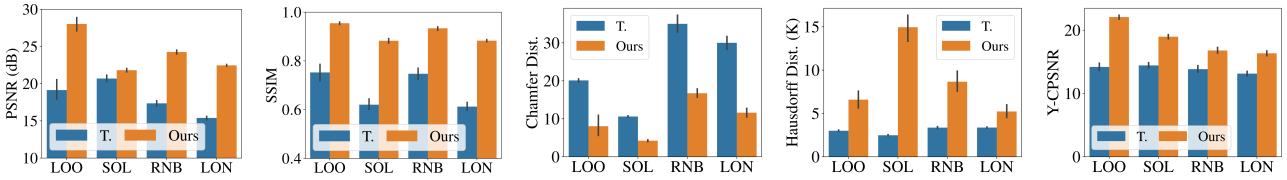

Figure 4: Per-sequence motion vector quality in: (a) PSNR, (b) SSIM, (c) CD, (d) HD, and (e) Y-PSNR.

93 frames of each sequence for experiments with a GoF size 31. We construct the representative camera parameters $\mathbf{V}$ by first horizontally dividing the equator at a step of 18° and then repeating this four times with ±30° and ±60° in latitude, leading to $|\mathbf{V}| = 100$. For $\mathbf{V'}$, we choose 32 random camera parameters that are not in $\mathbf{V}$. All cameras are placed at a distance so that the avatars occupy roughly 75% of the rendered image height.

We consider the following performance metrics:

- **2D video quality**: (i) PSNR of foreground objects to avoid bias due to identical background, and (ii) SSIM for human perceived quality.
- **3D point cloud quality** [33]: (i) Chamfer Distance (CD) representing the overall point-wise distance, (ii) Hausdorff Distance (HD) representing the maximal distance of all nearest point pairs, and (iii) Y-PSNR representing the color difference between the nearest point pairs.
- **Running time:** We report per-frame running time on a PC with an AMD 2.80 GHz CPU and NVIDIA RTX-3090 GPU.

We report the average results with 95% confidence intervals whenever possible. Notice that since our algorithm does not change key frames, they are excluded when calculating 2D and 3D quality.

Error concealment. For comparison, we consider two representative heuristics [13]: (i) point-to-point matching (P2P) and (ii) cube matching (Cube) with their default parameters. Although not initially proposed for error concealment, we also adopt a neural network, IDEA-Net (IDEA) [39], for comparison. As IDEA takes exactly 1,024 points as input, we have to divide each frame into multiple cubes with 1,024 points and send each cube through IDEA. We consider three frame loss patterns: 1, 2, and 4 consecutive and recurring frame losses. We report 2D video quality of concealed frames with per-frame running time from a PC with an Intel 3.50 GHz CPU and NVIDIA RTX-3080 Ti GPU.

4.2 Results

Optimality of selected parameters. We first check the selected scale parameter l^* and point size s^* chosen based on PSNR. Fig. 3 gives sample heatmaps of average PSNR and SSIM from Loot, where the top row gives the PSNR and SSIM from traditional. The numbers represent the quality improvements with the selected parameters: $l^* = 11$ and $s^* = 2$: significant boosts of +8.71 dB in PSNR and 0.20 in SSIM compared to traditional. Without our optimal parameter selector, an exhaustive search in this figure requires 13 3DGS trainings (check l) and 65 image renderings (check s). In contrast, our optimal selection algorithm achieves almost optimal quality (with small gaps of 0.02 dB in PSNR and 0.01 in SSIM), yet only takes 6 trainings (46.15% of the exhaustive search) and 10 renderings (15.38%).

Table 1 shows the optimal parameters and per-frame overhead from all sequences. In addition to the training and rendering numbers, per-frame running time is also provided. This table confirms that our optimal parameters can largely reduce the training and rendering numbers. Moreover, the per-frame training time is < 200 seconds, and the rendering time is < 7 seconds, which are both reasonable for prerecorded dynamic point clouds. Given the near optimality of our selected l^* and s^*, we only report their results in the following.

Our algorithm outperforms the current practice. Fig. 4 reports our derived motion vectors' 2D video and 3D point cloud quality. We observe that our motion prediction algorithm consistently outperforms traditional, in terms of 2D video quality: up to 8.71 dB in PSNR and 0.27 in SSIM are observed. Such quality boosts are as designed because our dynamic 3DGS trainer employs a loss function based on 2D image quality. For 3D point cloud quality, our motion prediction algorithm also performs well in CD and Y-PSNR: up to 61.31% reduction in CD and 7.88% rise in Y-PSNR are observed.

However, Fig. 4(d) shows that our motion prediction algorithm leads to higher HD, compared to traditional. HD is a metric emphasizing the worst-case point pairs. Indeed, a closer inspection

(a) (b)

Figure 5: A sample frame from RedandBlack: (a) a 2D input view and (b) a zoom-in rendered view of the predicted frame with a black background.

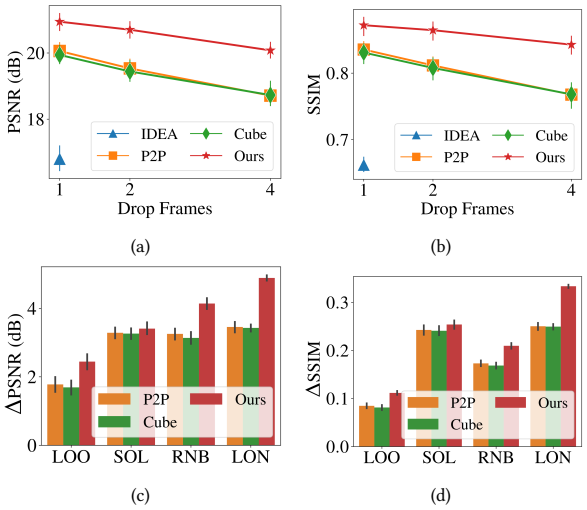

Figure 6: The 2D video quality of error concealment: (a) sample PSNR and (b) sample SSIM of concealed frames from RedandBlack under different numbers of dropped frames; (c) PSNR and (d) SSIM improvements from our motion vectors and heuristics, compared to a state-of-the-art neural network under a single dropped frame.

reveals that some outlying points appear in the background area, causing high HD. Fig. 5 gives a visual example, whereas Fig. 5(a) gives a ample input frame rendered into a 2D view with a white background. Because these outlying points are in white, they impose no negative impacts on the dynamic 3DGS trainers and 2D video quality. Fig. 5(b) zooms into a region with a black background, demonstrating the negative impact of the white outlying points on HD. We believe an updated loss function could mitigate this issue on HD and even improve 2D video quality, which is one of our future tasks.

Our motion vectors improve the performance of error concealment. Figs. 6(a) and 6(b) show the 2D video quality of the concealed frames from a sample sequence: RedandBlack under different frame drop patterns. These figures show that concealing frames with our derived motion vectors significantly improves over the latest heuristics [13]: improvements by up to 1.29 dB in PSNR and 0.07 in SSIM are observed with 4 consecutive frame drops. Compared to IDEA [39]: improvements by up to 3.93 dB in PSNR and 0.28 in SSIM are achieved with a single frame drop. Note that we did not run IDEA with longer consecutive frame drops because its performance with a single frame drop is already significantly inferior to others. Figs. 6(c) and 6(d) report the 2D video quality improvements achieved by heuristics and our motion vectors, compared to IDEA. Across all sequences, our motion vectors outperform IDEA by 1.97–3.93 dB in PSNR and 0.09–0.28 in SSIM, confirming the merits of using better-quality motion vectors. Additionally, concealing a missing frame with our motion vectors takes about 200 ms, while doing that with IDEA requires more than 2,000 s.

5 Potential Downstream Applications

Several sample downstream applications could benefit from our motion vectors: (i) **Error concealment** reconstructs missing dynamic point cloud frames by leveraging successfully received frames. Prior arts, like Hung et al. [13], employed a vague definition of point matching to approximate motion vectors. These works could directly benefit from our motion vectors; (ii) **Temporal super-resolution** uses motion vectors to interpolate or extrapolate additional frames for smoother dynamic scenes. Existing solutions based on neural networks [14, 22, 23, 32, 35, 39, 40] could adopt our unique idea to change input point clouds for better performance; (iii) **Source coding** compresses dynamic point cloud sequences by leveraging the spatial and temporal redundancy. Existing source coders [1, 6, 9] define motion vectors on the input point cloud frames, and may borrow our idea.

6 Conclusion and Outlook

We studied the problem of jointly learning point clouds and motion vectors across dynamic point cloud frames, which has never been considered in the literature. Our proposed motion prediction algorithm employs four components: (i) image renderer, (ii) dynamic 3DGS trainer, (iii) dynamic point cloud converter, and (iv) optimal parameter selector, to predict point clouds with explicitly defined motion vectors while retaining high 2D visual quality. Extensive experiments demonstrated the merits of our proposed algorithm, which: (i) delivers much higher 2D video quality, (ii) achieves good 3D point cloud quality, and (iii) benefits downstream applications, like error concealment in point cloud streaming.

Our work can be extended in several directions. For example, while the white outlying points do not affect the image quality of the rendered views, they result in excessive HD and waste storage/transmission resources. Eliminating the outlying points with recently proposed approaches, like Sauart et al. [30], is worth investigating. Moreover, although our motion prediction algorithm can work with any dynamic 3DGS training algorithms [12, 16, 20, 24, 25, 27, 34, 37], we only augment D3DGS [25] in this paper. Applying our methodology to other training algorithms, potentially some hybrid ones is among our future tasks. Last, our current motion prediction algorithm relies on Gaussian splats as proxies. In the future, we could cut the computational complexity by directly predicting non-key point cloud frames with differentiable 3D point cloud rasterization. Moreover, broader comparisons with other related methods and evaluations on a more diverse set of datasets could help further validate the effectiveness and generalization of our method.

References

[1] Anique Akhtar, Zhu Li, and Geert Van der Auwera. 2024. Inter-frame compression for dynamic point cloud geometry coding. *IEEE Transactions on Image Processing* 33 (2024), 584–594.

[2] Anique Akhtar, Zhu Li, Geert Van der Auwera, and Jianle Chen. 2022. Dynamic point cloud interpolation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2574–2578.

[3] Yanqi Bao, Tianyu Ding, Jing Huo, Yaoli Liu, Yuxin Li, Wenbin Li, Yang Gao, and Jiebo Luo. 2025. 3D Gaussian Splatting: Survey, Technologies, Challenges, and Opportunities. *IEEE Transactions on Circuits and Systems for Video Technology* (2025), 1–1.

[4] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A. Chou. 2017. 8i Voxelized Full Bodies, version 2 – A Voxelized Point Cloud Dataset. https://plenodb.jpeg.org/pc/8ilabs

[5] Haoqiang Fan, Hao Su, and Leonidas J Guibas. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 605–613.

[6] Tingyu Fan, Linyao Gao, Yiling Xu, Zhu Li, and Dong Wang. 2022. D-DPCC: Deep Dynamic Point Cloud Compression via 3D Motion Prediction. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 898–904. https://doi.org/10.24963/ijcai.2022/126 Main Track.

[7] Daniel Garrido, Rui Rodrigues, A Augusto Sousa, Joao Jacob, and Daniel Castro Silva. 2021. Point cloud interaction and manipulation in virtual reality. In *2021 5th International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. 15–20.

[8] Pedro de Medeiros Gomes, Silvia Rossi, and Laura Toni. 2024. AGAR-Attention Graph-RNN for Adaptative Motion Prediction of Point Clouds of Deformable Objects. *ACM Transactions on Multimedia Computing, Communications and Applications* 20, 8 (2024), 1–25.

[9] Danillo Graziosi, Ohji Nakagami, Satoru Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali Tabatabai. 2020. An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing* 9 (2020), e13.

[10] Reetu Hooda, W David Pan, and Tamseel M Syed. 2022. A survey on 3D point cloud compression using machine learning approaches. *SoutheastCon 2022* (2022), 522–529.

[11] Alain Horé and Djemel Ziou. 2010. Image Quality Metrics: PSNR vs. SSIM. In *Proc. of IEEE International Conference on Pattern Recognition*. Istanbul, Turkey, 2366–2369.

[12] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. 2024. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4220–4230.

[13] Tzu-Kuan Hung, I-Chun Huang, Samuel Rhys Cox, Wei Tsang Ooi, and Cheng-Hsin Hsu. 2022. Error concealment of dynamic 3d point cloud streaming. In *Proceedings of the 30th ACM International Conference on Multimedia*. 3134–3142.

[14] Chaokang Jiang, Dalong Du, Jiuming Liu, Siting Zhu, Zhenqiang Liu, Zhuang Ma, Zhujin Liang, and Jie Zhou. 2024. NeuroGauss4D-PCI: 4D Neural Fields and Gaussian Deformation Fields for Point Cloud Interpolation. *arXiv preprint arXiv:2405.14241* (2024).

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023).

[16] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. 2025. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. In *European Conference on Computer Vision*. Springer, 252–269.

[17] Li Li, Zhu Li, Vladyslav Zakharchenko, Jianle Chen, and Houqiang Li. 2019. Advanced 3D motion prediction for video-based dynamic point cloud compression. *IEEE Transactions on Image Processing* 29 (2019), 289–302.

[18] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A Chapman, Dongpu Cao, and Jonathan Li. 2020. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems* 32, 8 (2020), 3412–3432.

[19] Sohee Lim, Minwoo Shin, and Joonki Paik. 2022. Point cloud generation using deep adversarial local features for augmented and mixed reality contents. *IEEE Transactions on Consumer Electronics* 68, 1 (2022), 69–76.

[20] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. 2024. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21136–21145.

[21] Zhi Liu, Qiyue Li, Xianfu Chen, Celimuge Wu, Susumu Ishihara, Jie Li, and Yusheng Ji. 2021. Point cloud video streaming: Challenges and solutions. *IEEE Network* 35, 5 (2021), 202–209.

[22] Fan Lu, Guang Chen, Zhijun Li, Lijun Zhang, Yinlong Liu, Sanqing Qu, and Alois Knoll. 2021. Monet: Motion-based point cloud prediction network. *IEEE Transactions on Intelligent Transportation Systems* 23, 8 (2021), 13794–13804.

[23] Fan Lu, Guang Chen, Sanqing Qu, Zhijun Li, Yinlong Liu, and Alois Knoll. 2021. Pointinet: Point cloud frame interpolation network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 2251–2259.

[24] Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 2024. 3d geometry-aware deformable gaussian splatting for dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8900–8910.

[25] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. 2024. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In *3DV*.

[26] Cristiano Santos, Mateus Gonçalves, Guilherme Corrêa, and Marcelo Porto. 2021. Block-based inter-frame prediction for dynamic point cloud compression. In *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3388–3392.

[27] Richard Shaw, Michal Nazarczuk, Jifei Song, Arthur Moreau, Sibi Catley-Chandar, Helisa Dhamo, and Eduardo Pérez-Pellitero. 2025. Swings: sliding windows for dynamic 3D gaussian splatting. In *European Conference on Computer Vision*. Springer, 37–54.

[28] Guihua Shi, Chih-Chun Wu, and Cheng-Hsin Hsu. 2023. Error Concealment of Dynamic LiDAR Point Clouds for Connected and Autonomous Vehicles. In *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 5409–5415.

[29] André L Souto, Ricardo L de Queiroz, and Camilo Dorea. 2020. A 3D motion vector database for dynamic point clouds. *arXiv preprint arXiv:2008.08438* (2020).

[30] Lewis AG Stuart and Michael P Pound. 2025. 3DGS-to-PC: Convert a 3D Gaussian Splatting Scene into a Dense Point Cloud or Mesh. *arXiv preprint arXiv:2501.07478* (2025).

[31] Yuan-Chun Sun, I-Chun Huang, Yuang Shi, Wei Tsang Ooi, Chun-Ying Huang, and Cheng-Hsin Hsu. 2023. A Dynamic 3D Point Cloud Dataset for Immersive Applications. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. 376–383.

[32] Irene Viola, Jelmer Mulder, Francesca De Simone, and Pablo Cesar. 2019. Temporal interpolation of dynamic digital humans using convolutional neural networks. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. IEEE, 90–907.

[33] Cheng-Hao Wu, Chih-Fan Hsu, Ting-Chun Kuo, Carsten Griwodz, Michael Riegler, Géraldine Morin, and Cheng-Hsin Hsu. 2020. PCC arena: a benchmark platform for point cloud compression algorithms. In *Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems*. 1–6.

[34] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20310–20320.

[35] Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver Van Kaick, Hao Zhang, and Hui Huang. 2020. RPM-Net: recurrent prediction of motion and parts from point cloud. *arXiv preprint arXiv:2006.14865* (2020).

[36] Su Yang, Miaole Hou, and Songnian Li. 2023. Three-dimensional point cloud semantic segmentation for cultural heritage: a comprehensive review. *Remote Sensing* 15, 3 (2023), 548.

[37] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. 2024. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20331–20341.

[38] Maosheng Ye, Tongyi Cao, and Qifeng Chen. 2021. Tpcn: Temporal point cloud networks for motion forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11318–11327.

[39] Yiming Zeng, Yue Qian, Qijian Zhang, Junhui Hou, Yixuan Yuan, and Ying He. 2022. Idea-net: Dynamic 3d point cloud interpolation via deep embedding alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6338–6347.

[40] Zehan Zheng, Danni Wu, Ruisi Lu, Fan Lu, Guang Chen, and Changjun Jiang. 2023. Neuralpci: Spatio-temporal neural field for 3d point cloud multi-frame non-linear interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 909–918.

[41] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).