TSDF-Based Efficient Motion-Compensated Temporal Interpolation for 3D Dynamic Sequences

Soowoong Kim[1]*, Minseong Kwon[2]*, Junho Choi[2], Gun Bang[1], Seungjoon Yang[2]†

[1]Electronics and Telecommunications Research Institute
[2]Ulsan National Institute of Science and Technology
{soowoong.kim, gbang}@etri.re.kr, {minseongkwon, junhochoi, syang}@unist.ac.kr

Abstract

This paper introduces a method for efficiently interpolating 3D dynamic sequences using truncated signed distance function (TSDF) volumes. The method calculates bi-directional motions between TSDF volumes of two frames and refines them to reconstruct intermediate Unlike point cloud-based methods, which can suffer from varying and irregular point densities, the uniform and dense grid structure of TSDF offers a consistent framework for estimating the true motion of objects within a scene. In our experiments, the TSDF-based method offers more precise and reliable smooth motion prediction compared to the often error-prone surface depiction in point clouds. Experimental results demonstrate improved accuracy and reduced computational complexity, making it suitable for real-time applications.

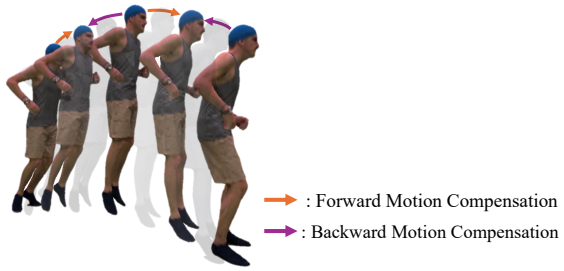


Figure 1: Visualization of our TSDF-based temporal interpolation. Intermediate TSDF is generated by a network that considers bi-directional flow and time.

Introduction

3D scanning devices, such as lidars, multi-view camera arrays, structured-light cameras, and time-of-flight (ToF) cameras, have a slower frame rate compared to 2D cameras. These devices require more computational power and time to capture color and depth information for each point in a scene. Additionally, the depth sensors used in 3D scanning devices often have a slower response time than traditional 2D camera sensors. Synchronizing multiple sensors in 3D scanning setups can also introduce delays and reduce the system's effective frame rate. Typically, these 3D acquisition devices provide 15∼30 frames per second (fps) of data.

High frame rates are extremely important for 3D applications such as virtual reality to deliver a smooth and immersive experience (Hofmeyer et al. 2019). Low frame rates are known to cause motion sickness. Higher frame rates help reduce blur and latency, which makes virtual reality more comfortable and immersive. Achieving and maintaining high frame rates is essential for a truly compelling virtual reality experience. 3D displays offer an immersive user experience and avoid motion sickness by supporting a high frame rate of 90 fps.

When the frame rates of 3D data do not match the refresh rates of display devices, lower-rate data is repeatedly shown on high-rate displays. This results in jerky motion of objects in a scene, which is known as motion judder (Roberts 2002; Allison et al. 2017). Motion judder is generally considered undesirable as it disrupts the viewer's immersion and can detract from the overall viewing experience. Temporal interpolation is a useful technique that involves adding frames between existing frames to reduce motion judder and create smoother motion (Ojo and de Haan 1997; Lee et al. 2002; Thaipanich, Wu, and Kuo 2009). This is achieved by estimating the motion of objects in a scene and displacing them in the inserted intermediate frames to create a consistent appearance that matches the eye's tracking trajectories of objects.

Accurate temporal interpolation for 3D dynamic sequences can be a challenging task. Real-time rendering is necessary to ensure a high level of immersion and prevent motion sickness. Display devices limit the computational complexity, so the conversion algorithm must be light enough to be executed at the display device yet still effective enough to provide high-quality conversion. Latency can negatively impact the user experience. To render a view, the viewpoint information due to the user's head or eye movement must be accessible. If the conversion starts after the viewpoint information becomes available, there can be a significant delay since only the past views can be interpolated. Therefore, the up-conversion needs to be done ahead with 3D data itself independently from viewpoints, allowing the views to be rendered as soon as the viewpoint information is available.

*These authors contributed equally.
†Corresponding author.

This work presents an efficient interpolation method for interpolating 3D dynamic sequence. The main idea is illustrated in Fig. 1. The approach involves estimating bi-directional motions between TSDF volumes of two frames and refining them to reconstruct intermediate frames. Unlike point cloud-based methods, TSDF contains information not only about the surface itself but also about all the surrounding voxels in the space. This enables the utilization of all available information in the space, resulting in much smoother motion vector fields that accurately depict the real motion of objects. Unlike dynamic NeRF and Gaussian Splatting-based methods, which require extensive optimization time to embed motion information across all frame sequences, the proposed method directly estimates and interpolates motion between two consecutive frames (previous and current) in real-time. Experimental results demonstrate that the interpolated frames exhibit fewer spurious errors, leading to a significant improvement in the geometric accuracy compared to pointcloud-based methods. Additionally, the method represents color information as volumes and interpolates the color information using the motion estimated from the surface volumes, ultimately achieving efficient interpolation of 3D dynamic scenes. We were able to achieve significantly improved rendered images compared to NeRF-based and Gaussian Splatting-based methods.

Contributions

- We have developed a real-time motion-compensated temporal interpolation method for 3D dynamic sequences, based on TSDF.
- The uniform and dense grid structure of TSDF volumes allows us to estimate smooth real motion vector fields, which are suitable for compensating objects' motion.
- Intermediate motion-compensated frames accurately depict moving objects' trajectories, reducing spurious errors.
- Motion estimated from the TSDF volume is utilized to interpolate the color volumes, enabling efficient temporal interpolation of textured 3D dynamic sequences.

Related Works

2D and 3D Motion Flow Estimation Flow estimation, encompassing both 2D optical flow and 3D scene flow, has seen significant advancements in recent years through deep learning techniques. (Ilg et al. 2017) introduced FlowNet 2.0, which marked a significant milestone in optical flow estimation using deep networks. Their work was built upon the original FlowNet (Dosovitskiy et al. 2015), incorporating stacked networks and a warping layer to enhance the accuracy of flow estimation. By utilizing a large amount of synthetic data, FlowNet 2.0 demonstrated superior performance compared to traditional methods. Extending flow estimation to 3D point clouds, (Liu, Qi, and Guibas 2019) developed FlowNet3D. This model integrates PointNet++ (Qi et al. 2017) with a novel flow embedding layer, enabling the direct learning of scene flow in 3D point clouds. FlowNet3D effectively captures geometric relationships and motion information, setting a foundation for future research in 3D scene flow estimation. (Wu et al. 2020) presented PointPWC-Net, which applies the cost volume approach from 2D optical flow to 3D point clouds. By using a pyramid, warping, and cost volume strategy, PointPWC-Net estimates scene flow in a self-supervised manner. This approach significantly improves accuracy, particularly in handling large displacements and detailed scene flow estimation.

RAFT (Recurrent All-Pairs Field Transforms), introduced by (Teed and Deng 2020), offers a novel optical flow estimation method. RAFT constructs a 4D correlation volume at high resolution and iteratively updates flow estimates using a recurrent unit. This method achieves state-of-the-art performance across several benchmarks, setting a new standard for optical flow accuracy and efficiency. Building on RAFT developed RAFT-3D (Teed and Deng 2021) for scene flow estimation. RAFT-3D employs rigid-motion embeddings to manage the complex motion in 3D scenes effectively. Extending the RAFT framework to point clouds, RAFT-3D maintains high accuracy and robustness in diverse scenarios. Lastly, (Wei et al. 2021) introduced PV-RAFT, which combines point and voxel-based representations to estimate scene flow. This hybrid approach captures both fine-grained details and global context, enhancing performance in 3D scene flow tasks. PV-RAFT demonstrates the advantages of integrating different data representations to improve scene flow estimation.

3D Dynamic Sequence Frame Interpolation 3D dynamic sequence frame interpolation has emerged as a vital area of research aimed at improving the temporal resolution and quality of 3D data sequences. Various methods have been developed based on point cloud representations. (Lu et al. 2021) introduced PointINet, which estimates bi-directional 3D scene flow between consecutive point clouds and warps them to generate intermediate frames. This method demonstrated effectiveness through extensive experiments on large-scale outdoor LiDAR datasets. (Zeng et al. 2022) proposed IDEA-Net, leveraging deep embedding alignment and a coarse-to-fine strategy to align and interpolate point clouds, capturing temporal consistency and compensating for trajectory errors. (Akhtar et al. 2022) contributed with a method tailored for dynamic scenes, focusing on accurate interpolation between point cloud frames and utilizing optimization techniques to ensure temporal coherence and spatial accuracy. (Zheng et al. 2023) introduced NeuralPCI, a spatio-temporal neural field framework that integrates multi-frame information into an end-to-end deep learning framework, handling large non-rigid deformations effectively. (Zhao et al. 2023) developed a technique for learning spatial-temporal embeddings for sequential point cloud frame interpolation, enhancing interpolation quality by embedding both spatial and temporal information. This approach has been validated through comprehensive experiments, highlighting its effectiveness in improving the temporal resolution of dynamic point clouds.

Despite these advancements, there remains a notable gap in the research on frame interpolation for TSDF volume sequences. Current methods have primarily focused on point

cloud data, and further exploration is needed to address the unique challenges and potentials of TSDF volume frame interpolation. Representing a dynamic 3D sequence as a TSDF volume allows for a clear definition of continuous surfaces, making mesh reconstruction via the marching cubes algorithm (Curless and Levoy 1996; Newman and Yi 2006) much faster and simpler compared to using point clouds.

Dynamic Neural Representation Recent research has seen a surge in efforts to represent dynamic scenes using neural representation techniques. Among these, NeRF-based methods have become particularly prominent. D-NeRF extends the original NeRF framework to model temporal changes in dynamic scenes (Pumarola et al. 2021). Tensor4D introduces an efficient neural 4D decomposition approach for high-fidelity dynamic reconstruction and rendering (Shao et al. 2023), while Sync-NeRF generalizes dynamic NeRFs to handle unsynchronized video inputs (Kim et al. 2024).

In addition to NeRF-based approaches, there are techniques based on 3D Gaussian Splatting. Deformable Gaussian Splatting represents dynamic scenes by deforming Gaussian splats over time to capture complex motions (Yang et al. 2024). 4D Gaussian Splatting enhances real-time dynamic scene rendering by efficiently managing temporal data (Wu et al. 2024), and Gaussian-Flow proposes a 4D reconstruction method utilizing dynamic 3D Gaussian particles (Lin et al. 2024).

These approaches offer several advantages, such as high-quality rendering and the ability to accurately capture the intricate details of dynamic scenes. However, they also present notable challenges. The training process is often computationally intensive and typically requires access to the entire sequence of data. Additionally, these methods can suffer from overfitting, which may lead to poor generalization when generating intermediate frames, thus limiting their effectiveness in certain scenarios (Shamsian et al. 2024).

TSDF-Based Motion-Compensated Temporal Interpolation

A neural network-based temporal interpolation architecture proposed in this study is shown in Fig. 2. Given two TSDF volumes $\mathbf{x}_0$ and $\mathbf{x}_1$, our objective is to estimate the intermediate volume $\mathbf{x}_t$ at an arbitrary time $t \in (0, 1)$. The TSDF volumes $\mathbf{x}_0, \mathbf{x}_t, \mathbf{x}_1 \in \mathbb{R}^{D \times H \times W}$ are represented as a 3D voxel grid where each voxel contains the signed distance to the nearest surface, truncated to a specified maximum distance (Curless and Levoy 1996). We estimate the intermediate TSDF volume $\mathbf{x}_t$ using a three-stage approach: (i) bi-directional flow estimation, leveraging RAFT for precise optical flow computation; (ii) attention-based intermediate flow estimation; and (iii) adaptive TSDF merging. Finally, a polygonal surface mesh with associated color can be reconstructed from the volume data using the Marching Cubes algorithm (Newman and Yi 2006).

Bi-Directional Flow Estimation

Our model is based on RAFT (Teed and Deng 2020) but with key modifications to accommodate the 3D nature of TSDF volumes and to enable bi-directional flow estimation.

Feature and Context Encoding Given a pair of TSDF volume frames, $\mathbf{x}_0$ and $\mathbf{x}_1$, we first process each through two separate encoders: a feature encoder and a context encoder. The feature encoder maps the input volumes into dense feature representations, which are subsequently used to construct bi-directional correlation volumes. Specifically, the feature encoder outputs pairs of feature volumes, $\mathbf{u}_0, \mathbf{u}_1 \in \mathbb{R}^{d \times h \times w \times C}$, each with C channels and at $1/8$ of the input volume resolution, *i.e.*, $d = D/8$, $h = H/8$, and $w = W/8$. In parallel, the context encoder, which shares the same architecture as the feature encoder but with distinct weights, extracts semantic and contextual information from the volumes. The resulting context vectors, $\mathbf{c}_0$ and $\mathbf{c}_1$, maintain the same spatial resolution as the outputs of the feature encoder.

Bi-directional Correlation Volumes Unlike RAFT, which constructs a uni-directional correlation volume, we create a bi-directional correlation volume. We compute a 6D correlation volume, $\mathbf{C} \in \mathbb{R}^{d \times h \times w \times d \times h \times w}$, by taking the dot product between all pairs of feature vectors from the input volumes:

$$\mathbf{C}_{ijklmn} = \sum_h (\mathbf{u}_0)_{ijkh} \cdot (\mathbf{u}_1)_{lmnh}. \tag{1}$$

To capture similarities across scales, the last three dimensions of the correlation volume are downsampled using a repeated 3D average pooling layer with a kernel size of 2 and a stride of 2, resulting in a 4-level correlation pyramid $\{\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3\}$. Thus, the volume $\mathbf{C}_k$ for $k = 0, 1, 2, 3$ has dimensions $d \times h \times w \times d/2^k \times h/2^k \times w/2^k$. In contrast to RAFT's uni-directional correlation pyramid, which reflects multi-scale correspondences from $\mathbf{x}_0$ to $\mathbf{x}_1$, we compute correlation volume in the reverse direction as well:

$$\mathbf{C}^T_{ijklmn} = \sum_h (\mathbf{u}_1)_{ijkh} \cdot (\mathbf{u}_0)_{lmnh}. \tag{2}$$

After obtaining the transposed correlation volume $\mathbf{C}^T$, we perform the same pooling operations to form the backward correlation pyramid $\{\mathbf{C}^T_0, \mathbf{C}^T_1, \mathbf{C}^T_2, \mathbf{C}^T_3\}$. These bi-directional correlation volumes need to be computed only once.

Following the construction of the correlation matrix, correlation lookup operators generate feature maps by indexing from the correlation pyramid. Similar to RAFT, these operators use local neighborhoods around the projected position on the deformed volume to index the correlation volumes for each pyramid level. The grids are interpolated using trilinear sampling and then concatenated to form feature maps.

Forward and Backward Flow Estimation Using the constructed correlation pyramids, we estimate the forward flow sequence $\mathbf{f}_{0\to1}$ from $\mathbf{x}_0$ to $\mathbf{x}_1$ and the backward flow sequence $\mathbf{f}_{1\to0}$ from $\mathbf{x}_1$ to $\mathbf{x}_0$. This is done iteratively using 3D recurrent update operators, which refine the displacement fields at each step: $f^{(k+1)} = f^{(k)} + \Delta f$. Our model, similar to RAFT, employs GRU-based 3D recurrent update operators to predict and refine a sequence of flow fields

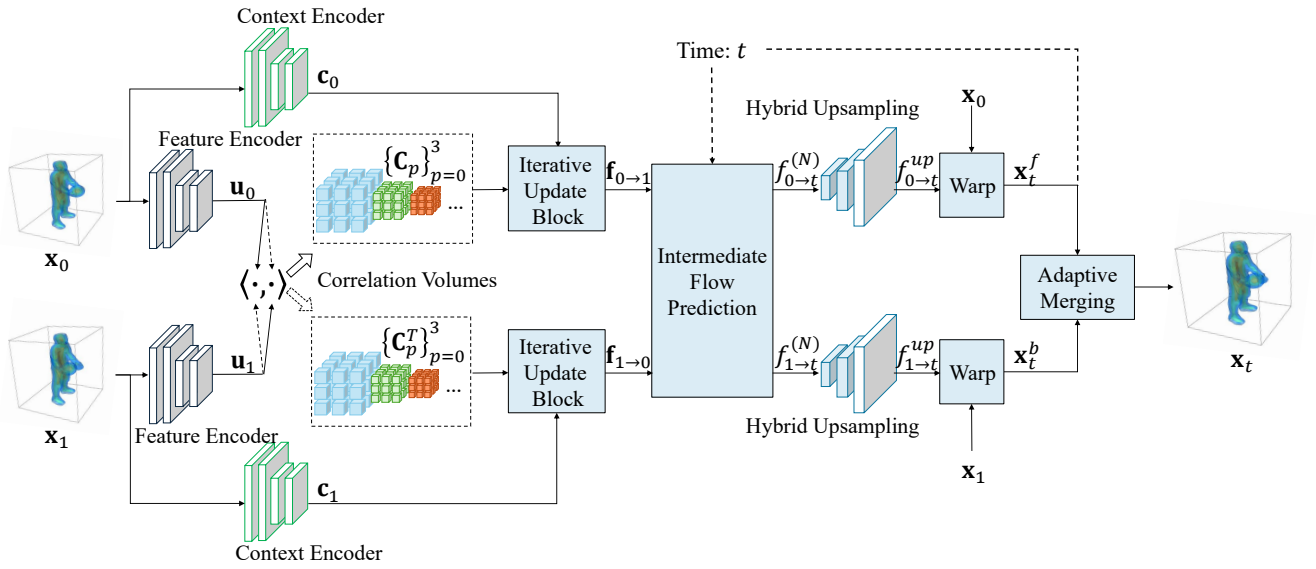


Figure 2: Schematics of the proposed TSDF-based motion-compensated temporal interpolation. The notation $\langle\cdot,\cdot\rangle$ denotes the matrix inner product between volumes. Forward and backward motions are estimated by GRUs, which access the correlation between the features extracted from the given TSDF frames. Estimated bi-directional motion is refined and scaled up to the voxel resolution. Intermediate frame is found by compensating objects' real motion.

$\{f^{(1)}, \ldots, f^{(N)}\}$, with the final flow used for the next step, $f = f^{(N)}$. The GRU consists of the following components:

$$z_k = \sigma(\text{Conv}_{3\times3\times3}([h_{k-1}, y_k]; \theta_z)), \quad (3)$$
$$r_k = \sigma(\text{Conv}_{3\times3\times3}([h_{k-1}, y_k]; \theta_r)), \quad (4)$$
$$q_k = \tanh(\text{Conv}_{3\times3\times3}([r_k \cdot h_{k-1}, y_k]; \theta_q)), \quad (5)$$
$$h_k = (1 - z_k) \cdot h_{k-1} + z_k \cdot q_k, \quad (6)$$

where σ denotes the sigmoid function, $\cdot$ represents element-wise multiplication, and y_k is the concatenation of flow, correlation, and context features defined earlier. The hidden state h_k is then processed through two convolutional layers to predict the motion update at a lower resolution.

Intermediate Flow Estimation

Given the forward flow $f_{0\to1}$ and backward flow $f_{1\to0}$ estimated from the previous step, the intermediate flows $f_{0\to t}$ and $f_{1\to t}$ could be approximated linearly by scailing the original flows:

$$f_{0\to t} \approx t \cdot f_{0\to1} \quad (7)$$
$$f_{1\to t} \approx (1 - t) \cdot f_{1\to0}. \quad (8)$$

However, instead of using this linear approximation, we employ a neural network to compute the intermediate flows non-linearly:

$$(f_{0\to t}, f_{1\to t}) = g_\theta(f_{0\to1}, f_{0\to1}, t), \quad (9)$$

where $g_\theta(\cdot)$ is a neural network with parameter θ trained to predict the intermediate flows. Simplified attention modules (Cheng et al. 2020) are incorporated to enhance the performance by paying closer attention to complex and non-linear flows. By leveraging a neural network for intermediate flow estimation, we significantly improve the ability to model non-linear and complex motions in 3D space. This approach leads to more accurate and robust predictions of the intermediate TSDF volumes, ultimately enhancing the quality of the reconstructed intermediate surfaces. The detailed network architecture is given in Fig. 3.

Hybrid Upsampling While RAFT uses a convex upsampling technique, we apply a hybrid upsampling method. This technique combines 4x neural network-based upsampling followed by 2x trilinear upsampling:

$$f^{\text{up}} = \text{TrilinearUpsample}(\text{ConvexUpsample}(f)). \quad (10)$$

This approach balances computational efficiency and accuracy, avoiding the complexity of convex upsampling while still providing high-quality flow estimation. These refined bi-directional motion vectors, scaled up to the voxel resolution, are crucial for accurately compensating for object motion and reconstructing intermediate TSDF volumes.

Adaptive TSDF Merging

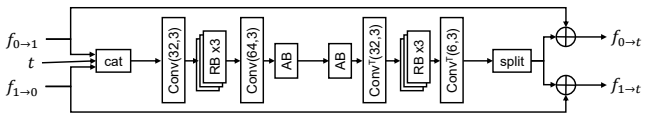
Given the intermediate flows $f_{0\to t}^{\text{up}}$ and $f_{1\to t}^{\text{up}}$, we have two candidates for estimating the intermediate TSDF volume $\mathbf{x}_t$:

1. $\mathbf{x}_t^f = \mathcal{W}(\mathbf{x}_0, f_{0\to t}^{\text{up}})$: The volume warped from $\mathbf{x}_0$ using the forward flow.
2. $\mathbf{x}_t^b = \mathcal{W}(\mathbf{x}_1, f_{1\to t}^{\text{up}})$: The volume warped from $\mathbf{x}_1$ using the backward flow.

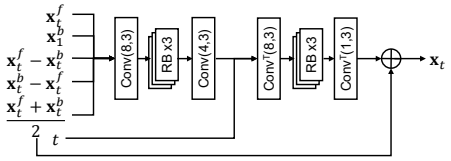
To obtain the final intermediate volume $\mathbf{x}_t$, instead of simply averaging these candidates, we propose a network that merges them adaptively based on the time parameter t:

$$\mathbf{x}_t = g_\phi(\mathbf{x}_t^f, \mathbf{x}_t^b, t). \quad (11)$$

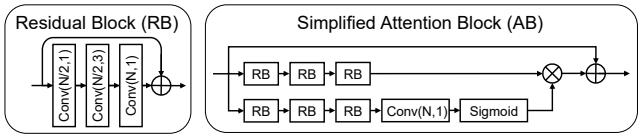
Our implementation incorporates the mean and difference of $\mathbf{x}_t^f$ and $\mathbf{x}_t^b$ as input to the network. This enables informed



(a) Intermediate flow estimation



(b) Adaptive TSDF merging



(c) Residual block (RB) and simplified attention block (AB)

Figure 3: Network architecture for intermediate flow estimation and adaptive TSDF merging. conv3d(C, k) denotes 3d convolution with C out channels and k kernel size.

decisions about volume reliability and better understanding of the relationship between the warped volumes, resulting in higher-quality reconstructed surfaces compared to simple averaging. The detailed network architecture is given in Fig. 3.

Training of End-to-end Network

Since ground truth motion flows are unavailable, the training objective is to minimize the distance between the ground truth TSDF volumes and the deformed TSDF volumes. Additionally, the flow estimation loss is considered, as accurate motion estimation is crucial for effective refinement.

The total loss $\mathcal{L}$ is a weighted sum of three losses:

$$\mathcal{L} = \alpha_f \mathcal{L}_f + \alpha_i \mathcal{L}_i + \alpha_m \mathcal{L}_m. \tag{12}$$

Flow Estimation Loss $\mathcal{L}_f$: This loss is computed using an exponential-weighted L1 loss, similar to the approach used in RAFT (Teed and Deng 2020). Specifically, the loss is computed over the full sequence of N predicted flows $\{f^{(1)}, \ldots, f^{(N)}\}$, with exponentially increasing weights:

$$\mathcal{L}_f = \sum_i \gamma^{N-i} \left(\|\mathbf{x}_1^{\text{GT}} - \mathbf{x}_1^{(i)}\|_1 + \|\mathbf{x}_0^{\text{GT}} - \mathbf{x}_0^{(i)}\|_1 \right), \tag{13}$$

where

$$\mathbf{x}_1^{(i)} = \mathcal{W}(\mathbf{x}_0, f_{0\to1}^{\text{up},(i)}), \quad \mathbf{x}_0^{i} = \mathcal{W}(\mathbf{x}_1, f_{1\to0}^{\text{up},(i)}). \tag{14}$$

Intermediate Flow Loss $\mathcal{L}_i$: This loss is calculated as the L1 loss with intermediate TSDFs warped with predicted intermediate flow:

$$\mathcal{L}_i = \|\mathbf{x}_t^{\text{GT}} - \mathbf{x}_t^f\|_1 + \|\mathbf{x}_t^{\text{GT}} - \mathbf{x}_t^b\|_1. \tag{15}$$

Merging Loss $\mathcal{L}_m$: This loss is computed as the L1 loss between the finally merged intermediate frame TSDF and the ground truth:

$$\mathcal{L}_m = \|\mathbf{x}_t^{\text{GT}} - \mathbf{x}_t\|_1. \tag{16}$$

In our experiments, we set the hyperparameters as follows: $N = 12$, $\alpha_f = \frac{1}{2N}$, $\alpha_i = \frac{1}{2}$, $\alpha_m = 1$, and $\gamma = 0.8$. This training strategy effectively balances the estimation of flow fields and the reconstruction of intermediate volumes, facilitating high-quality temporal interpolation of the TSDF volumes.

Experiments and Results

Experimental Setup

Datasets The study employed a diverse set of dynamic mesh datasets for training and evaluation: the MPEG-curated dynamic mesh dataset (Xu, Lu, and Wen 2021), MITAMA (Vlasic et al. 2008), and 4DHumanOutfit (4DHO) (Armando et al. 2023). For evaluation, we selected three sequences (Longdress, Loot, Soldier) from the MPEG dataset, three sequences (Swing, Crane, Handstand) from the MITAMA dataset, and four sequences (deb-jea-walk, leo-tig-jump, ray-own-torso, ted-opt-side) from the 4DHO dataset. The remaining sequences from each dataset were used for training: seven sequences from the MPEG dataset, seven from MITAMA, and nine from 4DHO.

Prior to training, the dynamic mesh data was preprocessed by converting the meshes into TSDF volumes. This involved partitioning the bounding box into a $128 \times 128 \times 128$ grid and computing the signed distance from each voxel query point to the surface, truncated to the range of $[-1, 1]$. Detailed information regarding these datasets is provided in the **supplementary material**.

Baselines We compared our approach with several state-of-the-art models across various 3D representation techniques. Specifically, we benchmarked against NeuralPCI (Zheng et al. 2023) for point-cloud interpolation, D-NeRF (Pumarola et al. 2021) for NeRF-based interpolation, and Deformable-GS (Yang et al. 2024) for Gaussian Splatting-based interpolation.

NeuralPCI samples 8096 points from a mesh, takes two frames as input and generates interpolated point clouds. Each optimization step is set to 500 iterations. D-NeRF and Deformable-GS take rendered images from randomly generated viewpoints as input to learn the entire scene. D-NeRF is optimized over 800,000 iterations, while Deformable-GS is optimized over 4000 iterations.

Metrics We quantitatively evaluated our method from two perspectives. First, we measured geometric distance using Chamfer Distance (CD) (Fan, Su, and Guibas 2017) and Earth Mover's Distance (EMD) (Rubner, Tomasi, and Guibas 2000) when comparing with NeuralPCI. Before computing CD and EMD, we normalized the interpolated outputs and ground truth to the range $[0, 1]$. Second, we evaluated the peak signal-to-noise ratio (PSNR) between the rendered and ground truth images from randomly sampled viewpoints when comparing with D-NeRF and Deformable-GS.

Methods	MPEG		MITAMA		4DHO		Overall	
	CD	EMD	CD	EMD	CD	EMD	CD ↓	EMD ↓
NeuralPCI	0.095	**9.502**	0.066	12.903	0.080	14.608	0.080	12.338
Ours	**0.089**	10.456	**0.026**	**5.649**	**0.016**	**4.441**	**0.043**	**6.849**

Table 1: Accuracy for $\times 3$ frame rate conversion in CD and EMD, average measures ($\times 10^{-3}$) of interpolated frames are shown.

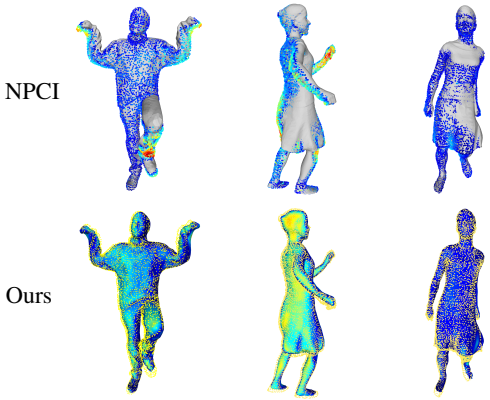


Figure 4: Temporal interpolation accuracy (geometry). NPCI vs Ours. 1st row: GT mesh vs NPCI point cloud, 2nd row: GT sampled vertices vs Ours mesh. Hausdorff distance from relevant GT vertices is shown in color, blue: close and red: far from original mesh.

Temporal Interpolation Accuracy (Geometry)

Table 1 shows the quantitative accuracy comparison with NeuralPCI, which shows the best performance among the pointcloud-based temporal interpolation methods (Zheng et al. 2023), across various datasets. Our method significantly improves performance on the MITAMA and 4DHO datasets and performs comparably on the MPEG dataset.

Point cloud-based methods focus on how points move between frames, while our method focuses on how TSDF values in neighboring voxels move. Consequently, our approach suffers less from spurious errors, resulting in smoother motion vector fields. The intermediate frames have fewer errors, leading to a significant improvement in CD and EMD.

Fig. 4 illustrates the difference between point-based frame interpolation and our approach using TSDF volume. The top row shows the 8192 predicted points by NeuralPCI and the ground truth mesh. Even after the warping process, the point density is inconsistent due to random initialization. Moreover, spurious errors can be observed in regions with fast motion, for example the ankle and hands. In contrast, our method, shown in the bottom row, better captures the movement of the whole scene, making it more suitable for frame interpolation.

Temporal Interpolation Accuracy (Image)

Fig. 5 shows the rendered result of each method. D-NeRF struggles to predict the intermediate frame at an unseen timestamp. Deformable-GS also suffers from intermittent blurring and ghosting effects where the movement happens. An input sequence used in the optimiztation steps of these methods may not include enough dynamic scene information for accurate temporal interpolation. Our method does not show any ghosting effect because the TSDF values are interpolated to generate geometric surface.

Table 2 shows the PSNR evaluated on each rendered images. As shown in the table, our method results in the highest PSNR in all cases and Deformable-GS shows comparable results to ours in some cases.

Dataset		D-NeRF	Deformable-GS	Ours
4DHO	leo-tig-jump	17.031	27.753	36.760
	deb-jea-walk	16.084	32.674	37.833
	ray-own-torso	17.200	32.477	37.165
	ted-opt-side	14.684	29.213	35.291
	Overall↑	16.250	30.529	**36.762**
MPEG	longdress	20.500	26.352	31.584
	soldier	19.870	33.276	33.748
	loot	19.173	32.135	32.958
	Overall↑	19.848	30.588	**32.763**

Table 2: PSNR comparison.

Steps	NPCI	D-NeRF	Deformable -GS	Ours	Ours (geometry)
Optimization	929.159	683.092	4.471	-	-
Inference	0.618	8.340	0.122	0.338	0.285

Table 3: Time complexity for generating one intermediate frame. Optimization time is the average time in seconds to optimize the entire scene. (500, 800000, and 40000 iterations were used for NPCI, D-NeRF, and Deformable-GS respectively.) Inference time is seconds per frame.

Computational Complexity

The computational complexity in terms of processing time is shown in Table 3. Other state-of-the-art methods in comparison require very long scene-wise optimization processes. In contrast, once trained, our method can operate without any scene-wise preparation step, which is a big advantage for the temporal interpolation task. The inference time per frame is also reported. The time complexity comparison is difficult because the data the methods operate and produce are in different forms. For example, D-NeRF and Gaussian Splatting outputs rendered 2D images, NPCI outputs a set of vertices, and our method produces warped TSDF. The proposed method is considerably faster than NPCI and D-Nerf, but slower than Deformable-GS.

Ablation Study

We conducted ablations to verify the effectiveness of intermediate motion estimation, hybrid upsampling, and adaptive merging in enhancing performance. When the motion estimation module was removed, we compensated for the forward and backward flow linearly to generate intermediate

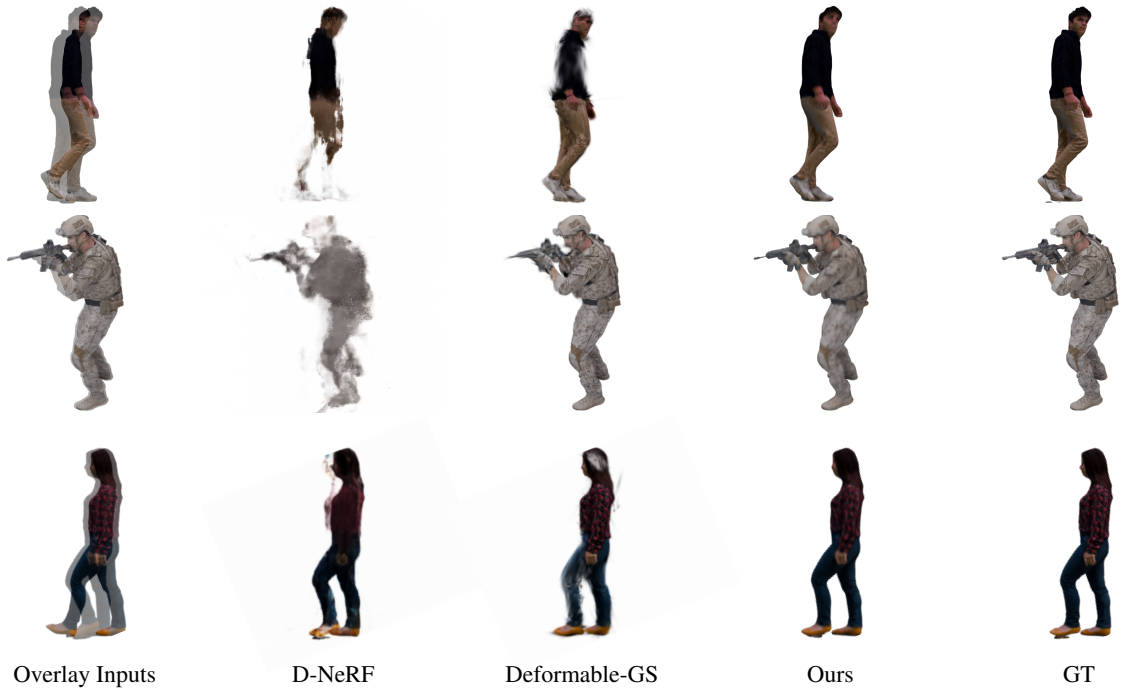


Figure 5: Temporal interpolation accuracy (image). Rendering results are shown.

	Motion Estimation	Hybrid Upsampling	Adaptive Merging	Metrics		Total Gain
				CD ↓	EMD ↓	
A	✗	✓	✓	0.065	8.527	−37.27%
B	✓	✗	✓	0.069	7.782	−36.72%
C	✓	✓	✗	0.047	7.434	−8.84%
D	✓	✓	✓	0.043	6.849	0.0%

Table 4: Ablation Study.

motion, as shown in Eq. (7), (8). Without the hybrid upsampling module, the estimated motions were spatially scaled up to the voxel resolution using trilinear interpolation. When adaptive merging was omitted, we used the average of the forward-warped TSDF and the backward-warped TSDF.

Table 4 demonstrates the necessity of each component. Notably, omitting motion estimation resulted in the most significant performance degradation, suggesting that the estimated flow can be nonlinear and complex. The absence of hybrid upsampling also caused substantial performance drops, likely because small changes in TSDF values can lead to significant differences in reconstruction due to the nature of TSDF volumes. Lastly, the presence or absence of adaptive merging resulted in slight performance variations, indicating that the time-dependent merging approach enables more accurate surface reconstruction.

Figure 6 shows qualitative results with and without learning-based motion estimation and hybrid upsampling. Without motion estimation, the surfaces of detailed parts like hands and feet appeared disconnected. Without hybrid upsampling, additional voxels were created, likely due to unnecessary TSDF value shifts caused by trilinear upsampling.

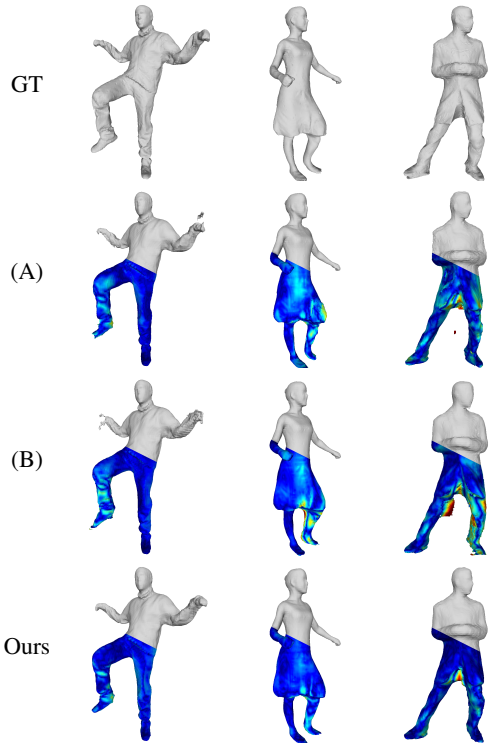


Figure 6: Qualitative results of ablation study. Hausdorff distance increases from blue to red. (A): w/o motion estimation; (B): w/o hybrid upsampling; and ours.

Conclusion

We presented a novel method for efficiently interpolating 3D dynamic sequences using TSDF volumes. Our approach leverages a motion-compensated interpolation framework to enhance motion accuracy, resulting in intermediate frames with smoother motions. Experiments demonstrate that our TSDF-based method generates geometrically more accurate and visually better intermediate frames compared to existing techniques. Our method currently utilizes only TSDF (geometric) information for motion estimation, which occasionally leads to temporal inconsistencies in the texture of the intermediate frames. In future work, we plan to extract and fuse features from both color and TSDF volumes to estimate temporally consistent motion for geometrically and texturally consistent temporal interpolation.

Acknowledgments

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00072, Development of Audio/Video Coding and Light Field Media Fundamental Technologies for Ultra Realistic Tera-media) (No. 2020-0-01336, Artificial Intelligence Graduate School Program (UNIST)). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00359358).

References

Akhtar, A.; Li, Z.; Van der Auwera, G.; and Chen, J. 2022. Dynamic point cloud interpolation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2574–2578. IEEE.

Allison, R. S.; Wilcox, L. M.; Anthony, R. C.; Helliker, J.; and Dunk, B. 2017. Expert Viewers' Preferences for Higher Frame Rate 3D Film. *Electronic Imaging*, 29: 20–28.

Armando, M.; Boissieux, L.; Boyer, E.; Franco, J.-S.; Humenberger, M.; Legras, C.; Leroy, V.; Marsot, M.; Pansiot, J.; Pujades, S.; et al. 2023. 4DHumanOutfit: a multi-subject 4D dataset of human motion sequences in varying outfits exhibiting large displacements. *Computer Vision and Image Understanding*, 237: 103836.

Cheng, Z.; Sun, H.; Takeuchi, M.; and Katto, J. 2020. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7939–7948.

Curless, B.; and Levoy, M. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 303–312.

Dosovitskiy, A.; Fischer, P.; Ilg, E.; Hausser, P.; Hazirbas, C.; Golkov, V.; van der Smagt, P.; Cremers, D.; and Brox, T. 2015. FlowNet: Learning Optical Flow With Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Fan, H.; Su, H.; and Guibas, L. J. 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.

Hofmeyer, F.; Fremerey, S.; Cohrs, T.; and Raake, A. 2019. Impacts of internal HMD playback processing on subjective quality perception. *Electronic Imaging*, 31: 1–7.

Ilg, E.; Mayer, N.; Saikia, T.; Keuper, M.; Dosovitskiy, A.; and Brox, T. 2017. FlowNet 2.0: Evolution of Optical Flow Estimation With Deep Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kim, S.; Bae, J.; Yun, Y.; Lee, H.; Bang, G.; and Uh, Y. 2024. Sync-NeRF: Generalizing Dynamic NeRFs to Unsynchronized Videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2777–2785.

Lee, S.-H.; Shin, Y.-C.; Yang, S.; Moon, H.-H.; and Park, R.-H. 2002. Adaptive motion-compensated interpolation for frame rate up-conversion. *IEEE Transactions on Consumer Electronics*, 48(3): 444–450.

Lin, Y.; Dai, Z.; Zhu, S.; and Yao, Y. 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.

Liu, X.; Qi, C. R.; and Guibas, L. J. 2019. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lu, F.; Chen, G.; Qu, S.; Li, Z.; Liu, Y.; and Knoll, A. 2021. PointINet: Point Cloud Frame Interpolation Network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(3): 2251–2259.

Newman, T. S.; and Yi, H. 2006. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5): 854–879.

Ojo, O. A.; and de Haan, G. 1997. Robust motion-compensated video upconversion. *IEEE Transactions on Consumer Electronics*, 43(4): 1045–1056.

Pumarola, A.; Corona, E.; Pons-Moll, G.; and Moreno-Noguer, F. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10318–10327.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30.

Roberts, A. 2002. The Film Look: It's Not Just Jerky Motion.... *BBC R&D White Paper WHP*, 53: 7.

Rubner, Y.; Tomasi, C.; and Guibas, L. J. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40: 99–121.

Shamsian, A.; Navon, A.; Zhang, D. W.; Zhang, Y.; Fetaya, E.; Chechik, G.; and Maron, H. 2024. Improved generalization of weight space networks via augmentations. *arXiv preprint arXiv:2402.04081*.

Shao, R.; Zheng, Z.; Tu, H.; Liu, B.; Zhang, H.; and Liu, Y. 2023. Tensor4d: Efficient neural 4d decomposition for

high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16632–16642.

Teed, Z.; and Deng, J. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, 402–419. Springer.

Teed, Z.; and Deng, J. 2021. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8375–8384.

Thaipanich, T.; Wu, P.-H.; and Kuo, C.-C. J. 2009. Low complexity algorithm for robust video frame rate up-conversion (FRUC) technique. *IEEE Transactions on Consumer Electronics*, 55(1): 220–228.

Vlasic, D.; Baran, I.; Matusik, W.; and Popović, J. 2008. *Articulated mesh animation from multi-view silhouettes*, 1–9. Association for Computing Machinery.

Wei, Y.; Wang, Z.; Rao, Y.; Lu, J.; and Zhou, J. 2021. PV-RAFT: Point-Voxel Correlation Fields for Scene Flow Estimation of Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6954–6963.

Wu, G.; Yi, T.; Fang, J.; Xie, L.; Zhang, X.; Wei, W.; Liu, W.; Tian, Q.; and Wang, X. 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.

Wu, W.; Wang, Z. Y.; Li, Z.; Liu, W.; and Fuxin, L. 2020. PointPWC-Net: Cost Volume on Point Clouds for (Self-) Supervised Scene Flow Estimation. In *European Conference on Computer Vision*, 88–107. Springer.

Xu, Y.; Lu, Y.; and Wen, Z. 2021. CfP for Dynamic Mesh Coding. In *ISO/IEC JTC 1/SC 29/WG02*, volume 145.

Yang, Z.; Gao, X.; Zhou, W.; Jiao, S.; Zhang, Y.; and Jin, X. 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.

Zeng, Y.; Qian, Y.; Zhang, Q.; Hou, J.; Yuan, Y.; and He, Y. 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 (CVPR)*, 6338–6347.

Zhao, L.; Sun, Z.; Ren, L.; Yin, Q.; Yang, L.; and Guo, M. 2023. Learning Spatial-Temporal Embeddings for Sequential Point Cloud Frame Interpolation. In *2023 IEEE International Conference on Image Processing (ICIP)*, 810–814.

Zheng, Z.; Wu, D.; Lu, R.; Lu, F.; Chen, G.; and Jiang, C. 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 (CVPR)*, 909–918.