國立清華大學
NATIONAL TSING HUA UNIVERSITY

國 立 清 華 大 學
National Tsing Hua University
碩 士 論 文
Master's Thesis

動態點雲與運動向量的聯合學習技術
Joint Learning of Dynamic Point Clouds and Motion Vectors

系所別： 資訊系統與應用研究所

Dept. / Grad. Inst.: Institute of Information Systems and Applications

學號 Student ID: 113065525

研究生 Author： 李承澤 Cheng-Tse Lee

指導教授 Advisor： 徐正炘博士 Cheng-Hsin Hsu Ph.D.

中 華 民 國 一一五 年 六 月

June 2026

國立清華大學
NATIONAL TSING HUA UNIVERSITY

國立清華大學
NATIONAL TSING HUA UNIVERSITY

國 立 清 華 大 學
National Tsing Hua University
碩 士 論 文
Master's Thesis

動態點雲與運動向量的聯合學習技術
Joint Learning of Dynamic Point Clouds and Motion Vectors

系所別： 資訊系統與應用研究所

Dept. / Grad. Inst.: Institute of Information Systems and Applications

學號 Student ID: 113065525

研究生 Author： 李承澤 Cheng-Tse Lee

指導教授 Advisor： 徐正炘博士 Cheng-Hsin Hsu Ph.D.

中 華 民 國 一一五 年 六 月

June 2026

113065525
國立清華大學
NATIONAL TSING HUA UNIVERSITY

國立清華大學學位論文授權書

Authorized Agreement for Thesis/Dissertation

(裝訂於紙本論文書名頁之次頁 For author to bind after the title page)

● 立書人（即論文作者）Author: 李承澤 （以下稱本人 hereinafter referred to as "I ")

● 授權標的 Authorized subject：本人於 國立清華大學 （以下稱學校）電機資訊學院 資訊系統與應用研究所（研究所、學位學程）114學年度第2學期之 ■碩士 □博士 學位論文
I hereby authorize that my ■ Master' s □ Doctoral thesis submitted in the 2 semester of the 114 academic year at National Tsing Hua University (hereinafter referred to as "the University"), College of Electrical Engineering and Computer Science College, Institute of Information Systems and Applications Department / Graduate Institute / Degree Program,
論文題目 Title：動態點雲與運動向量的聯合學習技術
指導教授 Advisor：徐正炘

（以下稱**本著作**，本著作並包含論文全部、摘要、目錄、圖檔、影音以及相關書面報告、技術報告或專業實務報告等，以下同
Hereinafter referred to as "the publication", which contains all thesis/dissertation, abstracts, catalogues, graphic documents, audiovisual reports, technical reports or professional practice reports, etc.)

緣依據**學位授予法**等相關法令，對於本著作及其電子檔，學校圖書館得依法進行保存等利用，而國家圖書館則得依法進行保存、以紙本或讀取設備於館內提供公眾閱覽等利用。此外，為促進學術研究及傳播，本人在此並進一步同意授權(1)**國立清華大學**與(2)**國家圖書館對**本著作進行以下各點所定之利用：

In accordance with the Degree Conferral Act and other relevant laws and regulations, for this publication and its electronic file, the school library can be preserved and used according to the law. The National Central Library must preserve it in accordance with the law and permit public access in the library with paper or reading equipment. In addition, in order to promote academic research and scholarly communication, I hereby agree to authorize (1)**National Tsing Hua University** and the (2)**National Central Library** to use this publication for the following purposes

國立清華大學
NATIONAL TSING HUA UNIVERSITY

113065525
國立清華大學
NATIONAL TSING HUA UNIVERSITY

一、授權部分：

本人**同意**授權國立清華大學與國家圖書館，無償、不限期間與次數重製本著作並得為教育、科學及研究等**非營利用途之利用**，其包括得將本著作之電子檔收錄於數位資料庫，並透過自有或委託代管之伺服器、網路系統或網際網路向位於全球之使用者(包含但不限於國立清華大學校園內、校園外及國家圖書館館內、館外)公開傳輸，以供使用者為非營利目的之檢索、閱覽、下載及/或列印。

I hereby agree to authorize National Tsing Hua University and the National Central Library to reproduce this publication free of charge, without limitation on time or the number of reproductions, and for education, science, research, and other non-profit use. This involves recording the electronic file of this publication into a digital database and the public transmission of them via self-owned or entrusted servers, network systems or the Internet to the user for the purpose of search, reading, downloading or printing for non-profit purposes.

二、本授權書第一點所定授權，均為非專屬且非獨家授權之約定，本人仍得自行或授權任何第三人利用本著作。

The authorized agreement is non-exclusive permission. I retain the right to use the publication myself or to authorize any third party to use it.

三、本授權書第一點所定授權對象，依各該點授權利用本著作時，均應尊重本人著作人格權及權利管理電子資訊等相關權利，不得以任何方式省略、增修或變更本人署名、本著作名稱、本著作內容及相關資料（包括本人原記載取得學位論文之學校全銜、書目等詮釋資料等）。

The authorized parties specified in this authorized agreement, when using this publication, shall respect my moral rights and the rights related to the management of electronic information. They shall not omit, alter, or modify my authorship, the title of the publication, the content of the publication, or related information (including the full name of the school where I obtained my degree, bibliographic details, and other interpretative materials) in any way.

四、依本授權書第一點將本著作及其電子檔對外公開之時間 Public date of the publication：

□ 於完成審核後立即對外公開 Public immediately after completion of the review

■ 於完成審核之六個月後對外公開 Public six months after completion of the review

□ 本人要求本著作應自 Public access after 民國　　年　　月　　日起始得對外公開，故因本授權書第一點所定授權亦應自該日起始生效力。

國立清華大學
NATIONAL TSING HUA UNIVERSITY

2026/05/22 15:06:50

113065525
國立清華大學 NATIONAL TSING HUA UNIVERSITY

五、本授權書第一點所定各該授權對象，均應各自遵守其授權範圍及相關約定。如有違反，由該違反之行為人自行承擔一切法律責任
The authorized party of this authorized agreement shall comply with the scope of authorization and relevant agreements. In case of any violation, the person responsible for the violation shall bear all legal responsibilities."

六、本人擔保本著作為本人創作而無侵害他人著作權或其他權利。如有違反，本人願意自行承擔一切法律責任
I guarantee that this publication was created by me without infringing copyright or other rights of others. In case of any violation, I am willing to assume full legal responsibility.

七、個資利用同意條款 Consent for the Use of Personal Data：
本人同意，學校及國家圖書館為本授權書所定各授權事項目的範圍內得蒐集、處理及利用本人所提供之個人資料，學校並可將該等個人資料提供給包括國家圖書館在內之相關第三人在同一目的範圍內處理及利用
I agree that National Tsing Hua University and the National Central Library may collect, process, and use the personal data I provide within the scope of the authorized matters specified in this authorized agreement. National Tsing Hua University may also provide such personal data to relevant third parties, including the National Central Library, for processing and use within the same purpose scope.

立授權書人 Signature of the author ： 李承澤

(Date in ROC :Year/MM/DD) 2026/5/29

國立清華大學 NATIONAL TSING HUA UNIVERSITY

2026/05/22 15:06:50

國立清華大學碩士學位論文
指導教授推薦書

National Tsing Hua University Master Thesis
Advisor Approval Form

國立清華大學
NATIONAL TSING HUA UNIVERSITY

資訊系統與應用研究所
李承澤 君（學號113065525）所提之論文

動態點雲與運動向量的聯合學習技術

經由本人指導撰述，同意提付審查。

Institute of Information Systems and Applications
Mr. Lee, Cheng-Tse (Student ID: 113065525) who has submitted the Thesis/Dissertation

Joint Learning of Dynamic Point Clouds and Motion Vectors

under my guidance, I approve for the submission to the oral defense committee.

☑ 論文題目與內容符合本系(所、班、學位學程)專業領域
The subject and content of the thesis/dissertation are conformed to the professional field of the department (institute, class or program).

☑ 符合本系(所、班、學位學程)論文相似度比對標準
The thesis/dissertation meets the standard for the thesis/dissertation originality check set by the department (institute, class or program).

指導教授 (Advisor) _____ (簽章) (Signature)

中華民國 115 年 5 月 29 日
2026 / / (YYYY/MM/DD)

國立清華大學碩士學位論文
考試委員審定書
National Tsing Hua University
Final Thesis/Dissertation Review Form

國立清華大學
NATIONAL TSING HUA UNIVERSITY

資訊系統與應用研究所
李承澤 君（學號113065525）所提之論文

動態點雲與運動向量的聯合學習技術

經本委員會審查，符合碩士資格標準。

Institute of Information Systems and Applications
Mr. Lee, Cheng-Tse (Student ID: 113065525) who has submitted the Thesis/Dissertation

Joint Learning of Dynamic Point Clouds and Motion Vectors

has passed the oral defense and has met the qualifications to be awarded the degree of Master of Science.

學位考試委員會 (Oral defense Committee)

主持人 (Chair) 黃敬群 (簽章)(Signature)

委員 (Members)

中華民國 115 年 6 月 18 日
2026 / / (YYYY/MM/DD)

國立清華大學
NATIONAL TSING HUA UNIVERSITY

摘要

動態點雲能夠以三維點集合表示眞實世界中的動態場景，並已廣泛應用於體積影片系統、沉浸式通訊與自由視角觀看等應用中。然而，動態點雲的連續影格通常具有不同的點數，且缺乏明確的點對點關係，使得錯誤隱藏、時間超解析度與來源編碼等跨影格處理變得困難。 本論文研究如何從動態點雲序列中聯合學習點雲與運動向量。我們將一組影格中的第一個影格作爲關鍵影格，並透過移動關鍵影格中的點來生成後續的非關鍵影格。透過此方式，本論文能夠同時取得學習後的非關鍵影格，以及關鍵影格與非關鍵影格之間明確的點對點關係。 爲了解決此問題，本論文提出兩種聯合學習方法。第一種方法使用動態三維高斯潑濺作爲代理表示，並從最佳化後的位置變化中推導學習後的點雲與運動向量。第二種方法則使用可微分點雲渲染器，直接最佳化點雲位置，使關鍵影格中的點能夠移動到目標非關鍵影格。 本論文進一步將學習後的運動向量應用於錯誤隱藏、時間超解析度與來源編碼。實驗結果顯示，本論文提出的方法能夠產生較平滑的運動向量，並提升學習後點雲的視覺品質。相較於基線方法，我們的方法在峰值訊噪比最多可提升 17.74 分貝，並在結構相似性指標最多可提升 0.66。整體而言，本論文證明了聯合學習點雲與運動向量能夠爲動態點雲建立明確的跨影格關係，並支援後續應用。

關鍵詞:動態點雲、錯誤隱藏、超解析度、點雲編碼

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Abstract

Dynamic point clouds represent real-world dynamic scenes as sets of 3D points and have been widely used in volumetric video systems, immersive communication, and free-viewpoint viewing. However, consecutive frames in dynamic point clouds usually contain different numbers of points and lack explicit point-to-point relationships, making cross-frame processing tasks such as error concealment, temporal super-resolution, and source coding difficult. This thesis studies how to jointly learn point clouds and motion vectors from dynamic point cloud sequences. We treat the first frame in a group of frames as the key frame and generate the following non-key frames by moving the points in the key frame. In this way, this thesis can obtain both learned non-key frames and explicit point-to-point relationships between the key frame and the non-key frames. This thesis proposes two joint learning methods to address this problem. The first method uses dynamic 3D Gaussian splatting as a proxy representation and derives the learned point clouds and motion vectors from the optimized position changes. The second method uses a differentiable point cloud renderer to directly optimize point positions, allowing the points in the key frame to move toward the target non-key frames. This thesis further applies the learned motion vectors to error concealment, temporal super-resolution, and source coding. Experimental results show that the proposed methods produce smoother motion vectors and improve the visual quality of learned point clouds. Compared with baseline methods, our methods improve the peak signal-to-noise ratio by up to 17.74 dB and the structural similarity index by up to 0.66. Overall, this thesis shows that jointly learning point clouds and motion vectors can build explicit cross-frame relationships for dynamic point clouds and support downstream applications.

Keywords: Dynamic Point Cloud, Error Concealment, Super-resolution, Source Coding

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Acknowledgements

I am deeply thankful to my advisor, Professor Cheng-Hsin Hsu, for his guidance and support during my master's study. His advice helped me clarify the direction of this research and improve the quality of this thesis. I also thank Yuan-Chun Sun for the many discussions on 3D Gaussian Splatting and point cloud processing. His help was important to the completion of my conference paper. I would like to thank the members of our lab for their help and encouragement over the past two years. Their companionship made my graduate life more meaningful.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Contents

國立清華大學
NATIONAL TSING HUA UNIVERSITY

國立清華大學
NATIONAL TSING HUA UNIVERSITY

國立清華大學
NATIONAL TSING HUA UNIVERSITY

List of Figures

國立清華大學
NATIONAL TSING HUA UNIVERSITY

國立清華大學
NATIONAL TSING HUA UNIVERSITY

國立清華大學
NATIONAL TSING HUA UNIVERSITY

List of Tables

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Chapter 1 Introduction

Volumetric video has become an important representation for immersive applications, including augmented reality, virtual reality, remote collaboration, cultural heritage preservation, and autonomous driving [1]. Different from traditional 2D videos, volumetric videos support six Degrees-of-Freedom (6-DoF) viewing experiences, allowing users to observe a scene from arbitrary positions and orientations. With this property, users are not restricted to a fixed camera viewpoint, making volumetric video suitable for next-generation immersive systems. Dynamic 3D point clouds are widely used among different volumetric representations because they can represent complex 3D shapes with attributes such as positions, colors, and normals. A dynamic point cloud sequence is composed of multiple point cloud frames, and each frame is an unstructured set of points in 3D space. This representation is simple, flexible, and compatible with many existing 3D processing tools. To support its use in immersive applications, the Moving Picture Experts Group (MPEG) has developed point cloud compression standards, including G-PCC and V-PCC [2], [3], for processing, compression, streaming, and visualization of 3D point clouds across software platforms and hardware devices.

However, this simple and flexible representation also brings a major challenge. Each point cloud frame is an unordered list of points, so adjacent frames do not naturally provide explicit point-wise relationships. This is different from 2D videos, where pixels are placed on a regular grid and neighboring pixels have clearly defined spatial relationships. In point clouds, points are distributed irregularly in 3D space, and the number of points may change across frames because of occlusion, sensing noise, object motion, or changes in the captured surface. For example, the Longdress sequence in the well-known 8i dataset [4] contains frames with different numbers of points, ranging from 733,580 to 916,250. The lack of explicit point-wise relationships makes temporal processing difficult. Fig. 1.1 illustrates this issue. In a general unstructured dynamic point cloud sequence, points in adjacent frames do not have explicit point-wise relationships, so motion

國立清華大學
NATIONAL TSING HUA UNIVERSITY

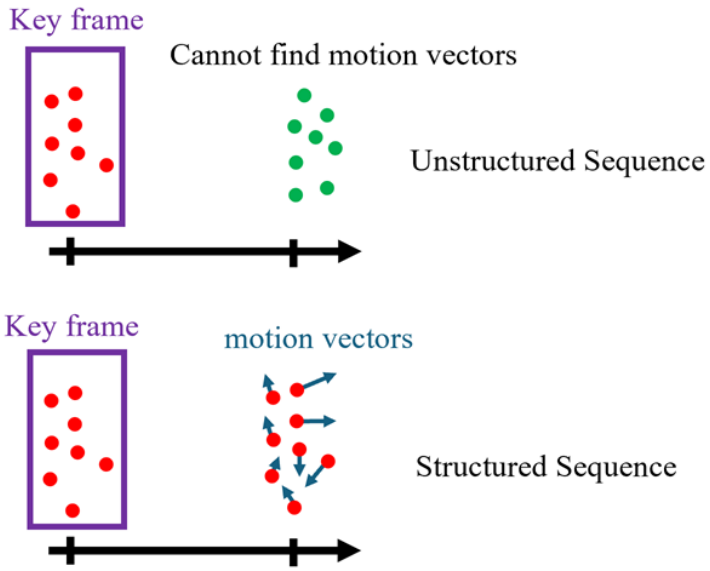


Fig. 1.1: Illustration of unstructured and structured dynamic point cloud sequences.

vectors cannot be directly found. In contrast, a structured sequence preserves point-wise relationships across frames, making motion vectors available. With reliable point-wise motion vectors, many dynamic point cloud applications become easier to realize. For example, missing frames can be reconstructed for error concealment, additional frames can be generated for temporal super-resolution, and temporal redundancy can be used for source coding.

Several prior studies have tried to estimate motion information between point cloud frames. Some of them use heuristic point-matching strategies to pair similar points across frames and approximate motion vectors [5], [6]. These methods are simple and practical, but they can become unstable under non-rigid motion, large displacement, occlusion, or large changes in point density. Other studies use neural networks to find point relationships directly in the 3D structure domain [7], [8], [9], [10], [11], [12], [13], [14]. These methods can model more complex spatial and temporal patterns, but many of them assume that adjacent frames contain the same number of points. This assumption is often invalid for real dynamic point cloud sequences. When the point numbers are different across frames, existing methods usually need voxel downsampling, k-nearest neighbors,

國立清華大學
NATIONAL TSING HUA UNIVERSITY

or subframe partitioning to make the input size consistent. These operations may remove useful geometric details or disturb the original temporal structure. As a result, estimating motion vectors directly from two independently captured and unstructured point clouds remains difficult.

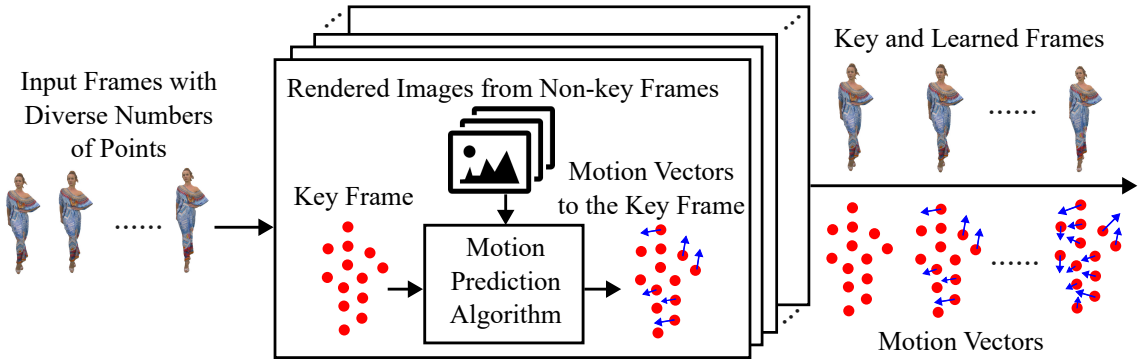


Fig. 1.2: Overview of the proposed paradigm for joint learning of point clouds and motion vectors.

To address this problem, this thesis studies a different paradigm. Fig. 1.2 gives an overview of the proposed joint learning paradigm. Instead of directly matching two independent point cloud frames, we learn non-key point cloud frames and their motion vectors together from a key frame. Given a group of consecutive dynamic point cloud frames, we use the first frame as the key frame and treat the remaining frames as non-key input frames. For each non-key input frame, we generate a learned frame whose rendered 2D views are close to those of the corresponding input frame. Because each learned frame is generated from the key frame, the point-wise motion vectors between the key frame and the learned non-key frame are explicitly defined. The main observation is that multiple point clouds can be rendered into comparable 2D views. Therefore, a non-key input point cloud does not need to be reproduced point by point, as long as the learned point cloud preserves the rendered visual appearance from representative camera views. This allows us to replace the original non-key point clouds with learned frames that are visually similar and structurally easier to process.

To solve the joint learning problem, this thesis develops several algorithms that learn non-key point cloud frames and point-wise motion vectors with different quality–complexity trade-offs. The first algorithm, Gaussian-as-a-Proxy (GaaP), uses 3D Gaus-

國立清華大學
NATIONAL TSING HUA UNIVERSITY

sian Splatting (3DGS) [15] as an intermediate representation. By leveraging dynamic 3DGS algorithms, such as Dynamic 3D Gaussians [16], GaaP derives dynamic 3D Gaussian sequences where different frames share consistent Gaussian identities. The learned Gaussians are then converted into point clouds, allowing the corresponding point-wise motion vectors to be obtained. Besides GaaP, this thesis also investigates Direct Point Cloud (DPC) algorithms, which learn point clouds directly without using Gaussians as intermediate proxies. These algorithms optimize point positions and colors through differentiable rendering with PyTorch3D [17] and Pulsar [18] renderers.

1.1 Usage Scenarios

After the joint learning process, each learned non-key frame is generated by moving the points from the key frame. Therefore, the learned frames share the same number of points and the same point order with the key frame. This gives explicit point-wise relationships between the key frame and every learned non-key frame. With these relationships, we can directly move, interpolate, or encode points across time without performing point matching again.

One usage scenario is volumetric video streaming. When a dynamic point cloud sequence is transmitted over a network, some frames may arrive late or be unavailable at the receiver. In this case, the receiver still needs to display a complete sequence to the user. With the learned motion vectors, the missing frame can be reconstructed from neighboring learned frames because the point-wise relationships are already defined. This type of frame recovery is known as error concealment.

Another usage scenario is frame-rate enhancement for volumetric video. A captured or transmitted dynamic point cloud sequence may have a limited frame rate, which can make the motion less smooth during playback. Since the learned motion vectors describe how points move across frames, they can be used to generate intermediate frames between existing frames. This allows the system to increase the frame rate of a volumetric video sequence without directly matching two independently captured point cloud frames. This task is known as temporal super-resolution.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

The learned motion vectors can also be used for source coding. Instead of treating all non-key point cloud frames as independent data, the encoder can represent them using the key frame and the learned motion vectors. Since the motion vectors usually contain smaller changes than the original point cloud values, they can reduce the amount of information required for compression. This makes the learned representation useful for storing or transmitting dynamic point cloud sequences more compactly.

1.2 Contributions

Part of this thesis is based on our earlier conference paper on joint learning of point clouds and motion vectors for volumetric video [19]. That paper introduced the initial Gaussian-as-a-Proxy learning method and evaluated its use in error concealment. This thesis extends the earlier work by reformulating the problem in a more complete thesis-level framework, introducing Direct Point Cloud learning, evaluating learned motion-vector quality, and studying additional downstream applications, including temporal super-resolution and source coding.

The main contributions of this thesis are summarized as follows:

- We study the joint learning problem of non-key point cloud frames and explicit point-wise motion vectors. Instead of directly matching independently captured unstructured point cloud frames, the proposed paradigm learns non-key frames from a key frame while preserving their rendered visual quality.
- We propose several joint learning algorithms with different design trade-offs. The Gaussian-as-a-Proxy algorithm uses dynamic 3D Gaussian Splatting as an intermediate representation, while the Direct Point Cloud algorithms learn point clouds directly using differentiable renderers.
- We evaluate the learned point clouds and motion vectors using rendered-view quality metrics and motion-vector quality metrics, including PSNR, SSIM, spatial smoothness, and temporal smoothness.
- We apply the learned explicit motion vectors to several downstream applications,

國立清華大學
NATIONAL TSING HUA UNIVERSITY

including error concealment, temporal super-resolution, and source coding. These applications show how the learned point-wise relationships can support frame recovery, frame interpolation, and compact representation of dynamic point cloud sequences.

1.3 Limitations

Although this thesis shows the potential of jointly learning point clouds and motion vectors, several limitations remain. First, the proposed framework learns point clouds mainly through rendered 2D views. Therefore, the learned frames are optimized to preserve visual appearance from representative camera views, rather than to exactly reproduce the original point-wise geometry of the input frames. This design is suitable for rendering-oriented applications, but may not fit applications that require exact geometric reconstruction. Second, the quality of the learned frames depends on the selected camera parameters and rendering settings. If the training views do not sufficiently cover the scene, some regions may receive weaker supervision. In addition, renderer-specific parameters, such as the rendering point size, can also affect the final visual quality. Third, the proposed joint learning algorithms require additional learning time. This cost is acceptable for offline processing and experimental analysis, but further optimization is needed for low-latency or real-time deployment. More experiments on longer sequences and more diverse datasets are also needed to further validate the generality of the framework.

1.4 Organization

The remainder of this thesis is organized as follows. Chapter 2 introduces the background knowledge, including volumetric video, volumetric video coding, motion vectors, dynamic point cloud matching, downstream applications, and quality assessment. Chapter 3 reviews related work on error concealment, temporal super-resolution, and source coding. Chapter 4 formulates the joint learning problem of point clouds and motion vectors, and gives an overview of the proposed solution approaches. Chapter 5 presents the Gaussian-as-a-Proxy learning method, which uses dynamic 3DGS as a proxy. Chapter 6

國立清華大學
NATIONAL TSING HUA UNIVERSITY

presents the direct point cloud learning method, which directly optimizes point positions and colors with differentiable point cloud renderers. Chapter 7 describes how the learned motion vectors are used in downstream applications. Chapter 8 presents the implementation, experimental setup, and results. Finally, Chapter 9 concludes the thesis and discusses future directions.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Chapter 2 Background

This chapter introduces the background knowledge required for understanding the proposed framework. We first review volumetric video and dynamic point clouds, which are the main data format considered in this thesis. We then introduce learning-based processing of volumetric video, volumetric video coding, motion vectors in different volumetric representations, dynamic point cloud matching, downstream applications, and quality assessment metrics. These topics are closely related to the proposed joint learning problem because our method aims to learn dynamic point clouds and their motion information at the same time. Therefore, this chapter provides the necessary context before presenting the proposed framework in later chapters.

2.1 Volumetric Video

Fig. 2.1: Volumetric Video Capture Studio [20].

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Volumetric video represents a 3D scene over time. Different from traditional 2D video, which records a scene from a fixed camera viewpoint, volumetric video allows users to observe the captured scene from different positions and viewing directions. This enables six Degrees-of-Freedom (6-DoF) viewing, where users can move their heads or change their viewpoints freely during playback. Because of this property, volumetric video is widely used in immersive applications, such as augmented reality, virtual reality, remote collaboration, cultural heritage preservation, and autonomous driving. In these applications, the user is not limited to a single pre-recorded viewpoint. Instead, the system needs to provide a consistent 3D representation that can be rendered from many possible viewpoints. Fig. 2.1 shows an example of a volumetric video capture studio, where multiple cameras and lighting devices are arranged around the capture space to record the scene from different directions. A volumetric video sequence needs to describe both the shape and appearance of a dynamic 3D scene. Several representations can be used for this purpose, including meshes, voxels, neural radiance fields, 3D Gaussian Splatting, and point clouds. These representations have different strengths in rendering quality, storage cost, editing flexibility, and processing complexity. For example, meshes provide explicit surfaces and are useful when connectivity is available, while voxels represent space in a regular grid but may require a large amount of memory at high resolution. Neural radiance fields and 3D Gaussian Splatting can produce high-quality rendered views, but they usually require a learning or optimization process. Point clouds are simple and flexible because they directly store 3D points and their attributes, but they are also unstructured and do not provide explicit connectivity between points. Therefore, the choice of representation has a direct impact on how the volumetric video is captured, compressed, transmitted, rendered, and processed.

2.2 Processing of Volumetric Video

Volumetric video stores a dynamic 3D scene over time. Unlike 2D video, each frame contains 3D geometry and color information. Common formats include point clouds, meshes, voxels, neural radiance fields, and 3D Gaussian Splatting. In this thesis, we mainly consider dynamic point clouds, where each frame is represented by 3D points

國立清華大學
NATIONAL TSING HUA UNIVERSITY

with color attributes. Many processing tasks can be applied to volumetric video, such as denoising, registration, resampling, rendering, and compression. For dynamic point clouds, temporal processing is especially important because neighboring frames are often similar. Instead of coding every frame separately, a key frame can be used to predict the following non-key frames. This prediction usually relies on motion vectors, which describe how points move across frames. Motion vectors are useful, but they also introduce extra data. When each point has its own motion vector, the motion data can become large. Directly storing these vectors may waste bits because nearby points often have related motion. Therefore, motion-vector processing is an important step in dynamic point cloud compression. A good coding scheme should organize these vectors before compression, rather than treating them as unrelated values. Rendering-based methods provide another way to process volumetric video. These methods render a 3D representation into 2D views and compare the rendered images with target images. With differentiable rendering, the image loss can be used to update the 3D representation. Neural radiance fields and 3D Gaussian Splatting are common examples. Dynamic 3D Gaussian Splatting further models scene changes over time. These methods are effective for view rendering, but they are different from the explicit motion-vector coding problem studied in this thesis. This thesis focuses on dynamic point clouds and the motion vectors used to reconstruct non-key frames. It does not replace point clouds with an implicit or Gaussian-based representation. Instead, it studies how to group, order, and encode motion vectors so that the non-key frames can be reconstructed with lower bitrate while keeping the visual quality of the volumetric video.

2.3 Volumetric Video Coding

Volumetric video usually requires a large amount of data because each frame needs to describe the geometry and appearance of a 3D scene. Compared with traditional 2D video, the data size is often much larger, since the representation may include 3D positions, colors, normals, opacities, or other attributes depending on the adopted format. In addition, a dynamic sequence contains many frames, so the total amount of data in-

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Fig. 2.2: Comparison of original and reconstructed point clouds under different coding methods [21].

creases quickly as the sequence length grows. Therefore, efficient coding is important for storing and transmitting volumetric video. A volumetric video codec needs to reduce the data size while preserving the quality of the reconstructed 3D content and its rendered views. Fig. 2.2 shows examples of original and reconstructed point clouds under different coding methods. The visual differences between the original and reconstructed results illustrate that point cloud coding may introduce geometry and appearance distortions, especially when the bitrate is limited. For example, in point-cloud-based volumetric video, MPEG has developed standard codecs such as Geometry-based Point Cloud Compression (G-PCC) and Video-based Point Cloud Compression (V-PCC) [2], [3]. G-PCC directly compresses point cloud geometry and attributes in the 3D domain, while V-PCC projects point clouds into 2D patches and reuses existing video coding tools. These two approaches reflect different ways of exploiting the structure of point cloud data. G-PCC keeps the coding process closer to the original 3D representation, whereas V-PCC benefits from mature 2D video coding techniques after projection. Similar to traditional video coding, dynamic point cloud coding can also exploit temporal redundancy across frames. If the motion between frames can be described effectively, non-key frames may be represented using changes from reference frames instead of being coded independently. This

國立清華大學
NATIONAL TSING HUA UNIVERSITY

makes motion information an important part of volumetric video coding, especially for dynamic sequences.

2.4 Motion Vectors in Volumetric Video

Motion vectors describe how visual elements or 3D positions move over time. In traditional 2D video coding, motion vectors are usually defined at the block level. A block in the current frame is predicted from a similar block in a reference frame, and the displacement between these two blocks is used as the motion vector. This idea reduces temporal redundancy, but it relies on the regular 2D grid structure of video frames. Since each pixel has a fixed location in the image grid, block matching and motion compensation can be performed in a relatively structured manner. For volumetric representations, motion vectors can be defined in different ways depending on the underlying data structure. In dynamic meshes, motion can be described by the movement of vertices when the mesh connectivity is known. In voxel-based representations, motion may be estimated for occupied cells or local regions in the 3D grid. In point-cloud-based coding, motion is often estimated at the cuboid, patch, or region level, where a group of points shares one motion vector. These group-level descriptions reduce complexity, but they may not capture detailed non-rigid motion when different points in the same region move differently. In addition to these group-level descriptions, motion can also be represented at the point level. A point-wise motion vector is defined for an individual point and describes its position and color changes over time. This representation is more flexible, but it is also more difficult to estimate because dynamic point clouds are usually unordered and do not provide explicit point correspondences across frames.

2.5 Dynamic Point Cloud Matching and Downstream Applications

Dynamic point cloud matching refers to the process of relating point cloud frames over time. Depending on the application, the relation can be built at different levels, such as points, local neighborhoods, cuboids, patches, or regions. The matched results are

國立清華大學
NATIONAL TSING HUA UNIVERSITY

often used to describe how positions and colors change across frames, or to estimate motion parameters for later processing. Unlike 2D video, point clouds do not have a regular grid structure, and the number of points may also change from frame to frame. Therefore, matching dynamic point clouds is more difficult than applying standard motion estimation in 2D video. A reliable matching result should keep the geometry and appearance aligned across time. Several applications use the motion information obtained from dynamic point cloud sequences. Error concealment reconstructs missing or late frames from available neighboring frames during streaming. Temporal super-resolution generates intermediate frames between existing frames to increase the frame rate. Source coding reduces the amount of data needed to store or transmit a sequence by using temporal similarity across frames. These applications use motion information in different ways, but they all require a description of how the dynamic point cloud changes over time. If the estimated motion is inaccurate, the reconstructed frames may contain ghosting artifacts, incorrect geometry, or unstable colors. Therefore, motion estimation is important not only for reconstruction quality, but also for compression and streaming. In this thesis, the matched motion information is treated as the input to the compression pipeline. The following sections focus on how these motion vectors are processed, organized, and encoded for dynamic point cloud compression.

(a) Reference image **(b)** Distorted image **(c)** Mask image

Fig. 2.3: Example of image quality assessment [4]. The mask is used to evaluate the foreground region and avoid the background dominating the metric values.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

2.6 Quality Assessment for Dynamic Point Clouds

Quality assessment for dynamic point clouds can be performed from different viewpoints. Rendered-view quality metrics evaluate the 2D images rendered from point cloud frames. Peak Signal-to-Noise Ratio (PSNR) [22] measures the pixel-wise difference between a reference image and a distorted image. It is usually computed from the Mean Squared Error (MSE), where a lower MSE leads to a higher PSNR value. Structural Similarity Index Measure (SSIM) [23] evaluates image quality from another perspective by comparing structural similarity, luminance, and contrast. Therefore, PSNR and SSIM are often used together because they describe different aspects of image distortion.

Fig. 2.3 shows an example of image quality assessment using MSE, PSNR, and SSIM. When the whole image is evaluated, the distorted image has MSE = 858, PSNR = 18.79 dB, and SSIM = 0.90 compared with the reference image. However, this full-image result can be misleading for rendered point cloud images. Since the foreground object only occupies part of the image, a large amount of background pixels may have little or no difference between the reference and distorted images. These background pixels reduce the overall MSE and increase the SSIM value, even when the foreground object contains visible distortion. Therefore, masked quality assessment is also important. By applying the foreground mask in Fig. 2.3(c), the metric calculation focuses on the object region instead of the whole image. In this example, the masked results are MSE = 8267, PSNR = 8.96 dB, and SSIM = 0.18. Compared with the full-image SSIM of 0.90, the masked SSIM shows that the visible foreground quality is actually poor. This example illustrates that full-image metrics may overestimate visual quality when the background dominates the rendered image. In volumetric video, users mainly observe the rendered foreground content from selected camera viewpoints. Therefore, rendered-view metrics should be interpreted carefully, and masked metrics are useful for evaluating the final viewing experience when the foreground object is the main target.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Chapter 3 Related Work

The joint learning problem addressed in this thesis has not been explicitly formulated in prior work. This chapter reviews existing studies related to three most popular downstream applications of dynamic point clouds: error concealment, temporal super-resolution, and source coding. These prior studies can be classified into two groups: (i) without and (ii) with motion vectors.

3.1 Error Concealment

Prior studies on error concealment for dynamic point cloud streaming can be divided into two groups, depending on whether motion vectors are used.

Without motion vectors. Wu et al. [24] studied how packet loss affects dynamic point cloud streaming when the V-PCC codec [3] is used. They also examined concealment strategies for different point cloud attributes under heterogeneous and dynamic network conditions. Their method did not consider motion vectors between consecutive frames.

With motion vectors. Hung et al. [5] designed three heuristic point matching methods for error concealment. The first method matches points according to position and color similarity, the second method uses triangular matching between a point and its neighboring points, and the third method performs cuboid matching with fixed-size spatial partitions. Souto et al. [25] followed a similar cuboid-based idea and matched cuboids in consecutive frames by reducing their position and color differences. Huang et al. [6] later proposed an end-to-end error concealment framework that combines motion estimation with post-processing to recover missing frames. Their framework achieved better visual quality than frame-copy and neural interpolation baselines. These works [5], [6], [25] show the usefulness of motion vectors for error concealment. However, their motion vectors are obtained through heuristic matching and are often defined at the cuboid level. In contrast, our algorithms learn explicit point-wise motion vectors, where each vector consists of position and color changes with respect to the key frame.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

3.2 Temporal Super-resolution

Without motion vectors. Rempe et al. [26] enabled temporal super-resolution by mapping dynamic point clouds into a canonical spatiotemporal space and modeling continuous shape evolution with a latent ordinary differential equation and a continuous normalizing flow. Akhtar et al. [27] first represented point cloud frames in a multi-scale feature space, and then used spatial and temporal feature correlations for temporal super-resolution. For LiDAR point clouds, He et al. [28] combined dynamic point extraction, tracking, and shape fusion with an error-state iterative Kalman filter. Sang et al. [29] used an implicit neural representation to learn a continuous velocity field for super-resolution of non-rigid and unstructured 3D point clouds. These studies [26], [27], [28], [29] performed temporal super-resolution without using motion vectors across frames.

With motion vectors. Some prior studies estimated motion vectors between frames for interpolation or extrapolation in temporal super-resolution. Viola et al. [7] designed a two-stage neural network with a point matching module and a refinement module, and minimized the L_2 distance between the learned and ground-truth point positions. Lu et al. [13] proposed a motion-based neural network that uses recurrent and attention modules to integrate motion features between consecutive frames. Lu et al. [8] further used k-nearest neighbor clusters to adaptively sample points for temporal matching. Zeng et al. [9] used deep learning to match 1024 points with an Earth Mover's Distance loss [30]. Zheng et al. [10] and Jiang et al. [11] both constructed 4D neural fields and trained them with Chamfer distance, Earth Mover's Distance, and smoothness loss in a self-supervised manner to map points across frames. Shi et al. [31] applied temporal estimation, spatial interpolation, and temporal interpolation to dynamic point clouds captured by LiDAR. Li et al. [32] decomposed the transformation into subproblems for accurate point matching across frames. Zhang et al. [14] proposed a hybrid convolution–transformer framework that jointly estimates motion and structure features with bidirectional attention for explicit scene flow. These works [7], [8], [9], [10], [11], [13], [14], [31], [32] assume a fixed number of points per frame. Therefore, each point cloud frame usually needs to be spatially downsampled or divided into multiple subframes before processing, which may

國立清華大學
NATIONAL TSING HUA UNIVERSITY

be difficult to realize and may lead to inferior quality. In contrast, our proposed paradigm supports sequences with varying point numbers and estimates explicit point-wise motion vectors.

3.3 Source Coding

Without motion vectors. A number of source coding studies have focused on static point clouds [33], [34], [35], [36]. Garcia et al. [36] designed an octree-based codec and used a context-adaptive arithmetic coder to perform intra-coding for point positions. Rudolph et al. [33] proposed a learning-based codec that encodes position and color attributes together with optimized rate control. Huang et al. [34] used generative priors to compress a single colored point cloud into sparse representations. Hu et al. [35] introduced an end-to-end scalable compression framework, where the decoded bitstreams are converted into renderable 3D primitives to balance bitrate and rendering quality. Inter-frame coding for dynamic point clouds has also been explored [3], [37], [38], [39]. V-PCC [3] projects point clouds into 2D patches so that existing 2D video codecs can be used for compression. Zeng et al. [37] followed a similar direction by converting dynamic point cloud sequences into regular 2D frames and compressing them with Versatile Video Coding (VVC) [40]. These projection-based methods [3], [37] may be affected by occlusion or topology changes over time. Zhou et al. [38] proposed a learning-based dynamic point cloud codec that models inter-frame redundancy in the latent space with spatio-temporal transformers. Akhtar et al. [39] represented point cloud frames with multi-scale features and reduced redundancy through feature-space matching. Different from these codecs [3], [36], [37], [38], [39], our proposed paradigm learns explicit point-wise motion vectors, which can help these codecs further use inter-frame redundancy.

With motion vectors. Dynamic point cloud source coding methods that explicitly use motion vectors can be grouped into structure-guided, cuboid-based, and feature-based codecs. For structure-guided codecs, Chao et al. [41] studied skeleton-based methods that segment or rig human point clouds and use affine or non-rigid transformations for motion compensation. Souto et al. [42] proposed a codec that treats small changes as having no inter-frame motion and applies motion compensation when motion needs to be handled.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Nguyen et al. [43] further introduced a learned lossy codec that uses a spatiotemporal latent context model for motion compensation.

Cuboid-based codecs form another group of methods. Mekuria et al. [44] developed a codec for tele-immersive videos, where octree-based intra coding and JPEG-based color compression are used, and Iterative Closest Point (ICP) is applied to estimate cuboid-wise rigid transforms for motion compensation. G-PCC [2] is an MPEG standard that includes cuboid-based motion compensation for dynamic point clouds. Santos et al. [45] designed an algorithm to choose between intra and inter modes, reducing bitrate without clear quality loss compared with G-PCC. An et al. [46] improved the inter-mode of G-PCC by using Hamming distance for cuboid matching, which improves compression efficiency and encoding speed. Hong et al. [47] proposed a cuboid-based inter-coding method with integer-precision motion compensation and adaptive graph filtering. Shao et al. [48] estimated patch-wise non-rigid deformations through registration and local refinement, and used affine motions of deformable patches to improve motion compensation and coding efficiency [49]. Mu et al. [50] proposed an adaptive decision method in G-PCC to adjust cuboid size, reducing inter-frame coding overhead and running time.

Feature-based codecs estimate motion in a different space. Fan et al. [51] learned feature-space motions between adjacent frames, and Xia et al. [52] extended this idea with hierarchical motions and kNN-attention cuboid matching for better coding efficiency. Jiang et al. [53] also compressed dynamic point clouds in the feature space by using inter-frame motions between latent representations. Fan et al. [54] later proposed a multi-scale inter-frame framework that compensates motions at both coarse and fine levels, while reducing redundancy in all attributes. These source coding codecs [2], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54] can be applied after our joint learning algorithms to further compress the learned point-wise motion vectors of non-key frames.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Chapter 4 Joint Learning of Point Clouds and Motion Vectors

Table 4.1: Summary of Common Notations Used in This Thesis

Symbol	Definition
N	Number of frames in a Group of Frames (GoF).
$\mathbf{X}_n = \{x_{n,1}, \ldots, x_{n,M_n}\}$	Input point cloud frame at frame n, containing M_n points; $x_{n,m}$ denotes the m-th point.
$p_{n,m}$, $c_{n,m}$	Position (p_x, p_y, p_z) and color (c_r, c_g, c_b) of point $x_{n,m}$.
$\hat{\mathbf{X}}_n = \{\hat{x}_{n,1}, \ldots, \hat{x}_{n,M_1}\}$	Learned point cloud frame at frame n; the key frame is kept unchanged, i.e., $\hat{\mathbf{X}}_1 = \mathbf{X}_1$.
$\mathbf{V}_n = \{v_{n,1}, \ldots, v_{n,M_1}\}$	Set of motion vectors from the key frame $\hat{\mathbf{X}}_1$ to the learned non-key frame $\hat{\mathbf{X}}_n$, where $v_{n,m}$ denotes the motion vector of point $\hat{x}_{n,m}$.
$\hat{\mathbf{X}}_n^*$, $\mathbf{V}_n^*$	Optimal learned non-key point cloud frame and its corresponding set of motion vectors.
Θ, θ	Set of camera parameters, with $\theta \in \Theta$ denoting one camera parameter.
$R(\mathbf{X}_n, \theta)$	Rendered 2D view of point cloud $\mathbf{X}_n$ under camera parameter θ.
i	Pixel index in a rendered 2D view.
$\mathcal{Q}(\cdot, \cdot)$	Quality degradation function combining pixel-wise and perceptual losses.

This chapter first defines the notations used in the joint learning problem and then gives an overview of the proposed joint learning algorithms. Table 4.1 summarizes the common notations used to formulate the joint learning problem.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

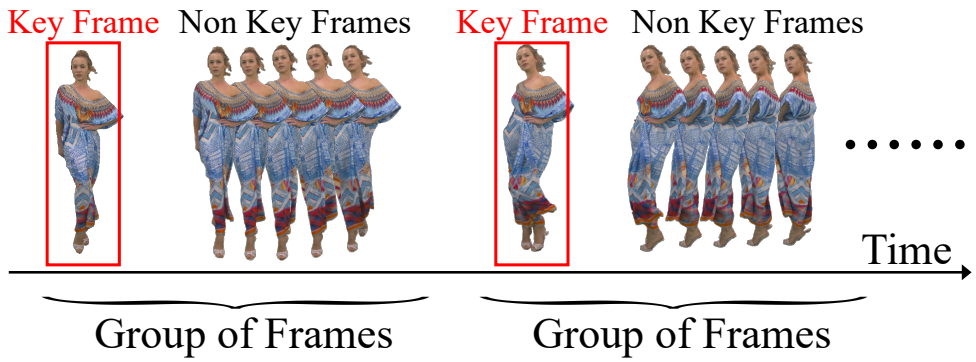


Fig. 4.1: Sample Group of Frames (GoFs).

4.1 Problem Formulation

We consider a dynamic point cloud sequence that is split into multiple Groups of Frames (GoFs). Each GoF is a fixed-length window consisting of consecutive point cloud frames. As shown in Fig. 4.1, a GoF includes one key frame and several non-key frames. More specifically, each GoF contains point cloud frames $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N\}$, where $\mathbf{X}_1$ is the key frame and $\{\mathbf{X}_2, \ldots, \mathbf{X}_N\}$ are the non-key frames. Let M_n denote the number of points in frame $\mathbf{X}_n$. The m-th point in frame n, denoted by $x_{n,m} \in \mathbf{X}_n$, has a position vector $p_{n,m} = (p^x_{n,m}, p^y_{n,m}, p^z_{n,m})$ and a color vector $c_{n,m} = (c^r_{n,m}, c^g_{n,m}, c^b_{n,m})$. Therefore, each point is represented by six dimensions in total.

Given input frames $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N\}$, we denote the non-key, learned frames as $\{\hat{\mathbf{X}}_2, \hat{\mathbf{X}}_3, \ldots, \hat{\mathbf{X}}_N\}$ and set $\hat{\mathbf{X}}_1 = \mathbf{X}_1$ for the key frame. All learned frames in $\{\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_2, \ldots, \hat{\mathbf{X}}_N\}$ contain the same number of points as $\mathbf{X}_1$, which is M_1. For the m-th point in frame $\hat{\mathbf{X}}_n$, where $m \in \{1, 2, \ldots, M_1\}$, its learned position vector is written as $\hat{p}_{n,m} = (\hat{p}^x_{n,m}, \hat{p}^y_{n,m}, \hat{p}^z_{n,m})$, and its learned color vector is written as $\hat{c}_{n,m} = (\hat{c}^r_{n,m}, \hat{c}^g_{n,m}, \hat{c}^b_{n,m})$. Since $\hat{\mathbf{X}}_n$ is obtained by shifting points in both positions and colors, the motion vector of the m-th point in non-key frame n, where $n \in \{2, 3, \ldots, N\}$ and $m \in \{1, 2, \ldots, M_1\}$, is defined with respect to the key frame $\hat{\mathbf{X}}_1 = \mathbf{X}_1$ as:

$$v_{n,m} = (\hat{p}_{n,m} - p_{1,m},\ \hat{c}_{n,m} - c_{1,m}). \tag{4.1}$$

國立清華大學
NATIONAL TSING HUA UNIVERSITY

The motion vectors of frame n are collected as:

$$\mathbf{V}_n = \{v_{n,1}, v_{n,2}, \ldots, v_{n,M_1}\}. \tag{4.2}$$

The joint learning problem also takes a set of camera parameters $\mathbf{\Theta}$ as input. For each camera parameter $\theta \in \mathbf{\Theta}$, the same renderer $R(\cdot)$ is used to render a 2D view $R(\mathbf{X}_n, \theta)$ from the non-key input frame $\mathbf{X}_n$ and another 2D view $R(\hat{\mathbf{X}}_n, \theta)$ from the corresponding learned frame $\hat{\mathbf{X}}_n$. The goal is to reduce the overall difference between $R(\mathbf{X}_n, \theta)$ and $R(\hat{\mathbf{X}}_n, \theta)$ for every $n \in \{2, 3, \ldots, N\}$ and every $\theta \in \mathbf{\Theta}$. We use $Q\big(R(\mathbf{X}_n, \theta), R(\hat{\mathbf{X}}_n, \theta)\big)$ to denote the quality degradation between two rendered views. It is defined as the weighted sum of the pixel-wise L1 loss $\mathcal{L}_1(\cdot)$ and the structural dissimilarity D-SSIM$(\cdot)$ [15]:

$$Q(R(\mathbf{X}_n, \theta), R(\hat{\mathbf{X}}_n, \theta)) = (1-\lambda)\mathcal{L}_1(R(\mathbf{X}_n, \theta), R(\hat{\mathbf{X}}_n, \theta)) + \lambda \text{D-SSIM}(R(\mathbf{X}_n, \theta), R(\hat{\mathbf{X}}_n, \theta)), \tag{4.3}$$

where λ is a hyperparameter and is set to 0.2 unless otherwise specified. The two components are defined as:

$$\mathcal{L}_1(R(\mathbf{X}_n, \theta), R(\hat{\mathbf{X}}_n, \theta)) = \frac{1}{|R(\mathbf{X}_n, \theta)|} \sum_{i \in R(\mathbf{X}_n, \theta)} |R(\mathbf{X}_n, \theta)_i, R(\hat{\mathbf{X}}_n, \theta)_i|; \tag{4.4}$$

$$\text{D-SSIM}(R(\mathbf{X}_n, \theta), R(\hat{\mathbf{X}}_n, \theta)) = 1 - \text{SSIM}(R(\mathbf{X}_n, \theta), R(\hat{\mathbf{X}}_n, \theta)), \tag{4.5}$$

where $R(\mathbf{X}_n, \theta)_i$ and $R(\hat{\mathbf{X}}_n, \theta)_i$ are the i-th rendered pixel values, and SSIM$(\cdot)$ is the 2D quality metric defined by Wang et al. [23].

Using the above notations, the joint learning problem is written as:

$$\{\hat{\mathbf{X}}_n^* \mid n = 2, 3, \ldots, N\} = \underset{\{\hat{\mathbf{X}}_n | n=2,3,\ldots,N\}}{\text{argmin}} \sum_{n=2}^{N} \sum_{\theta \in \mathbf{\Theta}} Q(R(\mathbf{X}_n, \theta), R(\hat{\mathbf{X}}_n, \theta)), \tag{4.6}$$

where $\hat{\mathbf{X}}_n^*$ denotes the optimal non-key, learned frame n. Once $\hat{\mathbf{X}}_n^*$ is obtained, the corresponding optimal motion vector can be directly derived and is written as $\mathbf{V}_n^*$. This completes the formulation of the joint learning problem.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

4.2 Solution Approaches

Optimizing Eq. (4.6) requires each non-key, learned frame $\hat{\mathbf{X}}_n$ to update its 3D point positions $\hat{p}_{n,m}$ and colors $\hat{c}_{n,m}$. The purpose is to make the rendered views $R(\hat{\mathbf{X}}_n, \theta)$ close to the rendered views of the corresponding non-key input frames $R(\mathbf{X}_n, \theta)$ for all $n \in \{2, 3, \ldots, N\}$ and $\theta \in \mathbf{\Theta}$. To achieve this goal, the renderer must provide useful sensitivity information, meaning that small changes in 3D point parameters should produce meaningful changes in the rendered 2D views. With such information, the discrepancy in Eq. (4.6) can be reduced through optimization. However, as discussed by Wang et al. [55], many rendering functions are discontinuous because of occlusion changes and visibility boundaries. As a result, their gradients with respect to 3D positions may be undefined or unstable, and thus not every renderer is suitable for solving Eq. (4.6).

To address this issue, we develop three joint learning algorithms based on different differentiable renderers. These algorithms learn non-key, learned frames and point-wise motion vectors using multiple rendered 2D views from non-key input point cloud frames. According to how differentiable rendering is used, the three algorithms are divided into two groups: (i) using existing dynamic 3DGS algorithms as proxies, and (ii) directly learning non-key point cloud frames and motion vectors. Before introducing the algorithms, we first define the notations for dynamic 3DGS sequences.

Compared with points, Gaussians have three additional attributes besides position and color: (i) opacity, which controls their contributions to rendered 2D views, (ii) scale, which specifies their influence ranges, and (iii) orientation, which determines the direction of their long axes. Thus, for each input point cloud frame $\mathbf{X}_n$, for $n \in \{1, 2, \ldots, N\}$, we generalize it into a corresponding input Gaussian frame G_n, which consists of M_n Gaussians, written as:

$$\mathbf{G}_n = \{g_{n,1}, \ldots, g_{n,M_n}\}, \tag{4.7}$$

where each Gaussian $g_{n,m}$ contains position $p_{n,m}$ and color $c_{n,m}$, together with opacity $o_{n,m}$, scale $a_{n,m}$, and orientation $q_{n,m}$. The values of these extra attributes are determined systematically. Since point cloud colors are view-independent, the colors of the proxy Gaussians are also set to be view-independent.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

We modify the dynamic 3DGS algorithm to generate non-key, learned Gaussian frames $\hat{\mathbf{G}}_n$ for $n \in \{2, 3, \ldots, N\}$ using the rendered 2D views from the input point cloud frames. Let $\hat{\mathbf{G}}_n^*$ denote the optimal non-key, learned frame n. We convert $\hat{\mathbf{G}}_n^*$ into $\hat{\mathbf{X}}_n^*$ by copying the position and color vectors, $\hat{p}_{n,m}$ and $\hat{c}_{n,m}$, for all $m \in \{1, 2, \ldots, M_n\}$, and then discarding opacity $o_{n,m}$, scale $a_{n,m}$, and orientation $q_{n,m}$. The details of these operations are introduced in the next section.

Based on this design, this thesis develops two types of algorithms for different needs. The first type is a fast Gaussian-based method that uses dynamic 3DGS [16] as a proxy. The second type directly optimizes point clouds and can provide better visual quality with longer learning time. These two methods are described in Chapters 5 and 6, respectively.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Chapter 5 Gaussian-as-a-Proxy Learning

Table 5.1: Summary of Notations for Gaussian as a Proxy (GaaP)

Symbol	Definition
$\mathbf{G}_n = \{g_{n,1}, \ldots, g_{n,M_n}\}$	Set of Gaussians at frame n.
$g_{n,m}$	A Gaussian splat with position $p_{n,m}$, color $c_{n,m}$, opacity $o_{n,m}$, scale $a_{n,m}$, and orientation $q_{n,m}$.
$\hat{\mathbf{G}}_n$	Learned Gaussian set for frame n.
A_0, Q_0, O_0	Fixed values of scale, orientation, and opacity during learning.
ℓ, s	Log-scale parameter for training and point size for rendering.
ℓ^*, s^*	Optimized log-scale and point size.
L_0, S_0	Initial log-scale and point size.
$\Delta L, \Delta S$	Step sizes for log-scale and point size.

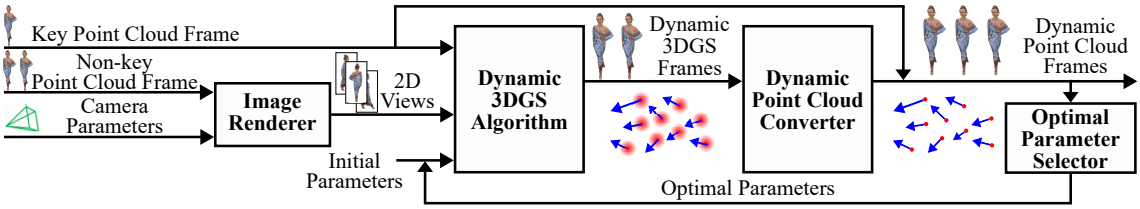


Fig. 5.1: Our proposed joint learning algorithm using 3DGS as a proxy.

This chapter presents the proposed Gaussian-as-a-Proxy (GaaP) learning method for learning non-key point cloud frames. We first describe the design of GaaP and explain how dynamic 3DGS [16] is used as a proxy for point cloud learning. We then discuss the challenges of converting learned Gaussian frames back to point cloud frames. Finally, we introduce the parameter selection process used to choose the Gaussian scale and rendering point size. Table 5.1 summarizes the additional notations used in the GaaP method.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

5.1 Design

Learning non-key point cloud frames for the objective function in Eq. (4.6) is difficult. We therefore first use recent dynamic 3DGS algorithms to build our joint learning algorithm, called GaaP.

Kerbl et al. [15] proposed 3DGS to construct a single 3DGS frame from three inputs: (i) input 2D views, (ii) the corresponding camera parameters, and (iii) an initial point cloud, which can be random or generated from SfM, Structure from Motion. The vanilla 3DGS algorithm computes gradients and iteratively updates all attributes to reduce the error between the input ground-truth views and the rendered 2D views. In this process, gradients of position attributes can be interpreted as moving points in 3D space, while gradients of color, opacity, scale, and orientation attributes can be interpreted as updating their values.

Applying vanilla 3DGS separately to individual frames often causes temporal flickering artifacts [56]. Dynamic 3DGS algorithms, such as D3DGS [16], address this issue by splitting each sequence into recurring GoFs and producing each non-key Gaussian frame by deforming the Gaussians of the corresponding key frame. This design reduces the randomness caused by independent per-frame optimization and keeps the resulting 3DGS frames with the same number of Gaussians. Based on this property, we adopt dynamic 3DGS algorithms in GaaP.

Fig. 5.1 illustrates the GaaP algorithm, which contains four components: (i) an image renderer, (ii) a dynamic 3DGS algorithm, (iii) a dynamic point cloud converter, and (iv) an optimal parameter selector. During the process, the key point cloud frame $\mathbf{X}_1$ remains fixed. For each non-key frame $\mathbf{X}_n$, the image renderer $R(\cdot)$ renders multi-view 2D observations $R(\mathbf{X}_n, \theta)$ with representative camera parameters $\theta \in \Theta$. These rendered views are then used as supervision for the dynamic 3DGS algorithm. We also initialize a key 3DGS frame $\mathbf{G}_1$ from the key point cloud $\mathbf{X}_1$. Since opacity, scale, and orientation do not exist in point clouds, these Gaussian attributes are initialized with fixed values O_0, A_0, and Q_0. In addition, 3DGS uses Spherical Harmonics (SH) coefficients to represent view-dependent colors, while point clouds use view-independent colors. We therefore set

國立清華大學
NATIONAL TSING HUA UNIVERSITY

the SH degree to 0 for view-independent colors.

Only the non-key Gaussian frames $\hat{\mathbf{G}}_\mathbf{n}$ for $n \in \{2, 3, \ldots, N\}$ are optimized afterward. With the differentiable Gaussian renderer, the selected dynamic 3DGS algorithm iteratively updates the Gaussian attributes of all non-key frames $\hat{\mathbf{G}}_\mathbf{n}$ for $n \in \{2, 3, \ldots, N\}$ to optimize Eq. (4.6). The output is a dynamic 3DGS sequence that matches the input point cloud frames in rendered views and preserves one-to-one Gaussian relationships with the key frame $\hat{\mathbf{G}}_1 = \mathbf{G}_1$.

5.2 Challenges

Two main challenges need to be addressed when a non-key, learned Gaussian frame $\hat{\mathbf{G}}_n^*$ is converted into a point cloud frame $\hat{\mathbf{X}}_n$:

1. Gaussians contain attributes that points do not have, including opacity, scale, and orientation. If these attributes are directly removed, the rendered quality may be affected.

2. The default scale A_0 used during learning and the rendering point size s used during rendering both strongly affect visual quality. Their best values are not obvious beforehand, so they need to be selected carefully.

5.3 Solutions

We handle the two challenges with the following designs:

1. We first set default values for the extra Gaussian attributes. For opacity O_0, we use its maximum value because point clouds are usually rendered without transparency. For scale A_0, we assign a small value to all Gaussians so that they behave more like points. The exact value of A_0 is determined empirically later. For orientation Q_0, we choose the setting that makes Gaussians spherical instead of elliptical, which better matches the shape of points. Unlike O_0 and Q_0, the best value of A_0 is not clear in advance, so it is selected through experiments. We also keep these

國立清華大學
NATIONAL TSING HUA UNIVERSITY

extra attributes fixed during the derivation process. In other words, we remove the loss terms related to these attributes, allowing the joint learning algorithm to focus on the spatial and temporal structures of each dynamic scene while reducing computational complexity.

2. To search over a wider range of A_0, we follow the original 3DGS paper [15] and use the scale parameter $l = \ln(A_0)$ as the control variable. The remaining task is to find the optimal l^* and s^* that produce the best visual quality in the rendered 2D views of learned point cloud frames. Since the Gaussians are trained with the representative camera parameters Θ, we use another validation set of camera parameters, denoted by Θ', to select the optimal parameters.

5.4 Optimal Parameter Selector

Our pilot tests showed that the scale parameter l affects the visual quality of rendered 2D views more strongly than the rendering point size s. Therefore, the proposed algorithm first finds the optimal scale l^* and then adjusts the point size s^*. This algorithm uses four parameters: (i) initial rendering point size S_0, (ii) initial scale L_0, (iii) rendering point size step ΔS, and (iv) scale step ΔL. The procedure is described below:

1. **Initialization.** We start from $s = S_0$ and run the dynamic 3DGS algorithm with three candidate scale values, $l \in \{L_0 - \Delta L, L_0, L_0 + \Delta L\}$. The rendered views are then evaluated using the validation camera parameters $\boldsymbol{\Theta}'$.

2. **Exploration.** When $l = L_0$ does not achieve better visual quality than both $l = L_0 \pm \Delta L$, we choose the better one from $l = L_0 - \Delta L$ and $l = L_0 + \Delta L$ and denote it by l'. A new scale is then tested at $l' + \Delta L$ if $l' = L_0 + \Delta L$, or at $l' - \Delta L$ if $l' = L_0 - \Delta L$. This step continues toward the more promising direction until a local best scale L_b is found, where L_b performs better than both $L_b \pm \Delta L$. We then let L_u be the better one from $L_b \pm \Delta L$. At this point, l^* lies in the interval $[L_b, L_u]$.

3. **Binary refinement.** We then search for l^* within $[L_b, L_u]$ using binary refinement. At iteration $k = 1, 2, \ldots, K$, the step size is set to $\Delta L / 2^{k-1}$. After K iterations,

國立清華大學
NATIONAL TSING HUA UNIVERSITY

the final value of l^* is obtained.

4. **Progressive rendering point size tuning.** After the Gaussians are derived with l^*, we tune the rendering point size by increasing it from S_0 to $S_0+\Delta S, S_0+2\Delta S, \ldots$. At each step, rendered views are generated and evaluated. The search stops when the visual quality no longer improves, and the corresponding point size is selected as s^*.

Unless otherwise specified, we set $K = 3$, $L_0 = -10$, $\Delta L = 1$, and $S_0 = \Delta S = 1$ in our experiments. Although the optimal parameter selector needs to be executed for each new dynamic point cloud sequence, it usually finishes with only a few dynamic 3DGS runs for finding l^* in steps 1–3 and a small number of 2D view renderings for selecting s^* in step 4. Therefore, the additional overhead remains manageable.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Chapter 6 Direct Point Cloud Learning

Table 6.1: Summary of Notations for Direct Point Cloud (DPC)

Symbol	Definition
β	Volume of the point cloud bounding box.
$\mathbf{F}_i$	Set of points that contribute to pixel i.
$\alpha_{n,m}$	Continuous opacity of point $x_{n,m}$.
$z_{n,m}$	Normalized depth of point $x_{n,m}$ along a ray.
$d_{n,m}$	Normalized orthogonal ray-to-center distance of point $x_{n,m}$.
$w_{n,m}$	Blending weight of point $x_{n,m}$.
γ	Depth sharpness parameter of $\hat{\mathbf{X}}_n$, where $\gamma \in [10^{-5}, 1]$.
ϵ	Background constant in the Pulsar renderer.

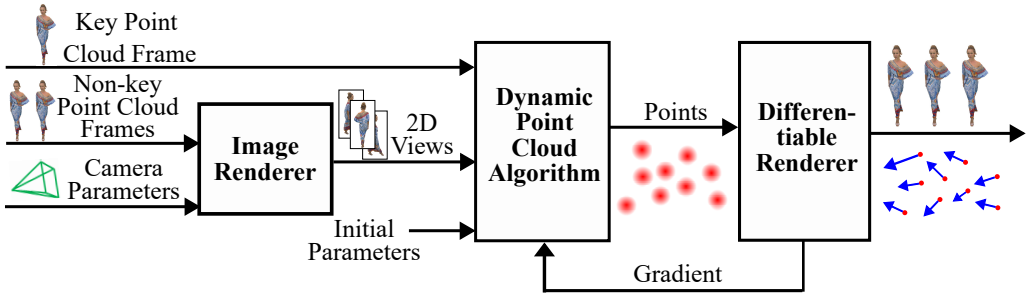
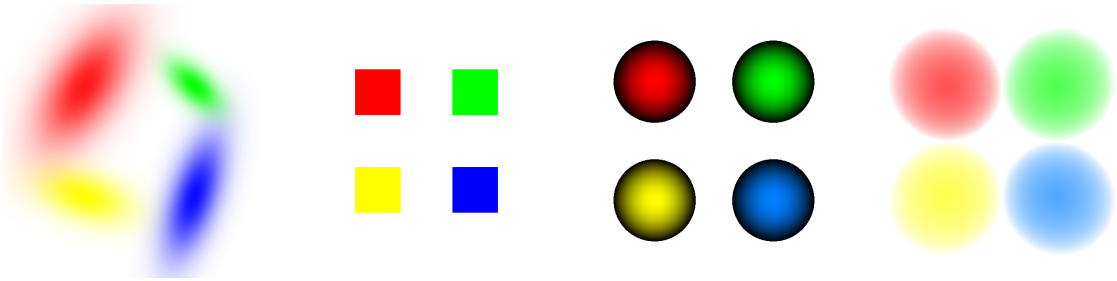


Fig. 6.1: Our proposed joint learning algorithm directly on dynamic point clouds.

This chapter presents the direct point cloud learning method. Unlike GaaP, this method does not convert point clouds into Gaussians. It directly updates point positions and colors through differentiable point cloud renderers. We first describe the main learning pipeline, and then introduce two versions based on PyTorch3D [17] and Pulsar [18]. These two versions show how different renderers affect the visual quality and learning time. Table 6.1 summarizes the additional notations used in the DPC method.

國立清華大學
NATIONAL TSING HUA UNIVERSITY



(a) 3DGS **(b)** Open3D **(c)** PyTorch3D **(d)** Pulsar

Fig. 6.2: 2D rendered views using different renderers: (a) 3DGS, (b) Open3D, (c) PyTorch3D, and (d) Pulsar.

6.1 Design

Fig. 6.1 presents the design of our joint learning algorithms that do not use Gaussians as intermediate proxies. Given a key frame $\mathbf{X}_1$, each non-key point cloud frame $\mathbf{X}_n$, where $n \in \{2, 3, \ldots, N\}$, is rendered into a 2D view $R(\mathbf{X}_n, \theta)$ with $\theta \in \mathbf{\Theta}$. We set $\hat{\mathbf{X}}_1 = \mathbf{X}_1$ and then iteratively learn each non-key frame $\hat{\mathbf{X}}_n$ for $n \in \{2, 3, \ldots, N\}$. In each iteration, the rendered view of the learned frame, $R(\hat{\mathbf{X}}_n, \theta)$, is compared with $R(\mathbf{X}_n, \theta)$ to compute the loss. The gradients are then backpropagated to update the current point positions $\hat{p}_{n,m}$ and colors $\hat{c}_{n,m}$ $\forall m \in \{1, 2, \ldots, M_1\}$. After optimization, we obtain the optimal non-key, learned frame $\hat{\mathbf{X}}^*_\mathbf{n}$, whose rendered views are visually close to those of the non-key input frame $\mathbf{X}_n$.

This joint learning process requires a renderer that can project point clouds into 2D images and support gradient backpropagation. Many point cloud renderers are available, including Open3D [57], MeshLab [58], CloudCompare [59], Blender [60], PCL Visualizer [61], PyTorch3D [17], and Pulsar [18]. However, most of them are designed mainly for visualization and are not differentiable. Fig. 6.2 shows example 2D views rendered by different renderers. The 3DGS renderer [15], shown in Fig. 6.2(a), produces smooth and continuous images, which supports stable gradients with respect to positions and colors. By contrast, the Open3D renderer [57], shown in Fig. 6.2(b), produces hard and discrete point footprints. The abrupt value changes make gradient propagation unavailable, showing that Open3D is not differentiable in this setting. Among the renderers listed above, we use two differentiable renderers for DPC: (i) PyTorch3D [17] and (ii) Pulsar [18]. Both

國立清華大學
NATIONAL TSING HUA UNIVERSITY

renderers provide Python front-end Application Programming Interfaces (APIs) and use C++/CUDA back-end kernels for efficient differentiable rendering.

Specifically, DPC$_{PT}$ adopts the PyTorch3D renderer, whose soft alpha-compositing method provides high-quality gradients, as shown in Fig. 6.2(c). On the other hand, DPC$_{PU}$ uses the Pulsar renderer, whose differentiable ray-based formulation gives smooth and stable gradients for both positions and colors, as shown in Fig. 6.2(d). The two DPC algorithms are described in the following sections.

6.2 DPC$_{PT}$ with PyTorch3D Renderer

PyTorch3D [17] supports differentiable rendering for both 3D meshes and point clouds. In DPC$_{PT}$, we use its point cloud renderer to project points onto a 2D image plane with splatting and alpha compositing. For any point $\hat{x}_{n,m}$ in a non-key, learned frame $\hat{\mathbf{X}}_n$, where $m \in \{1, 2, \ldots, M_1\}$, PyTorch3D maps $\hat{x}_{n,m}$ to the image plane with a circular footprint whose color is $\hat{c}_{n,m}$. It also computes a continuous opacity $\alpha_{n,m} \in [0, 1]$ according to the distance between the point and the camera [62]. This continuous opacity replaces a binary opacity, so the opacity changes smoothly when depth changes.

PyTorch3D then uses alpha composition to combine the color contributions from all points in a sorted list $\mathbf{F}$ for each pixel $i \in R(\hat{\mathbf{X}}_n, \theta)$. The list $\mathbf{F}$ is sorted in ascending order of the distance to the camera. The alpha-composited pixel color is written as:

$$R(\hat{\mathbf{X}}_n, \theta)_i = \sum_{m \in \mathbf{F}} \left(\alpha_{n,m} \prod_{\forall j \in \mathbf{F}_{<m}} (1 - \alpha_{n,j}) \right) \hat{c}_{n,m}. \tag{6.1}$$

where $\mathbf{F}_{<m}$ represents all points that are closer to the camera than m. This formulation is continuous and differentiable with respect to both $\hat{p}_{n,m}$ and $\hat{c}_{n,m}$, so gradient-based optimization can be applied.

Dynamic point cloud sequences may have different scales, so using a single rendering point size s for all sequences is not suitable. We address this issue by normalizing all point cloud sequences to a unified scale. For point $x_{n,m}$ from input point cloud frames, let β denote the volume of the point cloud sequence's bounding box. Both the point position

國立清華大學
NATIONAL TSING HUA UNIVERSITY

vector $p_{n,m}$ and the rendering point size s are normalized by the cubic root of this volume:

$$p_{n,m} = \frac{p_{n,m}}{\beta^{1/3}}, \qquad \text{and } s = \frac{s}{\beta^{1/3}}, \tag{6.2}$$

so that the scale is consistent across all three axes.

6.3 DPC$_{PU}$ with Pulsar Renderer

Pulsar [18] is a differentiable renderer based on spheres. It renders a scene by tracing camera rays and checking how the rays interact with point positions. Here, a sphere can be viewed as an enriched point with two extra attributes: size and opacity. For a point $\hat{x}_{n,m}$, where $m \in \{1, 2, \ldots, M_1\}$ and $n \in \{2, 3, \ldots, N\}$, we denote its sphere size by $s_{n,m}$ and its sphere opacity by $o_{n,m}$. To make the rendering process differentiable, Pulsar replaces hard boolean visibility with a soft-compositing rule weighted by depth along each ray. Let $\mathbf{F}$ denote all points that intersect a given ray. The blending weight $w_{n,m}$ of a sphere $\hat{x}_{n,m}$ on that ray is defined as:

$$w_{n,m} = \frac{\hat{o}_{n,m}\, \hat{d}_{n,m}\, \mathrm{e}^{\frac{\hat{o}_{n,m}\hat{z}_{n,m}}{\gamma}}}{\mathrm{e}^{\frac{\epsilon}{\gamma}} + \sum_{m \in \mathbf{F}} \hat{o}_{n,m}\, \hat{d}_{n,m}\, \mathrm{e}^{\frac{\hat{o}_{n,m}\hat{z}_{n,m}}{\gamma}}}, \tag{6.3}$$

where $\hat{z}_{n,m}$ is the normalized depth and $\hat{d}_{n,m}$ is the normalized distance to the view center. Here, ϵ is a small constant for the background contribution, and its default value is 10^{-7}. The parameter $\gamma \in [10^{-5}, 1]$ controls the sharpness of depth ordering, and its default value is 10^{-4}. The rendered pixel is then obtained by softly compositing the colors of all intersecting spheres:

$$R(\hat{\mathbf{X}}_n, \theta)_i = \sum_{m \in \mathbf{F}} w_{n,m} \hat{c}_{n,m}. \tag{6.4}$$

Since these weights vary continuously with the sphere position, opacity, and depth, Pulsar can provide stable gradients even when occlusion and depth change.

Each sphere in Pulsar also has two learnable attributes, namely opacity $\hat{o}_{n,m}$ and size $\hat{s}_{n,m}$. To better approximate the visual appearance of points, we fix $\hat{o}_{n,m} = 1$ and $\hat{s}_{n,m} = s$. For the selection of size s, we use the same normalization procedure as in

國立清華大學
NATIONAL TSING HUA UNIVERSITY

DPC$_{PT}$.

(a) **(b)** **(c)** **(d)** Pulsar

Fig. 6.3: Rendered views obtained using different γ settings in the Pulsar renderer: (a) default small value, (b) maximum value,(c) proposed two-phase strategy, and (d) reference frame.

Our pilot tests further show that the convergence behavior of the Pulsar renderer is highly affected by the choice of γ. Fig. 6.3 illustrates the rendered results obtained using different γ settings. When using the default small value $\gamma = 10^{-4}$ in Fig. 6.3(a), the rendered appearance becomes discrete and noisy because the gradients with respect to point positions largely vanish, making position optimization difficult. In contrast, when using the maximum value $\gamma = 1$ in Fig. 6.3(b), the rendered output becomes overly smooth and deviates from the target appearance, which negatively affects color learning. The proposed two-phase strategy shown in Fig. 6.3(c) combines the advantages of both settings and achieves a result that is visually much closer to the reference frame in Fig. 6.3(d). Therefore, we adopt the following two-phase approach to determine the optimal γ^*:

- **Phase 1: Position optimization with larger γ.** We keep colors fixed and optimize only positions. In this phase, we set $\gamma = 1$ so that stable gradients are available for position convergence.

- **Phase 2: Color refinement with smaller γ.** We keep sphere positions fixed and reduce γ to the default value 10^{-4} to train the remaining attributes and recover finer

國立清華大學
NATIONAL TSING HUA UNIVERSITY

visual details.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Chapter 7 Downstream Applications

Table 7.1: Summary of Additional Notations for Downstream Applications

Symbol	Definition
t	Number of consecutive dropped frames in error concealment.
j	Index of a missing frame to be concealed, where $j \in \{1, 2, \ldots, t\}$.
T	Number of interpolated frames generated between two adjacent frames in temporal super-resolution.

This chapter describes how the learned motion vectors are used in downstream applications. Since all learned frames are generated from the key frame, the point-wise relationships are already available after learning. These relationships allow the system to interpolate, extrapolate, or encode frames without performing point matching again. We consider three applications: error concealment, temporal super-resolution, and source coding. The symbols used in this chapter follow the definitions in Table 4.1. Additional variables introduced for downstream applications are summarized in Table 7.1.

7.1 Error Concealment

Error concealment reconstructs missing frames when some point cloud frames are lost or delayed during transmission. In this thesis, we use the explicit point-wise relationships in the learned frames to conceal missing frames by interpolation or extrapolation. This avoids the need to estimate point correspondences again at the receiver.

7.1.1 Challenge

When dynamic point clouds are transmitted over packet-switching networks, packets may arrive late or be lost, resulting in missing frames that need to be concealed at the receiver. Recovering a missing frame, such as $\mathbf{X}_n$, is not straightforward because the original motion vectors $\mathbf{V}_n$ between $\mathbf{X}_1$ and $\mathbf{X}_n$ are not directly available. Without these

國立清華大學
NATIONAL TSING HUA UNIVERSITY

motion vectors, interpolation becomes difficult.

Estimating motion vectors at the receiver from the received adjacent frames is also difficult. The number of points may differ across frames, and no predefined matching relationship exists between points in different frames. Existing works therefore used point-matching heuristics [5], [6], but these methods may introduce artifacts such as shape distortion, color biasing, or structural detail loss, reducing the visual quality of concealed point clouds. With our joint learning algorithms, motion vectors between learned point cloud frames are explicitly defined, making error concealment easier to perform.

7.1.2 Solution Approach

Consider $t \in \mathbb{Z}^+$ consecutive frames starting from $\hat{\mathbf{X}}^*_n$ are lost, while $\hat{\mathbf{X}}^*_{n-1}$ and $\hat{\mathbf{X}}^*_{n+t}$ are successfully received. We conceal the missing frames $\tilde{\mathbf{X}}^*_{n-1+j}$ by linear interpolation:

$$\tilde{\mathbf{X}}^*_{n-1+j} = \frac{t+1-j}{t+1}\,\hat{\mathbf{X}}^*_{n-1} + \frac{j}{t+1}\,\hat{\mathbf{X}}^*_{n+t}, \text{ for } j \in \{1, 2, \ldots, t\}. \tag{7.1}$$

In the extreme cases where either $\hat{\mathbf{X}}^*_{n-1}$ or $\hat{\mathbf{X}}^*_{n+t}$ is lost, we use linear extrapolation instead of interpolation.

7.2 Temporal Super-resolution

Temporal super-resolution increases the frame rate by generating intermediate frames between existing point cloud frames. Since our learned frames share explicit point-wise relationships, we can directly interpolate both point positions and colors. This makes the temporal super-resolution process simple and avoids the need for additional point matching.

7.2.1 Challenge

Temporal super-resolution aims to generate intermediate frames between existing frames and increase the frame rate. This task is difficult for point clouds because the motion can be large and non-rigid. In addition, the number of points may vary significantly

國立清華大學
NATIONAL TSING HUA UNIVERSITY

across frames, preventing direct interpolation.

Because of these issues, existing works [7], [8], [10], [11], [14], [27] often use neural networks. However, many of these methods assume a fixed number of points and require input point clouds to be downsampled to a predefined size, such as 1024 or 8192 points. Some of them also focus mainly on positions and pay less attention to colors, which may affect visual appearance. With our joint learning algorithms, motion vectors for both positions and colors are available, making temporal super-resolution more direct.

7.2.2 Solution Approach

Let $T \in \mathbb{Z}^+$ denote the number of new frames to be generated between two adjacent frames $\hat{\mathbf{X}}^*_n$ and $\hat{\mathbf{X}}^*_{n+1}$. We use linear interpolation to create additional frames $\bar{\mathbf{X}}^*_{n,t}$, where $t \in \mathbb{Z}^+$ and $t \in [1, T]$. Specifically, we write:

$$\bar{\mathbf{X}}^*_{n,t} = \frac{t}{T+1}\hat{\mathbf{X}}^*_n + (1 - \frac{t}{T+1})\hat{\mathbf{X}}^*_{n+1}, \ \forall t \in [1, T]. \tag{7.2}$$

By applying this process along the dynamic point cloud sequence, the frame rate is increased by $T + 1$ times.

7.3 Source Coding

Source coding reduces the amount of data required to represent a dynamic point cloud sequence. Instead of directly encoding every non-key input frame, we encode the key frame and the learned motion vectors.

7.3.1 Challenge

Source coding aims to encode dynamic point clouds into a compact bitstream while preserving the visual appearance of rendered views from the decoded point clouds. Some dynamic point cloud coders [2], [3], [45], [46], [47], [48], [49], [50], [63] perform motion estimation, prediction, and compensation at the cuboid level. However, applying these operations to unstructured point cloud frames can be time-consuming and may lead to

國立清華大學
NATIONAL TSING HUA UNIVERSITY

limited coding efficiency.

With our joint learning algorithms, the learned frames have explicit motion vectors. These motion vectors can reduce the entropy of the learned representation, making it easier to compress with existing source coders.

7.3.2 Solution Approach

After running our joint learning algorithms for each GoF, we obtain learned non-key frames $\hat{\mathbf{X}}_n^*$ for all $n \in \{2, 3, \ldots, N\}$, together with motion vector $\mathbf{V}_n^*$. Intuitively, $\mathbf{V}_n^*$ contains smaller values and therefore has lower entropy compared with $\hat{\mathbf{X}}_n^*$. For each GoF, we send $\mathbf{X}_1$ and $\mathbf{V}_n^*$ ($n \in \{2, 3, \ldots, N\}$) into an off-the-shelf dynamic point cloud coder for more efficient source coding. In this thesis, we evaluate this approach using bzip2 [64] for lossless compression, since the learned motion vectors are already compact.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Chapter 8 Experiments

This chapter evaluates the proposed joint learning algorithms and their downstream applications. We first describe the implementation details, datasets, parameters, and evaluation metrics. We then compare the quality and runtime of different joint learning algorithms. Finally, we evaluate how the learned motion vectors benefit error concealment, temporal super-resolution, and source coding.

8.1 Implementations

We implemented the proposed joint learning algorithms using Python and C++. For GaaP, the image renderer is built on Open3D [57]. The dynamic 3DGS part is based on D3DGS [16], with the loss function modified for the joint learning objective. The dynamic point cloud converter is also implemented with Open3D, and the optimal parameter selector is implemented in Python.

For the DPC algorithms, we rewrite the GaaP learning pipeline into a direct point cloud version. The data loaders, loss functions, and optimizers are kept close to those used in GaaP, but the differentiable renderer is replaced with a point cloud renderer. Specifically, DPC$_{PT}$ uses the PyTorch3D renderer [17], and DPC$_{PU}$ uses the Pulsar renderer [18]. Both algorithms directly update point positions and colors.

For comparison, we implement Adaptive Query Radius (AQR) proposed by Huang et al. [6] as the baseline. AQR is based on a point matching method that adjusts the search range according to position and color similarity. The matched point relationships are then used to derive non-key learned frames and motion vectors.

We also implement the three downstream applications described in Chapter 7. Error concealment and temporal super-resolution are implemented in Python. For source coding, we use bzip2 [64] for lossless compression. The source coding baseline directly compresses the input point cloud frames with bzip2, while our method compresses the

國立清華大學
NATIONAL TSING HUA UNIVERSITY

key frame and the learned motion vectors.

8.2 Setup

Dataset. Table 8.1 lists the dynamic point cloud sequences used in the experiments. We use two synthetic sequences and four real sequences. The synthetic sequences contain ground-truth motion vectors, so they are used to evaluate the quality of learned motion vectors. The real sequences are selected from the 8iVFB v2 dataset [4]. These real sequences are voxelized human point clouds and contain varying point numbers, lighting changes, and sensing noise. We focus on human subjects because they usually contain more complex motion than rigid objects. Therefore, they provide a challenging test case for evaluating the proposed methods.

Table 8.1: Considered dynamic point cloud sequences.

Sequence	Type	No. Pts. (M)	Voxelized
Man Dance (MAD)	Syn.	0.107 (const.)	No
Woman Kick (WOK)	Syn.	0.141 (const.)	No
Redandblack (RNB)	Real	0.727 ($\pm$0.05)	Yes
Loot (LOO)	Real	0.794 ($\pm$0.02)	Yes
Longdress (LON)	Real	0.834 ($\pm$0.04)	Yes
Soldier (SOL)	Real	1.075 ($\pm$0.02)	Yes

For the main evaluation, we use 72 consecutive frames from each sequence. Some experiments require repeated runs under different parameter settings, so we use 18 frames in those cases to reduce the running time. Unless otherwise specified, the GoF size is set to 6. We also evaluate GoF sizes of 3, 6, and 9 in the GoF-size analysis.

Parameters. We vary the number of rendering cameras $H \in \{25, 50, 100, 200\}$. The cameras are inward-facing and are distributed over latitudes of $0°$, $\pm 30°$, and $\pm 60°$, with equal spacing along the longitude. Unless otherwise specified, we use $H = 100$ for the main experiments.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

We also vary the rendering point size $s \in \{1, 2, 3, 4, 5, 6\}$. Since sparse and dense sequences require different point sizes, we use $s = 4$ for the synthetic sequences and $s = 2$ for the real sequences by default. For GaaP and DPC, we run the learning process for 2000 iterations. The running time is measured on a workstation with AMD EPYC 7543 CPUs at 2.80 GHz and NVIDIA RTX 3090 GPUs.

For downstream applications, we use the following settings. In error concealment, we vary the number of consecutive dropped frames among 1, 2, and 4. In temporal super-resolution, we generate 1, 2, or 4 intermediate frames between two adjacent frames. In source coding, bzip2 is used with its default parameters.

Metrics. For the learned point cloud frames, we report rendered-view quality and 3D-domain quality. Rendered-view quality is evaluated with foreground PSNR and foreground SSIM. We use foreground metrics to avoid the background dominating the quality values. The evaluation views are rendered from 24 additional cameras that are not used during learning.

For 3D-domain quality, we report PSNR-D1 for geometry quality and Y-PSNR for color quality. These metrics compare each learned point cloud frame with its corresponding input point cloud frame. Since the learning objective is based on rendered 2D views, PSNR-D1 and Y-PSNR are used as complementary metrics.

For motion-vector quality, we use spatial smoothness and temporal smoothness. Spatial smoothness measures the motion variation among nearby points, while temporal smoothness measures the motion variation across adjacent frames. Smaller values indicate smoother motion vectors.

For learning complexity, we report the per-frame learning time. The measured time includes image rendering, dynamic 3DGS or point cloud learning, differentiable rendering, and point cloud conversion. We also report the rendering time for detailed analyses.

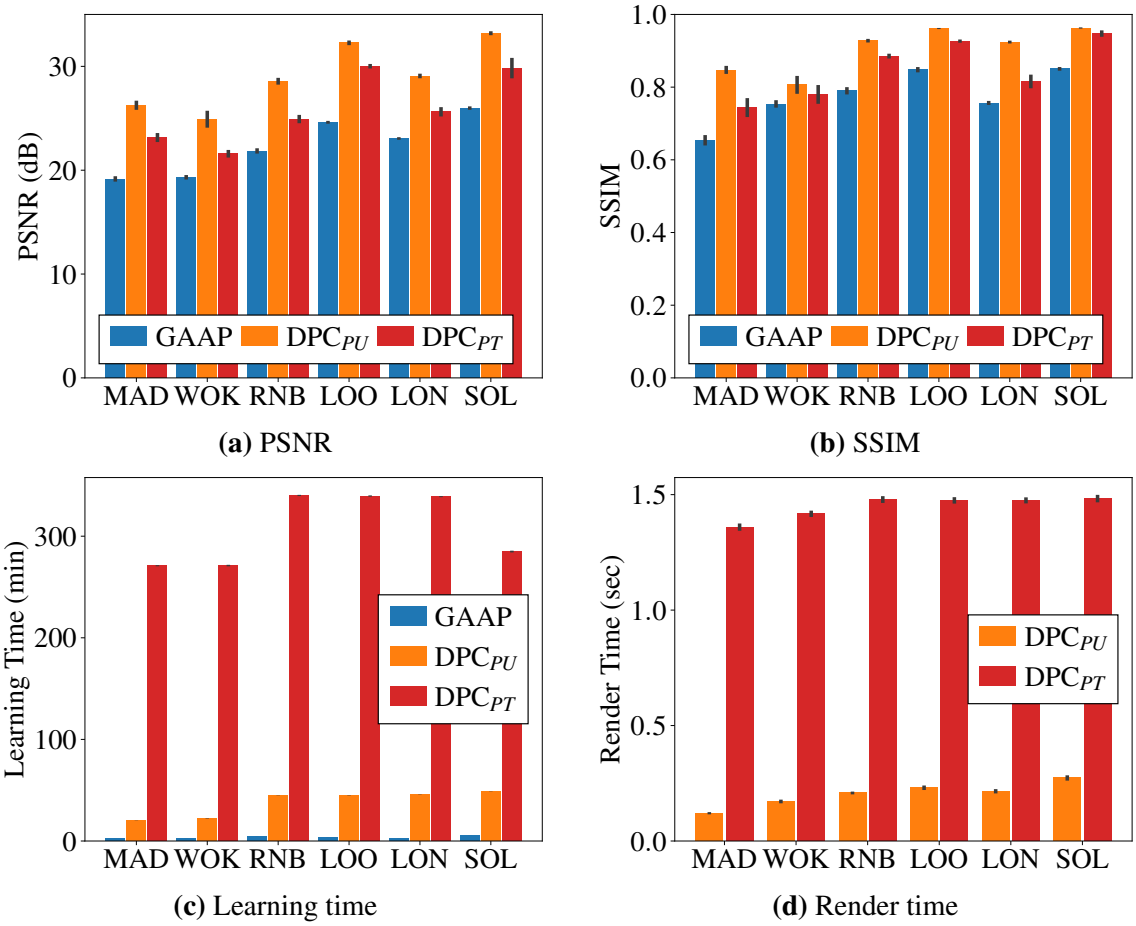
For downstream applications, we use different metrics according to the task. Error concealment is evaluated with foreground PSNR, foreground SSIM. Temporal super-resolution is evaluated with warping error [65], where a smaller value means smoother

國立清華大學
NATIONAL TSING HUA UNIVERSITY

temporal behavior. Source coding is evaluated with entropy and compressed bitstream size. Whenever possible, we report average values with 95% confidence intervals.

8.3 Results

This section evaluates the proposed joint learning algorithms and their downstream applications.



(a) PSNR **(b)** SSIM **(c)** Learning time **(d)** Render time

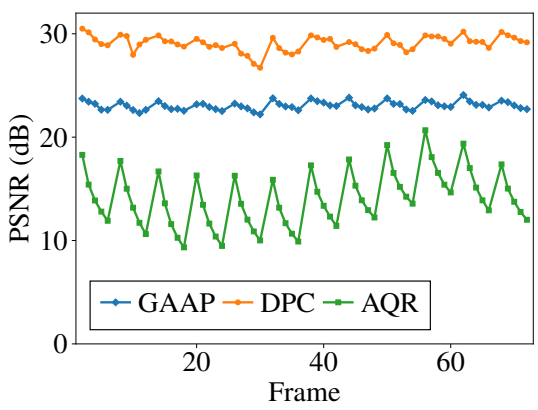
Fig. 8.1: Comparison of GaaP, DPC$_{PT}$, and DPC$_{PU}$ across evaluated sequences in: (a) PSNR, (b) SSIM, (c) learning time, and (d) render time.

Comparison among joint learning algorithms. Fig. 8.1 compares GaaP, DPC$_{PT}$, and DPC$_{PU}$ across all evaluated sequences. The goal is to examine the quality–time trade-off among the three joint learning algorithms. Figs. 8.1(a) and 8.1(b) show that DPC$_{PU}$ gives the highest rendered-view quality. Compared with GaaP, DPC$_{PU}$ improves PSNR by up

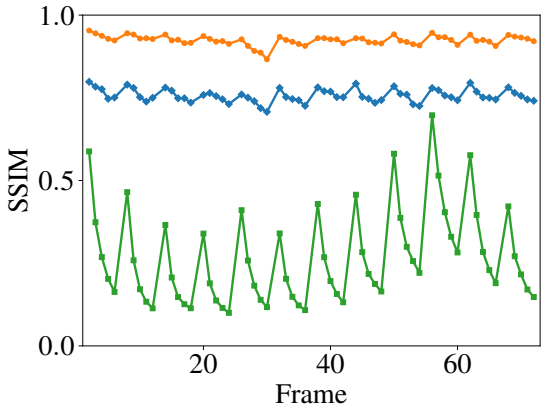
國立清華大學
NATIONAL TSING HUA UNIVERSITY

to 7.65 dB and 6.71 dB on average. It also improves SSIM by up to 0.19 and 0.13 on average. Compared with DPC$_{PT}$, DPC$_{PU}$ improves PSNR by up to 3.64 dB and 3.19 dB on average, and improves SSIM by up to 0.11 and 0.06 on average. Fig. 8.1(c) reports the learning time. GaaP has the shortest learning time among the three algorithms. On average, GaaP is 10.90$\times$ faster than DPC$_{PU}$ and 93.73$\times$ faster than DPC$_{PT}$. We further report render time in Fig. 8.1(d) to explain part of this gap. DPC$_{PU}$ renders 7.56$\times$ faster than DPC$_{PT}$ on average and up to 11.30$\times$ faster. Since rendering is repeatedly used during learning, the faster Pulsar renderer also reduces the learning time of DPC$_{PU}$. GaaP is not included in Fig. 8.1(d) because its render time is below 1 second per frame in all sequences.

國立清華大學
NATIONAL TSING HUA UNIVERSITY



(a) PSNR



(b) SSIM

(c) AQR

(d) DPC

Fig. 8.2: Sample visual quality of learned frames from longdress: (a) PSNR, (b) SSIM, (c) rendered 2D view from AQR, and (d) rendered 2D view from DPC.

Based on these results, DPC_{PU} gives the best rendered-view quality, while GaaP gives the shortest learning time. Since DPC_{PT} is worse than DPC_{PU} in both quality and runtime, we exclude DPC_{PT} from the following experiments. For simplicity, we refer to DPC_{PU} as DPC in the rest of this thesis.

Visual quality of learned frames. Fig. 8.2 shows sample results from longdress. Figs. 8.2(a) and 8.2(b) show the per-frame rendered-view quality. GaaP and DPC clearly outperform AQR, with gains of up to 20.53 dB in PSNR and 0.78 in SSIM. AQR also has larger qual-

國立清華大學
NATIONAL TSING HUA UNIVERSITY

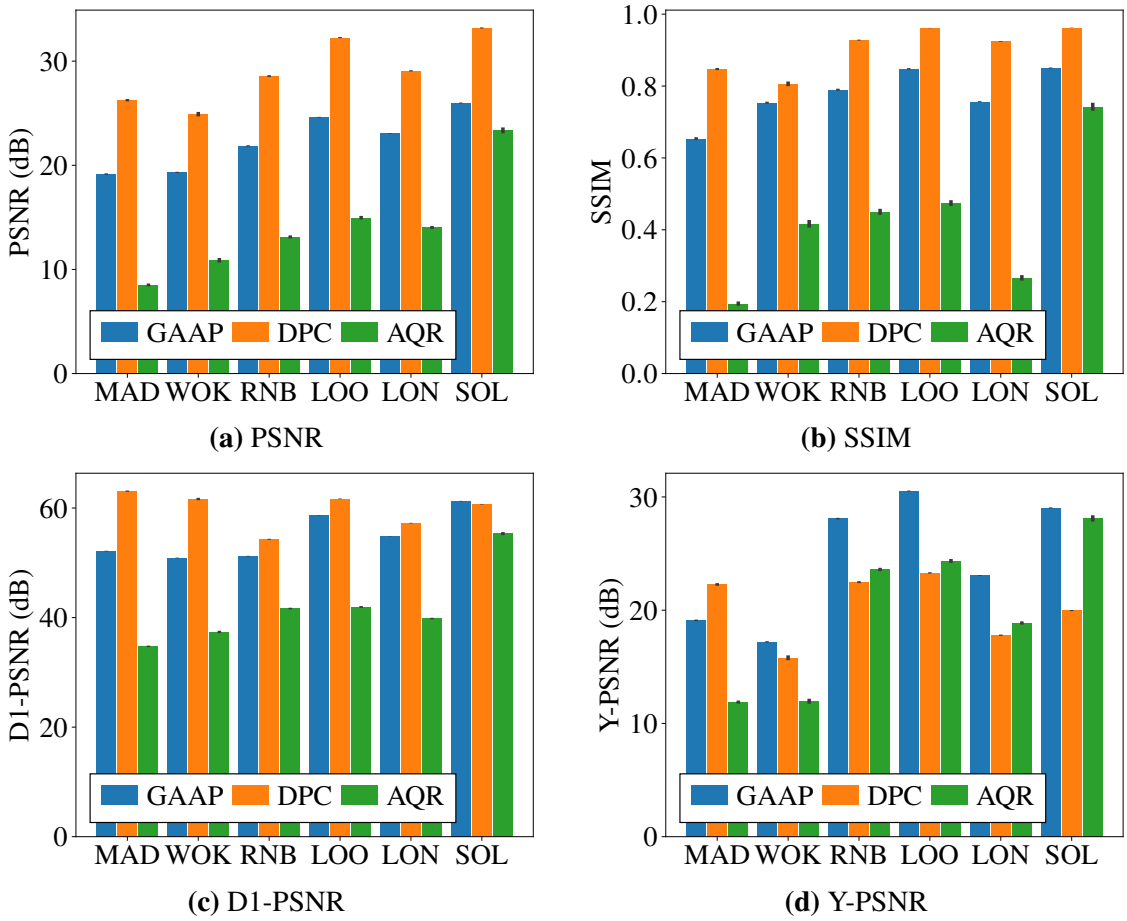


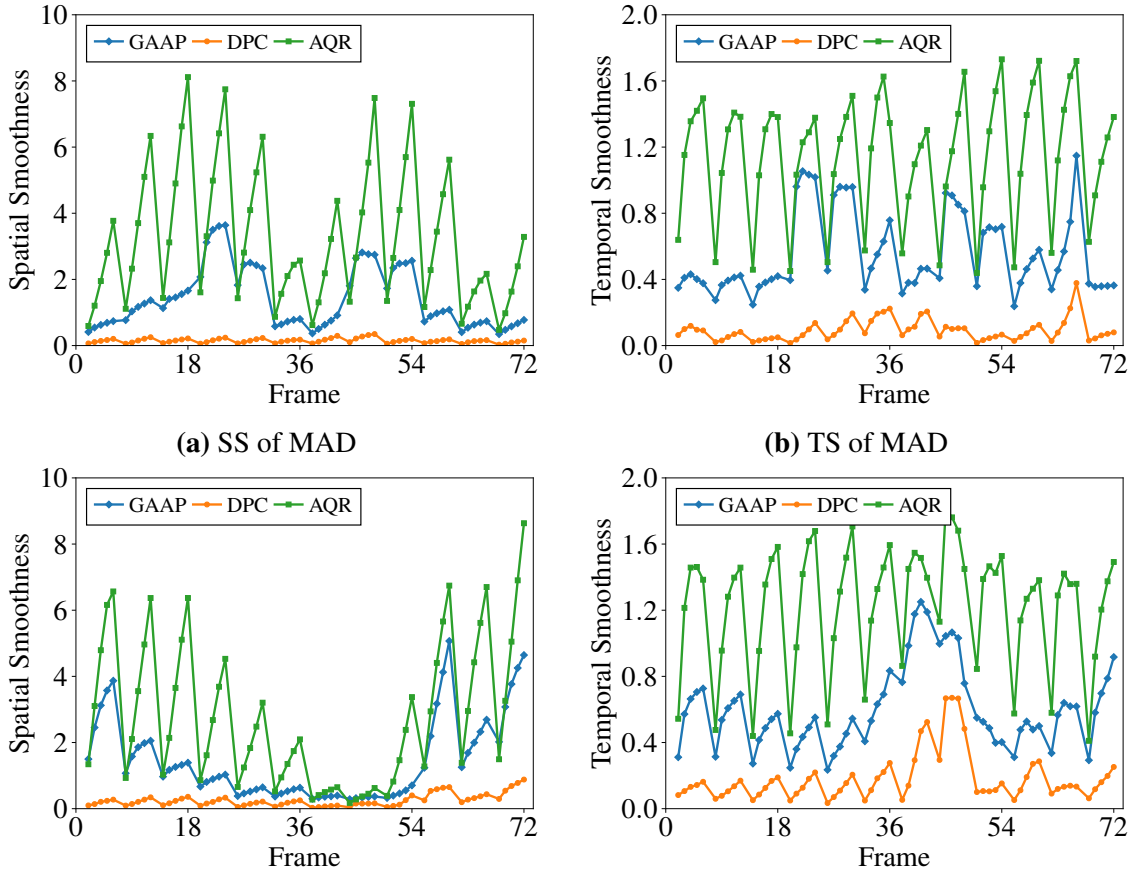
Fig. 8.3: Quality of learned frames from different joint learning algorithms in individual sequences: (a) rendered-view PSNR, (b) rendered-view SSIM, (c) 3D geometry quality measured by D1-PSNR, and (d) 3D color quality measured by Y-PSNR.

ity fluctuations. This is because its kNN-based matching becomes less reliable when the non-key frames are farther away from the key frame. This may create visible artifacts, such as holes, as shown in Fig. 8.2(c). In contrast, DPC produces a cleaner rendered view, as shown in Fig. 8.2(d).

Fig. 8.3 reports the quality of learned frames in individual sequences. We use PSNR and SSIM as rendered-view metrics, and use D1-PSNR and Y-PSNR as 3D-domain metrics. Figs. 8.3(a) and 8.3(b) show that both GaaP and DPC improve rendered-view quality over AQR. Compared with AQR, GaaP and DPC improve PSNR by up to 10.65 dB and 17.74 dB, respectively. Their average PSNR gains are 8.19 dB and 14.90 dB. For SSIM, GaaP and DPC improve over AQR by up to 0.49 and 0.66, with average gains of 0.35

國立清華大學
NATIONAL TSING HUA UNIVERSITY

and 0.48. Figs. 8.3(c) and 8.3(d) report the 3D-domain metrics. For D1-PSNR, GaaP and DPC improve over AQR by up to 17.38 dB and 28.33 dB, with average gains of 13.01 dB and 17.96 dB. For Y-PSNR, GaaP improves over AQR by up to 7.21 dB and 4.71 dB on average. DPC improves Y-PSNR by up to 10.39 dB and 0.47 dB on average, but it does not outperform AQR in every sequence. We believe this is due to the color refinement stage in DPC, which updates point colors to improve rendered-view quality rather than directly optimizing 3D-domain color similarity. Therefore, D1-PSNR and Y-PSNR are treated as complementary metrics, while rendered-view PSNR and SSIM remain the primary quality metrics.



(a) SS of MAD

(b) TS of MAD

(c) SS of WOK

(d) TS of WOK

Fig. 8.4: Motion vector quality of learned frames from two representative synthetic sequences: (a) spatial smoothness of man dance, (b) temporal smoothness of man dance, (c) spatial smoothness of woman kick, and (d) temporal smoothness of woman kick.

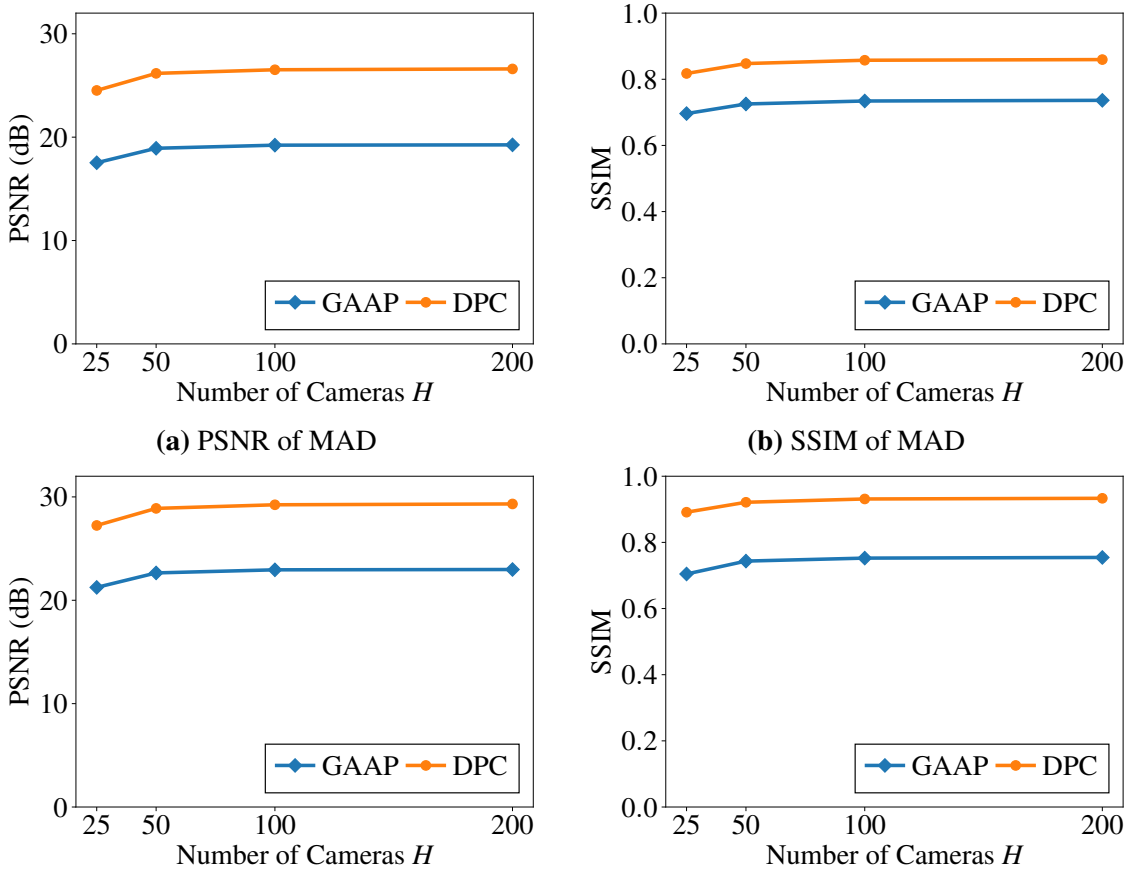
國立清華大學
NATIONAL TSING HUA UNIVERSITY

Our joint learning algorithms generate smoother motion vectors. Fig. 8.4 reports spatial smoothness and temporal smoothness for man dance and woman kick. The results show that the motion vectors learned by GaaP and DPC are smoother than those derived by AQR in both spatial and temporal domains. Compared with AQR, DPC reduces spatial smoothness and temporal smoothness by 93.35% and 88.46% on average, respectively. These two metrics are useful because they directly measure the stability of motion vectors, which cannot be captured by PSNR or SSIM alone. From Fig. 8.4, we also observe that AQR shows stronger zigzag patterns across frames. This is because AQR relies on frame-wise point matching, and its matching quality becomes unstable when the motion from the key frame becomes larger. In contrast, GaaP and DPC learn point-wise motion vectors through rendered-view optimization, so their smoothness curves are more stable across frames.

Implications of the number of rendering cameras. Fig. 8.5 shows the effect of the number of rendering cameras (H) on rendered-view quality. We use the first three GoFs of man dance and longdress for this analysis. These two sequences are selected because they represent two different data types in our evaluation: man dance is a sparse synthetic sequence, while longdress is a dense real sequence. The results show that increasing H generally improves PSNR and SSIM. However, the gain becomes small after H is larger than 50. For example, increasing H from 100 to 200 improves PSNR by at most 0.08 dB and SSIM by 0.0020 for GaaP and DPC on these two sequences. Therefore, we use at least 50 rendering cameras in our joint learning algorithms.

Implications of rendering point size. Fig. 8.6 studies how the rendering point size (s) affects visual quality. We again use the first three GoFs of man dance and longdress because they represent sparse synthetic and dense real sequences, respectively. This comparison shows that the best point size depends on point density. For the sparse man dance sequence, Figs. 8.6(a) and 8.6(b) show that both GaaP and DPC reach the best visual quality when $s = 4$. For the dense longdress sequence, Figs. 8.6(c) and 8.6(d) show that both methods perform best when $s = 2$. We omit $s = \{5, 6\}$ for longdress because they result in much lower visual quality and make the trend of smaller point sizes harder to

國立清華大學
NATIONAL TSING HUA UNIVERSITY



(a) PSNR of MAD **(b)** SSIM of MAD

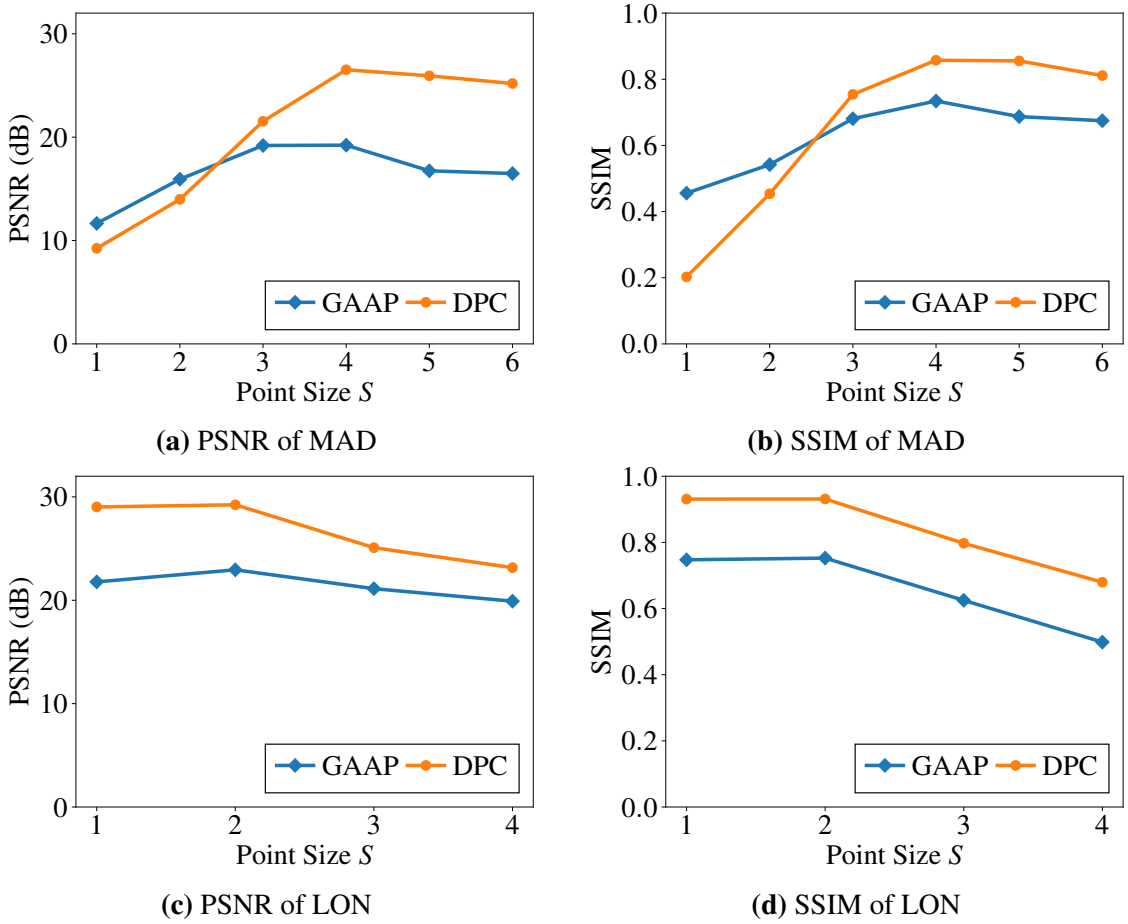
(c) PSNR of LON **(d)** SSIM of LON

Fig. 8.5: Visual quality of learned frames under different numbers of rendering cameras: (a) PSNR of man dance, (b) SSIM of man dance, (c) PSNR of longdress, and (d) SSIM of longdress.

observe. These results indicate that s should be chosen according to the point density of each sequence. The gap caused by different s can reach 17.28 dB in PSNR and 0.66 in SSIM. This effect is especially clear for sparse sequences, where a small point size may create holes in rendered views.

Implications of GoF size. Fig. 8.7 compares different GoF sizes over the first 18 frames. We use this shorter range because this analysis requires repeated runs under different GoF settings. Across all evaluated sequences and methods, GoF=3 gives higher PSNR than GoF=6 by 0.57 dB on average, while GoF=6 gives higher PSNR than GoF=9 by 0.29 dB on average. For SSIM, GoF=3 improves over GoF=6 by 0.02 on average, and GoF=6 improves over GoF=9 by 0.01 on average. This trend is expected because a smaller

國立清華大學
NATIONAL TSING HUA UNIVERSITY



(a) PSNR of MAD

(b) SSIM of MAD

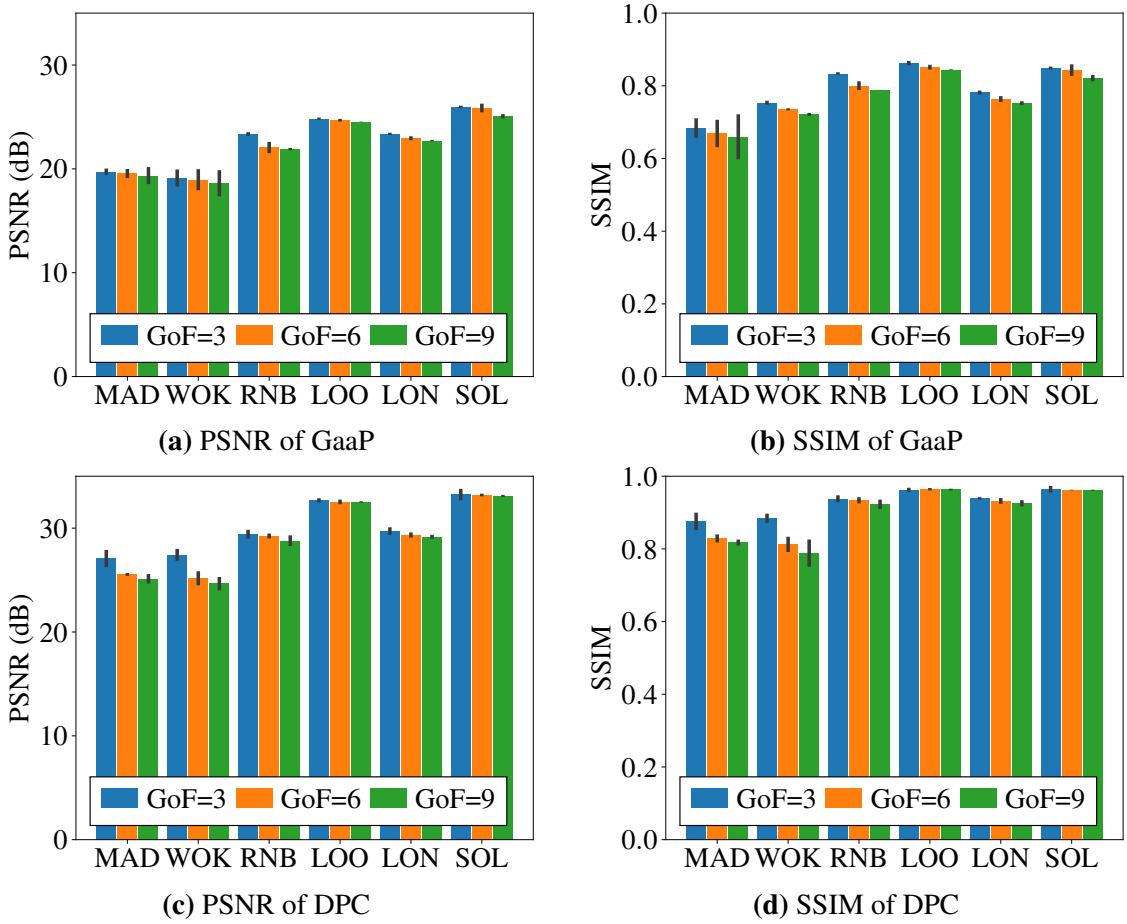
(c) PSNR of LON

(d) SSIM of LON

Fig. 8.6: Visual quality of learned frames under different rendering point sizes: (a) PSNR of man dance, (b) SSIM of man dance, (c) PSNR of longdress, and (d) SSIM of longdress.

GoF keeps each non-key frame closer to its key frame, reducing the motion that must be learned. However, a smaller GoF also creates more key frames. This is less desirable for downstream applications because fewer frames can be represented as learned non-key frames with explicit motion vectors. Therefore, even though GoF=3 gives the best visual quality, we use GoF=6 as the default setting because it provides a better balance between visual quality and the number of learned non-key frames.

國立清華大學
NATIONAL TSING HUA UNIVERSITY



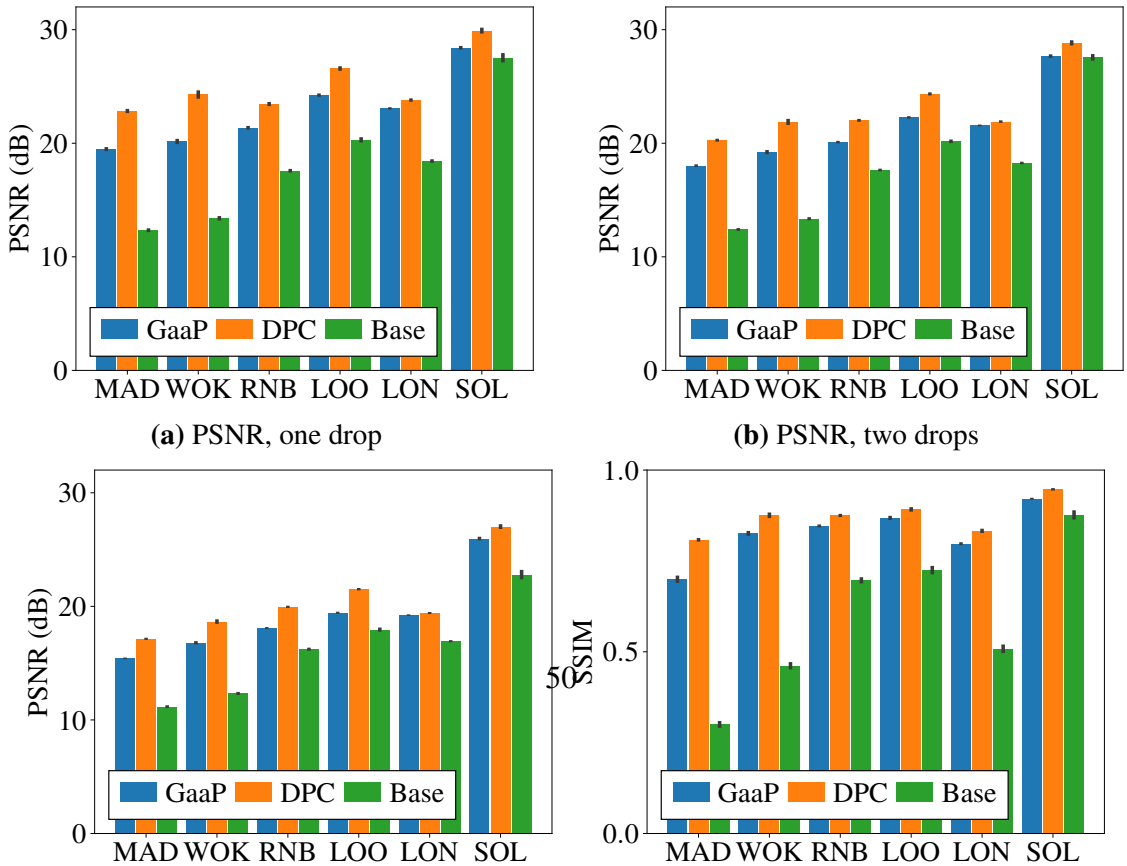
(a) PSNR of GaaP

(b) SSIM of GaaP

(c) PSNR of DPC

(d) SSIM of DPC

Fig. 8.7: GoF-size analysis over the first 18 frames: visual quality of learned frames under GoF=3, GoF=6, and GoF=9 for (a) PSNR of GaaP, (b) SSIM of GaaP, (c) PSNR of DPC, and (d) SSIM of DPC.



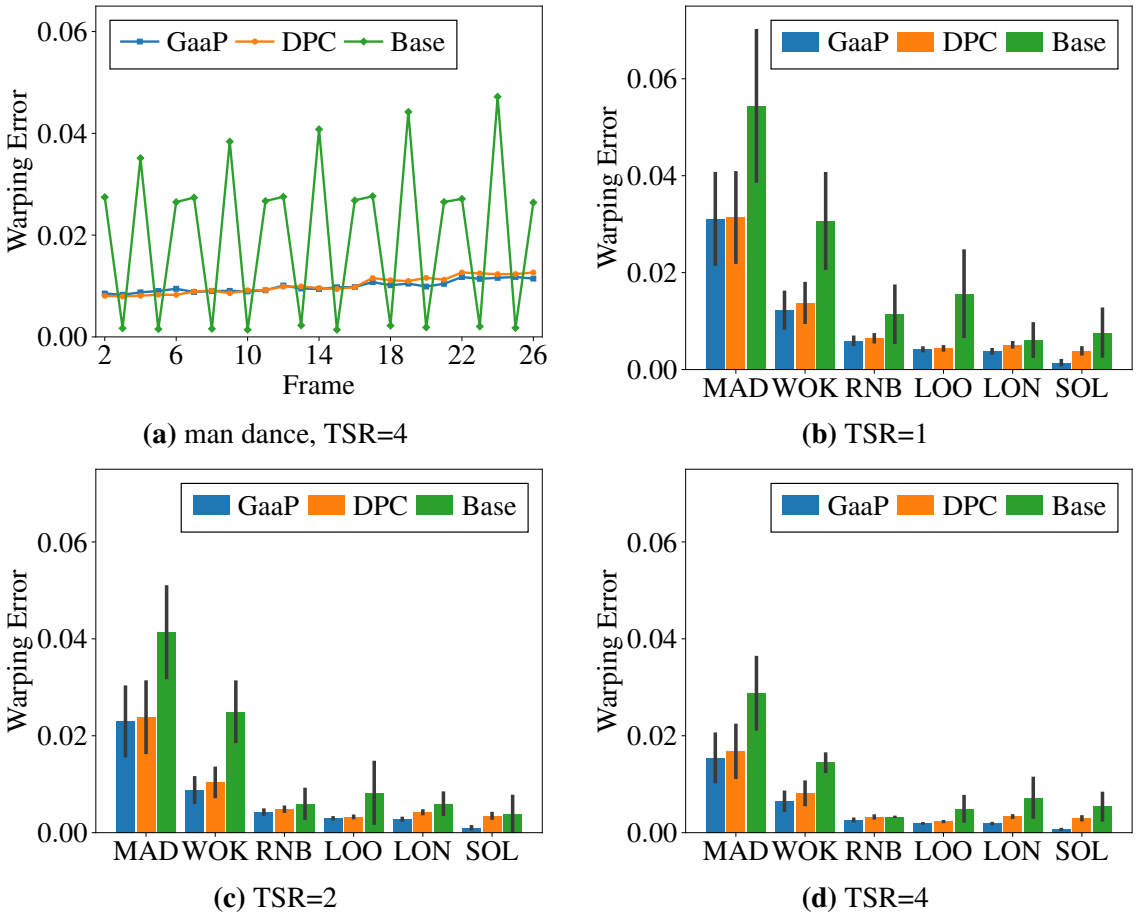
(a) PSNR, one drop

(b) PSNR, two drops

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Error concealment with learned motion vectors. Fig. 8.8 compares the quality of concealed frames under different numbers of consecutive frame drops. The baseline is the high-quality variant of the pipeline in Huang et al. [6]. Across the evaluated sequences, using the motion vectors learned by GaaP and DPC leads to higher concealed-frame quality than the baseline. For the single-frame-drop case, GaaP and DPC increase PSNR by up to 7.14 dB and 10.90 dB, respectively. They also increase SSIM by up to 0.40 and 0.51, respectively. The same trend can be observed when two or four consecutive frames are dropped. For two-frame drops, GaaP and DPC improve PSNR by 3.24 dB and 4.97 dB on average, while improving SSIM by 0.19 and 0.21 on average. For four-frame drops, the average PSNR gains are 2.92 dB and 4.38 dB, and the average SSIM gains are 0.17 and 0.18, respectively. In addition, our concealment process is much faster than the baseline pipeline. For example, concealing one missing frame takes about 30 ms with our motion vectors, while the baseline pipeline takes nearly one minute because of its point-matching process.

國立清華大學
NATIONAL TSING HUA UNIVERSITY



(a) man dance, TSR=4 **(b)** TSR=1

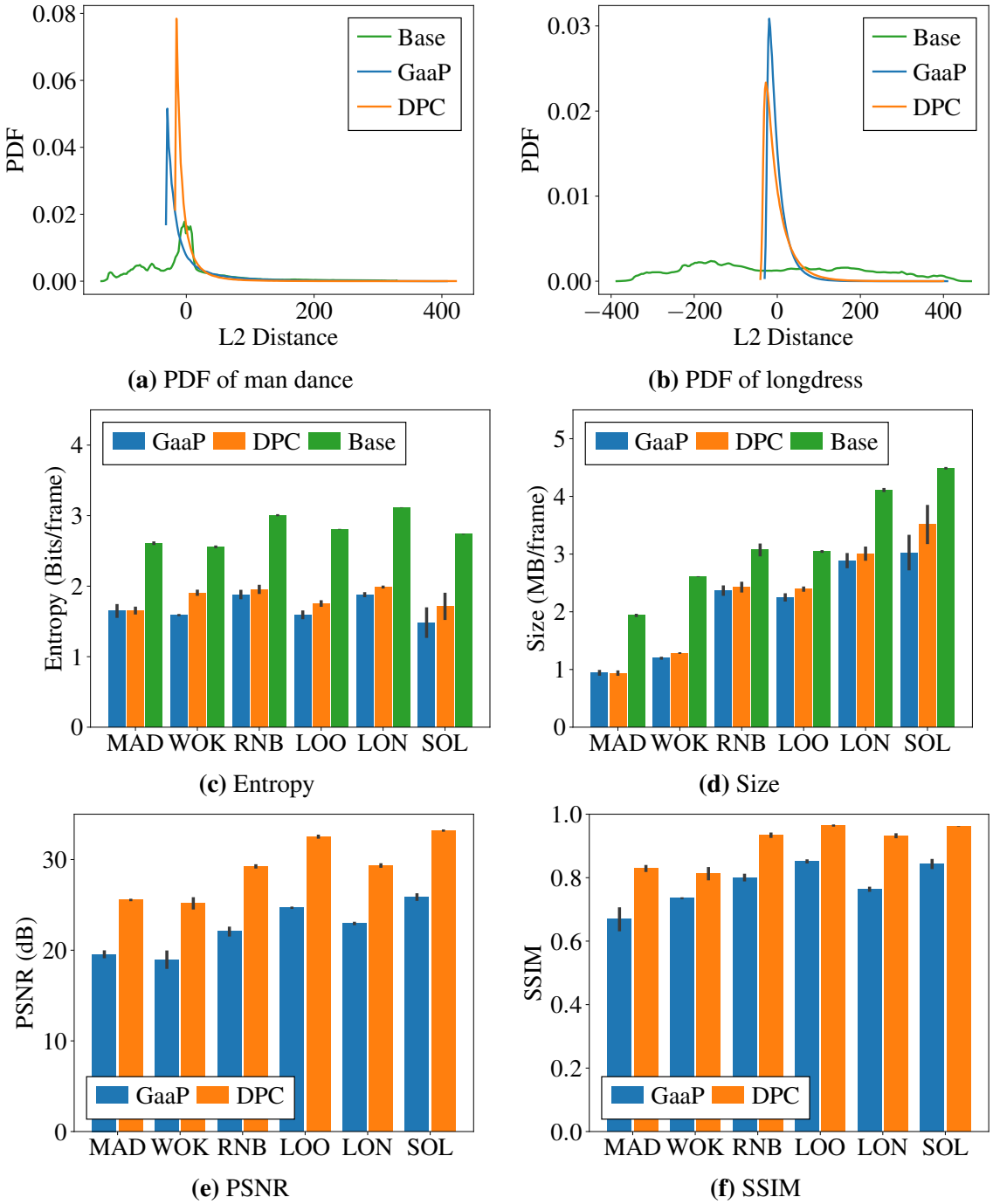
(c) TSR=2 **(d)** TSR=4

Fig. 8.9: Warping errors after temporal super-resolution: (a) per-frame results from the first GoF of man dance with four interpolated frames between adjacent input frames, and overall results with (b) one, (c) two, and (d) four interpolated frames.

Temporal super-resolution with learned motion vectors. Fig. 8.9 reports warping error [65] after temporal super-resolution. This metric measures temporal jitter and motion discontinuity in rendered views. Fig. 8.9(a) shows a per-frame example from the first GoF of man dance, where four frames are interpolated between adjacent input frames. The key frame is not plotted because its warping error is zero by definition. The baseline, which is based on the high-quality variant of the pipeline in Huang et al. [6], shows large fluctuations in this example. In comparison, GaaP and DPC produce smoother warping-error curves. Figs. 8.9(b)–8.9(d) further show the per-sequence results with one, two, and four interpolated frames. Compared with the baseline, GaaP reduces the average warping error by 57.28%, 54.70% and 57.18%, while DPC reduces by 46.42%, 36.27% and

國立清華大學
NATIONAL TSING HUA UNIVERSITY

39.65%. The increase mainly occurs in more difficult cases with four interpolated frames.



(a) PDF of man dance

(b) PDF of longdress

(c) Entropy

(d) Size

(e) PSNR

(f) SSIM

Fig. 8.10: Source coding evaluation using learned motion vectors: shifted L2-distance distributions from (a) man dance and (b) longdress, per-frame (c) entropy, (d) bzip2-compressed bitstream size, (e) rendered-view PSNR, and (f) rendered-view SSIM.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Source coding with learned motion vectors. Fig. 8.10 evaluates whether the learned motion vectors are suitable for source coding. Figs. 8.10(a) and 8.10(b) show the shifted L2-distance distributions for man dance and longdress. Compared with the original point cloud values, the learned motion vectors from GaaP and DPC are more concentrated around zero. This indicates that the learned motion vectors contain less variation and are easier to encode. This trend is also supported by the entropy results in Fig. 8.10(c). Compared with directly encoding the original input point clouds, GaaP and DPC reduce entropy by up to 45.81% and 37.54%, respectively. As shown in Fig. 8.10(d), this reduction leads to smaller bitstream sizes after lossless bzip2 compression. The largest bitstream size reductions are 54.18% for GaaP and 51.62% for DPC. Finally, Figs. 8.10(e) and 8.10(f) show that the compressed representation still keeps high rendered-view quality, reaching up to 33.19 dB in PSNR and 0.96 in SSIM.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Chapter 9 Conclusion

This chapter concludes the thesis and discusses possible future research directions. The first section summarizes the main findings of this thesis, including the proposed joint learning problem, the two types of learning algorithms, the learned motion-vector quality, the learned point cloud quality, and the performance in downstream applications. The second section discusses remaining limitations and outlines several future research problems.

9.1 Concluding Remarks

This thesis studied the joint learning problem of dynamic point clouds and motion vectors. Instead of directly matching two independently captured point cloud frames, this thesis learns non-key frames from a key frame. Since each learned non-key frame is generated from the key frame, the point-wise motion vectors can be directly obtained. This thesis proposed two types of joint learning algorithms. The first type is Gaussian-as-a-Proxy (GaaP), which uses dynamic 3DGS as a proxy to learn non-key frames and derive motion vectors. The second type is Direct Point Cloud (DPC) learning, which directly updates point positions and colors with differentiable point cloud renderers. GaaP has shorter learning time, while DPC can provide better rendered visual quality.

The main findings of this thesis are summarized as follows:

- **Joint learning of point clouds and motion vectors.** This thesis formulated the joint learning problem for dynamic point clouds. Instead of first estimating point relationships between two unstructured frames and then deriving motion vectors, the proposed method learns non-key frames from a key frame. This gives each learned non-key frame the same point order as the key frame. As a result, point-wise motion vectors can be obtained directly after learning.

- **Gaussian-as-a-Proxy and Direct Point Cloud learning.** This thesis designed two

國立清華大學
NATIONAL TSING HUA UNIVERSITY

types of learning algorithms for the same joint learning problem. GaaP uses dynamic 3DGS as a proxy and converts the learned Gaussian frames back to point cloud frames. DPC removes the Gaussian proxy and directly optimizes point positions and colors. These two methods provide different choices between learning time and rendered visual quality. DPC gives higher rendered-view quality, improving PSNR over GaaP by 6.71 dB on average and up to 7.65 dB. It also improves SSIM over GaaP by 0.13 on average and up to 0.19. On the other hand, GaaP has the shortest learning time, being 10.90× faster than DPC on average.

- **Motion-vector quality.** By learning non-key frames from the key frame, the proposed algorithms avoid heuristic point matching between independently captured frames. The learned motion vectors are smoother in both the spatial and temporal domains. Compared with AQR, DPC reduces spatial smoothness and temporal smoothness by 93.35% and 88.46% on average, respectively. This shows that the learned point-wise relationships are more suitable for describing motion in dynamic point cloud sequences.

- **Quality of learned point clouds.** The proposed algorithms learn non-key point cloud frames by matching rendered 2D views of the input frames. This makes the learned frames preserve the visual appearance of the original dynamic point cloud sequence. Compared with AQR, GaaP and DPC improve foreground PSNR by up to 10.65 dB and 17.74 dB, respectively. Their average PSNR gains are 8.19 dB and 14.90 dB. For foreground SSIM, GaaP and DPC improve over AQR by up to 0.49 and 0.66, with average gains of 0.35 and 0.48. These results show that the learned frames can provide better rendered visual quality while keeping explicit motion vectors.

- **Downstream applications.** This thesis applied the learned motion vectors to three downstream applications: error concealment, temporal super-resolution, and source coding. For error concealment, the learned point-wise relationships allow missing frames to be reconstructed by interpolation or extrapolation without running point matching again. Compared with the baseline, GaaP and DPC improve PSNR by up to 7.14 dB and 10.90 dB for the single-frame-drop case, respectively. They also

國立清華大學
NATIONAL TSING HUA UNIVERSITY

improve SSIM by up to 0.40 and 0.51, respectively. For temporal super-resolution, the same point order across learned frames allows intermediate frames to be generated by directly interpolating point positions and colors. The learned motion vectors reduce temporal discontinuity in rendered views, and the warping error can be reduced by up to 77.61% compared with the baseline. For source coding, the key frame and learned motion vectors are encoded instead of all non-key input point clouds, since many motion-vector values are close to zero and are easier to compress. Compared with directly encoding the original input point clouds, GaaP and DPC reduce entropy by up to 45.81% and 37.54%, respectively. The largest compressed-size reductions are 54.18% for GaaP and 51.62% for DPC. The compressed representation still keeps high rendered-view quality, reaching up to 33.19 dB in PSNR and 0.96 in SSIM.

These results show that the proposed joint learning method is useful not only for learning non-key point cloud frames, but also for producing motion vectors that can support later processing. The same learned motion information can be used for error concealment, temporal super-resolution, and source coding. Among the two proposed learning methods, GaaP is more suitable when learning time is the main concern, while DPC is more suitable when rendered visual quality is more important. Overall, this thesis shows that learning dynamic point clouds and motion vectors together is a practical way to create point-wise relationships for dynamic point cloud sequences.

9.2 Future Directions

Several limitations remain in this thesis. Instead of listing them only as general future work, this section formulates them as concrete future research problems. Each problem can be studied as an independent research direction for improving joint learning of dynamic point clouds and motion vectors.

Future Research Problem 1 (Outlier-Aware Joint Learning)**.**

Given a key point cloud frame, a set of learned non-key frames, and a set of training cameras, how can we remove points that contribute little to rendered-view quality while

國立清華大學
NATIONAL TSING HUA UNIVERSITY

preserving the visual quality of the learned sequence?

This problem comes from the observation that rendered-view optimization may keep points that are not important for the final rendered images. Some of these points may be outliers or redundant points. They may have little influence on PSNR or SSIM, but they still increase the number of points, the motion-vector size, and the compression cost.

A possible research direction is to design an importance score for each learned point. The score may be based on visibility, rendering contribution, accumulated opacity, or the change in rendered loss after removing the point. Based on this score, a pruning algorithm can remove low-importance points after learning. Another direction is to add a sparsity or pruning term during learning, so that unimportant points are gradually removed while the learned frames are optimized.

The main challenge is to remove points without creating holes or visible artifacts in rendered views. A successful solution should reduce the number of points and the compressed bitstream size while keeping rendered-view PSNR and SSIM close to the unpruned learned sequence.

Future Research Problem 2 (Occlusion-Aware Joint Learning under Strong Motion)**.**

Given a GoF with large non-rigid motion, occlusion, or newly visible surfaces, how can a joint learning algorithm generate learned non-key frames when some target regions cannot be represented well by moving only the key-frame points?

This problem addresses a key limitation of the current formulation. This thesis learns each non-key frame by moving points from the key frame. This works well when the key frame contains most of the visible surfaces in the following non-key frames. However, when strong motion, self-occlusion, disocclusion, or topology change occurs, some regions in a non-key frame may not have good corresponding points in the key frame.

A possible research direction is to make the GoF structure adaptive. For example, the system can shorten the GoF when motion becomes large, or choose a new key frame when the current key frame no longer covers the visible content. Another direction is to allow controlled point generation for newly visible regions. In this case, the method

國立清華大學
NATIONAL TSING HUA UNIVERSITY

must decide when a new point should be added, where it should be initialized, and how its motion vector should be represented.

The main challenge is to support newly visible content without losing the advantage of explicit point-wise relationships. A successful solution should improve rendered-view quality in difficult motion regions while keeping the learned sequence structured enough for error concealment, temporal super-resolution, and source coding.

Future Research Problem 3 (Dynamic 3DGS Proxy Selection for Joint Learning)**.**

Given multiple dynamic 3DGS algorithms, how can we determine which proxy is most suitable for learning point clouds and motion vectors under different dynamic point cloud sequences?

This problem extends the GaaP direction. GaaP uses dynamic 3DGS as a proxy because dynamic 3DGS can maintain temporally related Gaussian representations. However, different dynamic 3DGS methods may have different assumptions, deformation models, training costs, and temporal consistency properties. These differences may affect the learned point clouds, the motion vectors, and the final downstream application quality.

A possible research direction is to build a proxy evaluation framework. The framework should compare different dynamic 3DGS proxies under the same point cloud inputs, camera settings, and conversion procedure. The comparison should include rendered-view quality, learning time, motion-vector smoothness, point cloud conversion quality, and downstream performance.

The main challenge is that a proxy with high 3DGS rendering quality may not always produce the best point cloud or motion vectors after conversion. A successful solution should identify what proxy properties are important for joint learning and provide guidelines for choosing or designing better dynamic 3DGS proxies.

Future Research Problem 4 (Efficient Direct Point Cloud Learning)**.**

Given a direct point cloud learning algorithm with high rendered-view quality but

國立清華大學
NATIONAL TSING HUA UNIVERSITY

long learning time, how can we reduce the learning cost while preserving point-wise motion-vector quality?

This problem focuses on the practical cost of DPC. DPC directly optimizes point positions and colors with differentiable point cloud renderers. This design avoids the Gaussian proxy and can provide high rendered-view quality, but it also requires many rendering and optimization iterations. This cost limits the use of DPC on long sequences or large-scale datasets.

A possible research direction is to reduce the optimization space. For example, points can be divided into local regions, and each region can be optimized with shared or hierarchical motion parameters before point-level refinement. Another direction is to use multi-resolution learning, where the algorithm first learns a coarse motion field and then refines only difficult regions. Faster differentiable point cloud renderers or cached visibility information may also reduce the repeated rendering cost.

The main challenge is to reduce learning time without turning point-wise motion into overly coarse region-wise motion. A successful solution should lower per-frame learning time while maintaining rendered-view PSNR, SSIM, and motion-vector smoothness close to the original DPC result.

Future Research Problem 5 (Application-Aware Motion Vector Learning and Coding)**.**

Given that different downstream applications use motion vectors in different ways, how can we learn motion vectors that are optimized for a target application instead of using the same learned motion vectors for all applications?

This problem comes from the fact that error concealment, temporal super-resolution, and source coding prefer different motion-vector properties. Error concealment needs motion vectors that remain reliable when frames are missing. Temporal super-resolution needs temporally smooth motion to avoid jitter. Source coding needs motion vectors that are compact, low-entropy, and easy to encode. A single learning objective may not be optimal for all of these goals.

A possible research direction is to add application-aware terms to the joint learn-

國立清華大學
NATIONAL TSING HUA UNIVERSITY

ing objective. For error concealment, the loss can include reconstruction quality after simulated frame drops. For temporal super-resolution, the loss can include temporal consistency or warping error after interpolation. For source coding, the loss can include entropy, quantization error, or bitstream size after encoding. Another direction is to learn different motion-vector representations for different applications while keeping the same key-frame-based structure.

The main challenge is to balance rendered-view quality and application-specific performance. A successful solution should show that application-aware learning improves the target downstream task without causing unacceptable degradation in the learned frame quality.

Future Research Problem 6 (Generalization to Diverse Dynamic Point Cloud Datasets)**.**

Given that dynamic point clouds may contain different object categories, motion patterns, and scene structures, how can we evaluate whether joint learning of point clouds and motion vectors generalizes beyond human avatar sequences?

This problem comes from the scope of the current evaluation. This thesis mainly evaluates the proposed methods on dynamic human point cloud sequences. Human avatars are challenging targets because they often contain non-rigid motion, self-occlusion, changing silhouettes, and fine geometric details. Therefore, showing good performance on these sequences provides strong evidence for the effectiveness of the proposed method. However, other types of dynamic point clouds, such as rigid objects, vehicles, animals, or outdoor scenes, may have different motion properties and visibility changes.

A possible research direction is to evaluate the proposed framework on a broader set of dynamic point cloud datasets. These datasets may include both rigid and non-rigid objects, controlled synthetic sequences, real captured sequences, and scenes with different point densities. The evaluation should compare rendered-view quality, motion-vector smoothness, learning time, and downstream application performance across these categories.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

The main challenge is that different datasets may require different camera settings, point sizes, GoF lengths, and training parameters. A successful solution should show whether the current framework can be directly applied to new datasets, or whether adaptive parameter selection is needed for different object categories and motion types.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

References

[1] Statista. "Augmented and virtual reality market size, share and growth". https://www.statista.com/outlook/amo/ar-vr/worldwide.

[2] MPEG, "G-PCC codec description v12", ISO/IEC JTC 1/SC 29/WG 7 MPEG Output Document, Tech. Rep., 2021.

[3] E. S. Jang et al., "Video-based point-cloud-compression standard in mpeg: From evidence collection to committee draft [standards in a nutshell]", *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 118–123, 2019.

[4] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, *8i voxelized full bodies, version 2 – a voxelized point cloud dataset*, 2017. [Online]. Available: https://plenodb.jpeg.org/pc/8ilabs.

[5] T.-K. Hung, I.-C. Huang, S. R. Cox, W. T. Ooi, and C.-H. Hsu, "Error concealment of dynamic 3d point cloud streaming", in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 3134–3142.

[6] I.-C. Huang, Y. Shi, Y.-C. Sun, W. T. Ooi, C.-Y. Huang, and C.-H. Hsu, "Composing error concealment pipelines for dynamic 3d point cloud streaming", *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 21, no. 6, pp. 1–28, 2025.

[7] I. Viola, J. Mulder, F. De Simone, and P. Cesar, "Temporal interpolation of dynamic digital humans using convolutional neural networks", in *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, IEEE, 2019, pp. 90–907.

[8] F. Lu, G. Chen, S. Qu, Z. Li, Y. Liu, and A. Knoll, "Pointinet: Point cloud frame interpolation network", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 2251–2259.

[9] Y. Zeng, Y. Qian, Q. Zhang, J. Hou, Y. Yuan, and Y. He, "Idea-net: Dynamic 3d point cloud interpolation via deep embedding alignment", in *Proceedings of*

國立清華大學
NATIONAL TSING HUA UNIVERSITY

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 6338–6347.

[10] Z. Zheng, D. Wu, R. Lu, F. Lu, G. Chen, and C. Jiang, "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*, 2023, pp. 909–918.

[11] C. Jiang et al., "Neurogauss4d-pci: 4d neural fields and gaussian deformation fields for point cloud interpolation", *arXiv preprint arXiv:2405.14241*, 2024.

[12] Z. Yan et al., "Rpm-net: Recurrent prediction of motion and parts from point cloud", *arXiv preprint arXiv:2006.14865*, 2020.

[13] F. Lu et al., "Monet: Motion-based point cloud prediction network", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 13 794–13 804, 2021.

[14] T. Zhang, G. Qian, J. Xie, and J. Yang, "Fastpci: Motion-structure guided fast point cloud frame interpolation", in *European Conference on Computer Vision*, Springer, 2024, pp. 251–267.

[15] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering", *ACM Transactions on Graphics*, vol. 42, no. 4, Jul. 2023.

[16] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, "Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis", in *3DV*, 2024.

[17] N. Ravi et al., "Accelerating 3d deep learning with pytorch3d", *arXiv:2007.08501*, 2020.

[18] C. Lassner and M. Zollhofer, "Pulsar: Efficient sphere-based neural rendering", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 1440–1449.

[19] C.-T. Lee et al., "Joint learning of point clouds and motion vectors for volumetric video", in *Proceedings of the 17th International Workshop on IMmersive Mixed and Virtual Environment Systems*, 2025, pp. 1–7.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

[20] P. Kakkar and H. Ragothaman, "The evolution of volumetric video: A survey of smart transcoding and compression approaches", *arXiv preprint arXiv:2411.02095*, 2024.

[21] S. Schwarz et al., "Emerging mpeg standards for point cloud compression", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2018.

[22] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim", in *2010 20th international conference on pattern recognition*, IEEE, 2010, pp. 2366–2369.

[23] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity", *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[24] C.-H. Wu, X. Li, R. Rajesh, W. T. Ooi, and C.-H. Hsu, "Dynamic 3d point cloud streaming: Distortion and concealment", in *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2021, pp. 98–105.

[25] A. L. Souto, R. L. de Queiroz, and C. Dorea, "A 3d motion vector database for dynamic point clouds", *arXiv preprint arXiv:2008.08438*, 2020.

[26] D. Rempe, T. Birdal, Y. Zhao, Z. Gojcic, S. Sridhar, and L. J. Guibas, *Caspr: Learning canonical spatiotemporal point cloud representations*, 2020. arXiv: 2008.02792 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2008.02792.

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

[28] R. He et al., "4drecon: Spatio-temporal reconstruction of multiple dynamic objects in lidar streams", in *2025 9th International Conference on Robotics and Automation Sciences (ICRAS)*, IEEE, 2025, pp. 379–386.

[29] L. Sang, Z. Canfes, D. Cao, R. Marin, F. Bernard, and D. Cremers, "4deform: Neural surface deformation for robust shape interpolation", in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 6542–6551.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

[30] H. Fan, H. Su, and L. J. Guibas, "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*, 2017, pp. 605–613.

[31] G. Shi, C.-C. Wu, and C.-H. Hsu, "Error concealment of dynamic lidar point clouds for connected and autonomous vehicles", in *GLOBECOM 2023-2023 IEEE Global Communications Conference*, IEEE, 2023, pp. 5409–5415.

[32] J. Li, Q. Hu, and M. Ai, "Point cloud registration based on one-point ransac and scale-annealing biweight estimation", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 11, pp. 9716–9729, 2021.

[33] M. Rudolph, A. Riemenschneider, and A. Rizk, "Learned compression of point cloud geometry and attributes in a single model through multimodal rate-control", *arXiv preprint arXiv:2408.00599*, 2024.

[34] T. Huang and G. H. Lee, "Unified geometry and color compression framework for point clouds via generative diffusion priors", *arXiv preprint arXiv:2503.18083*, 2025.

[35] Y. Hu, R. Gong, and Y. Wang, "Bits-to-photon: End-to-end learned scalable point cloud compression for direct rendering", in *2025 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2025, pp. 953–958.

[36] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. De Queiroz, "Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts", *IEEE Transactions on Image Processing*, vol. 29, pp. 313–322, 2019.

[37] Y. Zeng, J. Hou, Q. Zhang, S. Ren, and W. Wang, "Dynamic 3d point cloud sequences as 2d videos", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 9371–9386, 2024.

[38] Y. Zhou, X. Zhang, X. Ma, Y. Xu, K. Zhang, and L. Zhang, "Dynamic point cloud compression with spatio-temporal transformer-style modeling", in *2024 Data Compression Conference (DCC)*, IEEE, 2024, pp. 53–62.

[39] A. Akhtar, Z. Li, and G. Van der Auwera, "Inter-frame compression for dynamic point cloud geometry coding", *IEEE Transactions on Image Processing*, vol. 33, pp. 584–594, 2024.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

[40] ISO/IEC JTC 1/SC 29, *ISO/IEC 23090-3: Versatile Video Coding*, International Standard, 2020.

[41] C. Chao, C. Tulvan, M. Preda, and T. Zaharia, "Skeleton-based motion estimation for point cloud compression", in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2020, pp. 1–6.

[42] A. L. Souto, R. L. De Queiroz, and C. Dorea, "Motion-compensated predictive raht for dynamic point clouds", *IEEE Transactions on Image Processing*, vol. 32, pp. 2428–2437, 2023.

[43] D. T. Nguyen, D. Zieger, M. Stamminger, and A. Kaup, "End-to-end learned lossy dynamic point cloud attribute compression", in *2024 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2024, pp. 3355–3360.

[44] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, 2016.

[45] C. Santos, M. Gonçalves, G. Corrêa, and M. Porto, "Block-based inter-frame prediction for dynamic point cloud compression", in *2021 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2021, pp. 3388–3392.

[46] Y. An, Y. Shao, G. Li, W. Gao, and S. Liu, "A fast motion estimation method with hamming distance for lidar point cloud compression", in *2022 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, IEEE, 2022, pp. 1–5.

[47] H. Hong, E. Pavez, A. Ortega, R. Watanabe, and K. Nonaka, "Motion estimation and filtered prediction for dynamic point cloud attribute compression", in *2022 Picture Coding Symposium (PCS)*, IEEE, 2022, pp. 139–143.

[48] Y. Shao, G. Li, Q. Zhang, W. Gao, and S. Liu, "Nonrigid registration-based progressive motion compensation for point cloud geometry compression", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–14, 2023.

[49] Y. Shao, W. Gao, S. Liu, and G. Li, "Advanced patch-based affine motion estimation for dynamic point cloud geometry compression", *Sensors*, vol. 24, no. 10, p. 3142, 2024.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

[50] X. Mu, W. Gao, H. Yuan, S. Wang, and G. Li, "Adaptive lpu decision for dynamic point cloud compression", *IEEE Signal Processing Letters*, 2024.

[51] T. Fan, L. Gao, Y. Xu, Z. Li, and D. Wang, "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*, L. D. Raedt, Ed., Main Track, International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 898–904. DOI: 10.24963/ijcai.2022/126. [Online]. Available: https://doi.org/10.24963/ijcai.2022/126.

[52] S. Xia, T. Fan, Y. Xu, J.-N. Hwang, and Z. Li, "Learning dynamic point cloud compression via hierarchical inter-frame block matching", in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 7993–8003.

[53] Z. Jiang, G. Wang, G. K. Tam, C. Song, F. W. Li, and B. Yang, "An end-to-end dynamic point cloud geometry compression in latent space", *Displays*, vol. 80, p. 102 528, 2023.

[54] T. Fan, Y. Hu, and Y. Wang, "U-motion: Learned point cloud video compression with u-structured motion estimation", *arXiv e-prints*, arXiv–2411, 2024.

[55] W. Yifan, F. Serena, S. Wu, C. Öztireli, and O. Sorkine-Hornung, "Differentiable surface splatting for point-based geometry processing", *ACM Transactions On Graphics (TOG)*, vol. 38, no. 6, pp. 1–14, 2019.

[56] R. Shaw et al., "Swings: Sliding windows for dynamic 3d gaussian splatting", in *European Conference on Computer Vision*, Springer, 2025, pp. 37–54.

[57] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing", *arXiv:1801.09847*, 2018.

[58] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool", in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, Eds., The Eurographics Association, 2008, ISBN: 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[59] D. Girardeau-Montaut et al., "Cloudcompare", *France: EDF R&D Telecom ParisTech*, vol. 11, no. 5, p. 2016, 2016.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

[60] B. Foundation, *Blender*, 2025. [Online]. Available: http://www.blender.org.

[61] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)", in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 1–4.

[62] S. Liu, W. Chen, T. Li, and H. Li, "Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction", *arXiv preprint arXiv:1901.05567*, 2019.

[63] Google. "Draco 3D data compression". https://github.com/google/draco.

[64] J. Seward, "Bzip2 and libbzip2", *avaliable at http://www. bzip. org*, pp. 8–18, 1996.

[65] W.-S. Lai, J.-B. Huang, O. Wang, E. Shechtman, E. Yumer, and M.-H. Yang, "Learning blind video temporal consistency", in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 170–185.

[66] Y.-C. Sun et al., "Lts: A dash streaming system for dynamic multi-layer 3d gaussian splatting scenes", in *Proceedings of the 16th ACM Multimedia Systems Conference*, 2025, pp. 136–147.

[67] Z. Ding et al., "Eyenavgs: A 6-dof navigation dataset and record-n-replay software for real-world 3dgs scenes in vr", in *Proceedings of the 33rd ACM International Conference on Multimedia*, 2025, pp. 13 126–13 132.

[68] C.-Y. Wu, Y.-C. Sun, C.-T. Lee, and C.-H. Hsu, "Optimally planning drone trajectories to capture 3d gaussian splatting objects", in *International Conference on Multimedia Modeling*, Springer, 2024, pp. 171–185.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Appendices

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Appendix A Other Research Contributions

This appendix briefly summarizes other research projects that I participated in during my graduate study. These works are not the main focus of this thesis, but they are related to immersive media, 3D Gaussian Splatting, and volumetric content systems. They also helped build the background and system experience used in this thesis.

A.1 DASH Streaming for Dynamic 3DGS Scenes

In our work on LTS [66], we studied how to stream dynamic multi-layer 3D Gaussian Splatting scenes through a DASH-based system. The main goal was to support adaptive streaming for dynamic 3DGS content, where the system can adjust the delivered representation according to network and playback conditions. This work focused more on the delivery side of immersive 3D content, while this thesis focuses on learning point clouds and motion vectors for volumetric video. Both works share the same broader goal: making dynamic 3D media easier to store, transmit, and render.

A.2 6-DoF Navigation Dataset for 3DGS Scenes

In EyeNavGS [67], we built a 6-DoF navigation dataset and record-and-replay software for real-world 3DGS scenes in VR. The dataset records user navigation behavior, including head pose and eye gaze, when users explore 3DGS scenes through a VR headset. This work provides data and tools for studying viewport prediction, adaptive streaming, foveated rendering, and user behavior in 3DGS environments. Compared with the main topic of this thesis, EyeNavGS focuses on user navigation and system evaluation, while this thesis focuses on generating learned point clouds and explicit motion vectors. However, both works are connected by the same need to support practical 6-DoF immersive experiences.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

A.3 Drone Trajectory Planning for 3DGS Capture

In our drone trajectory planning work [68], we studied how to plan drone paths for capturing objects that will later be reconstructed with 3D Gaussian Splatting. The goal was to choose useful camera viewpoints so that the captured images can better support high-quality 3DGS reconstruction. This work focuses on the capture stage of 3D content creation. In contrast, this thesis focuses on the processing stage after dynamic point clouds are already available. Together, these works cover different parts of the 3D media pipeline, from data capture to representation learning and downstream applications.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Appendix B Additional Rendered View Results

This appendix provides additional sample rendered non-key frames for each evaluated sequence. For each sequence, the first row shows the GT rendered frames, and the second row shows the rendered frames from our learned point clouds. These examples complement the visual quality results in Sec. 8 by showing frame-level rendered views across different synthetic and real sequences. Fig. B.1, Fig. B.2, Fig. B.3, Fig. B.4, Fig. B.5, and Fig. B.6 show additional rendered-view examples for all evaluated sequences.

(a) GT

(b) Ours

Fig. B.1: Sample rendered frames from man dance: (a) GT and (b) Ours.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

(a) GT

(b) Ours

Fig. B.2: Sample rendered frames from woman kick: (a) GT and (b) Ours.

(a) GT

(b) Ours

Fig. B.3: Sample rendered frames from redandblack: (a) GT and (b) Ours.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

(a) GT

(b) Ours

Fig. B.4: Sample rendered frames from loot: (a) GT and (b) Ours.

(a) GT

(b) Ours

Fig. B.5: Sample rendered frames from longdress: (a) GT and (b) Ours.

國立清華大學
NATIONAL TSING HUA UNIVERSITY

(a) GT

(b) Ours

Fig. B.6: Sample rendered frames from soldier: (a) GT and (b) Ours.